

Introduction

1.1 Scope

The scope of IEEE Std. 1003.1-200x is described in the Base Definitions volume of IEEE Std. 1003.1-200x.

1.2 Conformance

Conformance requirements for IEEE Std. 1003.1-200x are defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 2, Conformance.

1.3 Normative References

Normative references for IEEE Std. 1003.1-200x are defined in the Base Definitions volume of IEEE Std. 1003.1-200x.

1.4 Changes from Issue 4

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

The change history is subject to revision. The intention is to keep change history from Issue 4, and in the Issue 5 to Issue 6 change history to note changes from POSIX.2-1992 as well as Issue 5.

The following sections describe changes made to this volume of IEEE Std. 1003.1-200x since Issue 4. The CHANGE HISTORY section for each utility describes technical changes made to that utility since Issue 4. Changes made between Issue 2 and Issue 4 are not included.

1.4.1 Changes from Issue 4 to Issue 4, Version 2

The following list summarizes the major changes that were made in this volume of IEEE Std. 1003.1-200x from Issue 4 to Issue 4, Version 2:

- The X/Open UNIX extension was added, which specifies the common core utilities of 4.3 Berkeley Software Distribution (4.3 BSD), the OSF AES, and SVID Issue 3.

24 1.4.2 Changes from Issue 4, Version 2 to Issue 5

25 The following list summarizes the major changes that were made in this volume of
26 IEEE Std. 1003.1-200x from Issue 4, Version 2 to Issue 5:

- 27 • Large File Summit (LFS) Extensions were added.
- 28 • Some utilities were updated to reflect changes for the POSIX Realtime Extension.
- 29 • Some utilities were updated to reflect changes for the POSIX Threads Extension.
- 30 • The LEGACY category of utilities was introduced as a replacement for the TO BE
31 WITHDRAWN, WITHDRAWN, and Possibly Unsupportable categories.
- 32 • The following utilities were added:

33 *fuser*
34 *ipcrm*
35 *ipcs*
36 *link*
37 *unlink*

38 1.4.3 Changes from Issue 5 to Issue 6

39 The following list summarizes the major changes that were made in this volume of
40 IEEE Std. 1003.1-200x from Issue 5 to Issue 6:

- 41 • This volume of IEEE Std. 1003.1-200x is extensively revised so it can be both an IEEE POSIX
42 Standard and an Open Group Technical Standard.
- 43 • this volume of IEEE Std. 1003.1-200x is updated to mandate support of FIPS 151-2. The
44 following changes were made:
 - 45 — Support is mandated for the capabilities associated with the following symbolic
46 constants:
47 *_POSIX_CHOWN_RESTRICTED*
48 *_POSIX_JOB_CONTROL*
49 *_POSIX_SAVED_IDS*
 - 50 — In the environment for the login shell, the environment variables *LOGNAME* and *HOME*
51 shall be defined and have the properties described in the Base Definitions volume of
52 IEEE Std. 1003.1-200x, Chapter 7, Locale.
- 53 • this volume of IEEE Std. 1003.1-200x is updated to align with some features of the Single
54 UNIX Specification.
- 55 • A RATIONALE section is added to each reference page.

56 1.5 Terminology

57 This section appears in the Base Definitions volume of IEEE Std. 1003.1-200x, but is repeated
58 here for convenience:

59 For the purposes of IEEE Std. 1003.1-200x, the following terminology definitions apply:

60 **can**

61 Describes a permissible optional feature or behavior available to the user or application. The
62 feature or behavior is mandatory for an implementation that conforms to
63 IEEE Std. 1003.1-200x. An application can rely on the existence of the feature or behavior.

64 **implementation-defined**

65 Describes a value or behavior that is not defined by IEEE Std. 1003.1-200x but is selected by
66 an implementor. The value or behavior may vary among implementations that conform to
67 IEEE Std. 1003.1-200x. An application should not rely on the existence of the value or
68 behavior. An application that relies on such a value or behavior cannot be assured to be
69 portable across conforming implementations.

70 The implementor shall document such a value or behavior so that it can be used correctly
71 by an application.

72 **legacy**

73 Describes a feature or behavior that is being retained for compatibility with older
74 applications, but which has limitations which make it inappropriate for developing portable
75 applications. New applications should use alternative means of obtaining equivalent
76 functionality.

77 **may**

78 Describes a feature or behavior that is optional for an implementation that conforms to
79 IEEE Std. 1003.1-200x. An application should not rely on the existence of the feature or
80 behavior. An application that relies on such a feature or behavior cannot be assured to be
81 portable across conforming implementations.

82 To avoid ambiguity, the opposite of *may* is expressed as *need not*, instead of *may not*.

83 **shall**

84 For an implementation that conforms to IEEE Std. 1003.1-200x, describes a feature or
85 behavior that is mandatory. An application can rely on the existence of the feature or
86 behavior.

87 For an application or user, describes a behavior that is mandatory.

88 **should**

89 For an implementation that conforms to IEEE Std. 1003.1-200x, describes a feature or
90 behavior that is recommended but not mandatory. An application should not rely on the
91 existence of the feature or behavior. An application that relies on such a feature or behavior
92 cannot be assured to be portable across conforming implementations.

93 For an application, describes a feature or behavior that is recommended programming
94 practice for optimum portability.

95 **undefined**

96 Describes the nature of a value or behavior not defined by IEEE Std. 1003.1-200x which
97 results from use of an invalid program construct or invalid data input.

98 The value or behavior may vary among implementations that conform to
99 IEEE Std. 1003.1-200x. An application should not rely on the existence or validity of the
100 value or behavior. An application that relies on any particular value or behavior cannot be

101 assured to be portable across conforming implementations.

102 **unspecified**

103 Describes the nature of a value or behavior not specified by IEEE Std. 1003.1-200x which
104 results from use of a valid program construct or valid data input.

105 The value or behavior may vary among implementations that conform to
106 IEEE Std. 1003.1-200x. An application should not rely on the existence or validity of the
107 value or behavior. An application that relies on any particular value or behavior cannot be
108 assured to be portable across conforming implementations.

109 **1.6 Definitions**

110 Concepts and definitions are defined in the Base Definitions volume of IEEE Std. 1003.1-200x. |

111 1.7 Relationship to Other Documents

112 1.7.1 The System Interfaces volume of IEEE Std. 1003.1-200x

113 This subsection describes some of the features provided by the System Interfaces volume of
 114 IEEE Std. 1003.1-200x that are assumed to be globally available by all systems conforming to this
 115 volume of IEEE Std. 1003.1-200x. This subsection does not attempt to detail all of the features
 116 defined in the System Interfaces volume of IEEE Std. 1003.1-200x that are required by all of the
 117 utilities defined in this volume of IEEE Std. 1003.1-200x; the utility and function descriptions
 118 point out additional functionality required to provide the corresponding specific features
 119 needed by each.

120 The following subsections describe frequently used concepts. Many of these concepts are
 121 described in the Base Definitions volume of IEEE Std. 1003.1-200x. Utility and function
 122 description statements override these defaults when appropriate.

123 1.7.1.1 Process Attributes

124 The following process attributes, as described in the System Interfaces volume of
 125 IEEE Std. 1003.1-200x, are assumed to be supported for all processes in this volume of
 126 IEEE Std. 1003.1-200x:

127	Controlling Terminal	Real Group ID
128	Current Working Directory	Real User ID
129	Effective Group ID	Root Directory
130	Effective User ID	Saved Set-Group-ID
131	File Descriptors	Saved Set-User-ID
132	File Mode Creation Mask	Session Membership
133	Process Group ID	Supplementary Group IDs
134	Process ID	

135 A conforming implementation may include additional process attributes.

136 1.7.1.2 Concurrent Execution of Processes

137 The following functionality of the *fork()* function defined in the System Interfaces volume of
 138 IEEE Std. 1003.1-200x shall be available on all systems conforming to this volume of
 139 IEEE Std. 1003.1-200x:

- 140 1. Independent processes shall be capable of executing independently without either process
 141 terminating.
- 142 2. A process shall be able to create a new process with all of the attributes referenced in
 143 Section 1.7.1.1, determined according to the semantics of a call to the *fork()* function
 144 defined in the System Interfaces volume of IEEE Std. 1003.1-200x followed by a call in the
 145 child process to one of the *exec* functions defined in the System Interfaces volume of
 146 IEEE Std. 1003.1-200x.

147 1.7.1.3 File Access Permissions

148 The file access control mechanism described by the Base Definitions volume of
 149 IEEE Std. 1003.1-200x, Section 4.1, File Access Permissions applies to all files on an
 150 implementation conforming to this volume of IEEE Std. 1003.1-200x.

151 1.7.1.4 File Read, Write, and Creation

152 If a file that does not exist is to be written, it shall be created as described below, unless the
153 utility description states otherwise.

154 When a file that does not exist is created, the following features defined in the System Interfaces
155 volume of IEEE Std. 1003.1-200x shall apply unless the utility or function description states
156 otherwise:

- 157 1. The user ID of the file is set to the effective user ID of the calling process.
- 158 2. The group ID of the file is set to the effective group ID of the calling process or the group
159 ID of the directory in which the file is being created.
- 160 3. If the file is a regular file, the permission bits of the file are set to:
161 S_IROTH | S_IWOTH | S_IRGRP | S_IWGRP | S_IRUSR | S_IWUSR
162 (see the description of *File Modes* in the Base Definitions volume of IEEE Std. 1003.1-200x,
163 Chapter 13, Headers, <sys/stat.h>) except that the bits specified by the file mode creation
164 mask of the process are cleared. If the file is a directory, the permission bits are set to:
165 S_IRWXU | S_IRWXG | S_IRWXO
166 except that the bits specified by the file mode creation mask of the process are cleared.
- 167 4. The *st_atime*, *st_ctime*, and *st_mtime* fields of the file shall be updated as specified in the
168 System Interfaces volume of IEEE Std. 1003.1-200x, Section 2.5, Standard I/O Streams.
- 169 5. If the file is a directory, it shall be an empty directory; otherwise, the file shall have length
170 zero.
- 171 6. If the file is a symbolic link, the effect shall be undefined unless the {POSIX2_SYMLINKS}
172 variable is in effect for the directory in which the symbolic link would be created.
- 173 7. Unless otherwise specified, the file created shall be a regular file.

174 When an attempt is made to create a file that already exists, the action shall depend on the file
175 type:

- 176 1. For directories and FIFO special files, the attempt shall fail and the utility shall either
177 continue with its operation or exit immediately with a non-zero status, depending on the
178 description of the utility.
- 179 2. For regular files:
 - 180 a. The user ID, group ID, and permission bits of the file shall not be changed.
 - 181 b. The file shall be truncated to zero length.
 - 182 c. The *st_ctime* and *st_mtime* fields shall be marked for update.
- 183 3. For other file types, the effect is implementation-defined.

184 When a file is to be appended, the file shall be opened in a manner equivalent to using the
185 O_APPEND flag, without the O_TRUNC flag, in the *open()* function defined in the System
186 Interfaces volume of IEEE Std. 1003.1-200x.

187 When a file is to be read or written, the file shall be opened with an access mode corresponding
188 to the operation to be performed. If file access permissions deny access, the requested operation
189 shall fail.

190 1.7.1.5 *File Removal*

191 When a directory that is the root directory or current working directory of any process is
 192 removed, the effect is implementation-defined. If file access permissions deny access, the
 193 requested operation fails. Otherwise, when a file is removed:

- 194 1. Its directory entry is removed from the file system.
- 195 2. The link count of the file is decremented.
- 196 3. If the file is an empty directory (see the Base Definitions volume of IEEE Std. 1003.1-200x,
 197 Section 3.145, Empty Directory):
 - 198 a. If no process has the directory open, the space occupied by the directory is freed and
 199 the directory is no longer accessible.
 - 200 b. If one or more processes have the directory open, the directory contents are
 201 preserved until all references to the file have been closed.
- 202 4. If the file is a directory that is not empty, the *st_ctime* field is marked for update.
- 203 5. If the file is not a directory:
 - 204 a. If the link count becomes zero:
 - 205 i. If no process has the file open, the space occupied by the file is freed and the
 206 file is no longer accessible.
 - 207 ii. If one or more processes have the file open, the file contents are preserved until
 208 all references to the file have been closed.
 - 209 b. If the link count is not reduced to zero, the *st_ctime* field is marked for update.
- 210 6. The *st_ctime* and *st_mtime* fields of the containing directory are marked for update.

211 1.7.1.6 *File Time Values*

212 All files shall have the three time values described by the Base Definitions volume of
 213 IEEE Std. 1003.1-200x, Section 4.3, File Times Update.

214 1.7.1.7 *File Contents*

215 When a reference is made to the contents of a file, *pathname*, this means the equivalent of all of
 216 the data placed in the space pointed to by *buf* when performing the *read()* function calls in the
 217 following operations defined in the System Interfaces volume of IEEE Std. 1003.1-200x:

```
218     while (read (fildes, buf, nbytes) > 0)
219         ;
```

220 If the file is indicated by a path name *pathname*, the file descriptor shall be determined by the
 221 equivalent of the following operation defined in the System Interfaces volume of
 222 IEEE Std. 1003.1-200x:

```
223     fildes = open (pathname, O_RDONLY);
```

224 The value of *nbytes* in the above sequence is unspecified; if the file is of a type where the data
 225 returned by *read()* would vary with different values, the value is one that results in the most
 226 data being returned.

227 If the *read()* function calls would return an error, it is unspecified whether the contents of the file
 228 are considered to include any data from offsets in the file beyond where the error would be
 229 returned.

230 1.7.1.8 *Path Name Resolution*

231 The path name resolution algorithm, described by the Base Definitions volume of
232 IEEE Std. 1003.1-200x, Section 4.5, Path Name Resolution, is used by implementations
233 conforming to this volume of IEEE Std. 1003.1-200x; see also the Base Definitions volume of
234 IEEE Std. 1003.1-200x, Section 4.4, File Hierarchy.

235 1.7.1.9 *Changing the Current Working Directory*

236 When the current working directory (see the Base Definitions volume of IEEE Std. 1003.1-200x,
237 Section 3.438, Working Directory) is to be changed, unless the utility or function description
238 states otherwise, the operation shall succeed unless a call to the *chdir()* function defined in the
239 System Interfaces volume of IEEE Std. 1003.1-200x would fail when invoked with the new
240 working directory path name as its argument.

241 1.7.1.10 *Establish the Locale*

242 The functionality of the *setlocale()* function defined in the System Interfaces volume of
243 IEEE Std. 1003.1-200x is assumed to be available on all systems conforming to this volume of
244 IEEE Std. 1003.1-200x; that is, utilities that require the capability of establishing an international
245 operating environment shall be permitted to set the specified category of the international
246 environment.

247 1.7.1.11 *Actions Equivalent to Functions*

248 Some utility descriptions specify that a utility performs actions equivalent to a function defined
249 in the System Interfaces volume of IEEE Std. 1003.1-200x. Such specifications require only that
250 the external effects be equivalent, not that any effect within the utility and visible only to the
251 utility be equivalent.

252 1.8 Portability

253 Some of the utilities in the Shell and Utilities volume of IEEE Std. 1003.1-200x and functions in
254 the System Interfaces volume of IEEE Std. 1003.1-200x describe functionality that might not be
255 fully portable to systems meeting the requirements for POSIX conformance (see the Base
256 Definitions volume of IEEE Std. 1003.1-200x, Chapter 2, Conformance).

257 Where optional, enhanced, or reduced functionality is specified, the text is shaded and a code in
258 the margin identifies the nature of the option, extension, or warning (see Section 1.8.1). For
259 maximum portability, an application should avoid such functionality.

260 Unless the primary task of a utility is to produce textual material on its standard output,
261 application developers should not rely on the format or content of any such material that may be
262 produced. Where the primary task *is* to provide such material, but the output format is
263 incompletely specified, the description is marked with the OF margin code and shading.
264 Application developers are warned not to expect that the output of such an interface on one
265 system is any guide to its behavior on another system.

266 1.8.1 Codes

267 Codes and their meanings are listed in the Base Definitions volume of IEEE Std. 1003.1-200x, but
268 are repeated here for convenience:

269 ADV **Advisory Information**

270 The functionality described is optional. The functionality described is also an extension to the
271 ISO C standard.

272 Where applicable, functions are marked with the ADV margin legend in the SYNOPSIS section.
273 Where additional semantics apply to a function, the material is identified by use of the ADV
274 margin legend.

275 AIO **Asynchronous Input and Output**

276 The functionality described is optional. The functionality described is also an extension to the
277 ISO C standard.

278 Where applicable, functions are marked with the AIO margin legend in the SYNOPSIS section.
279 Where additional semantics apply to a function, the material is identified by use of the AIO
280 margin legend.

281 BAR **Barriers**

282 The functionality described is optional. The functionality described is also an extension to the
283 ISO C standard.

284 Where applicable, functions are marked with the BAR margin legend in the SYNOPSIS section.
285 Where additional semantics apply to a function, the material is identified by use of the BAR
286 margin legend.

287 BE **Batch Environment Services and Utilities**

288 The functionality described is optional.

289 Where applicable, utilities are marked with the BE margin legend in the SYNOPSIS section.
290 Where additional semantics apply to a utility, the material is identified by use of the BE margin
291 legend.

292 CD **C-Language Development Utilities**

293 The functionality described is optional.

294 Where applicable, utilities are marked with the CD margin legend in the SYNOPSIS section.
295 Where additional semantics apply to a utility, the material is identified by use of the CD margin

296 legend.

297 CPT **Process CPU-Time Clocks**
298 The functionality described is optional. The functionality described is also an extension to the
299 ISO C standard.

300 Where applicable, functions are marked with the CPT margin legend in the SYNOPSIS section.
301 Where additional semantics apply to a function, the material is identified by use of the CPT
302 margin legend.

303 CS **Clock Selection**
304 The functionality described is optional. The functionality described is also an extension to the
305 ISO C standard.

306 Where applicable, functions are marked with the CS margin legend in the SYNOPSIS section.
307 Where additional semantics apply to a function, the material is identified by use of the CS
308 margin legend.

309 CX **Extension to the ISO C standard**
310 The functionality described is an extension to the ISO C standard. Application writers may
311 make use of an extension as it is supported on all IEEE Std. 1003.1-200x-conforming systems.

312 FD **FORTRAN Development Utilities**
313 The functionality described is optional.

314 Where applicable, utilities are marked with the FD margin legend in the SYNOPSIS section.
315 Where additional semantics apply to a utility, the material is identified by use of the FD margin
316 legend.

317 FR **FORTRAN Runtime Utilities**
318 The functionality described is optional.

319 Where applicable, utilities are marked with the FR margin legend in the SYNOPSIS section.
320 Where additional semantics apply to a utility, the material is identified by use of the FR margin
321 legend.

322 FSC **File Synchronization**
323 The functionality described is optional. The functionality described is also an extension to the
324 ISO C standard.

325 Where applicable, functions are marked with the FSC margin legend in the SYNOPSIS section.
326 Where additional semantics apply to a function, the material is identified by use of the FSC
327 margin legend.

328 IP6 **IPV6**
329 The functionality described is optional. The functionality described is also an extension to the
330 ISO C standard.

331 Where applicable, functions are marked with the IP6 margin legend in the SYNOPSIS section.
332 Where additional semantics apply to a function, the material is identified by use of the IP6
333 margin legend.

334 MAN **Mandatory in the Next Draft**
335 This is an interim draft code used to aid reviewers during the development of
336 IEEE Std. 1003.1-200x. It denotes a feature that was previously an option or extension that is
337 being brought into the mandatory base functionality. This margin code will be removed from the
338 final draft.

339 MF **Memory Mapped Files**
340 The functionality described is optional. The functionality described is also an extension to the

341 ISO C standard.

342 Where applicable, functions are marked with the MF margin legend in the SYNOPSIS section.
343 Where additional semantics apply to a function, the material is identified by use of the MF
344 margin legend.

345 ML **Process Memory Locking**
346 The functionality described is optional. The functionality described is also an extension to the
347 ISO C standard.

348 Where applicable, functions are marked with the ML margin legend in the SYNOPSIS section.
349 Where additional semantics apply to a function, the material is identified by use of the ML
350 margin legend.

351 MLR **Range Memory Locking**
352 The functionality described is optional. The functionality described is also an extension to the
353 ISO C standard.

354 Where applicable, functions are marked with the MLR margin legend in the SYNOPSIS section.
355 Where additional semantics apply to a function, the material is identified by use of the MLR
356 margin legend.

357 MON **Monotonic Clock**
358 The functionality described is optional. The functionality described is also an extension to the
359 ISO C standard.

360 Where applicable, functions are marked with the MON margin legend in the SYNOPSIS section.
361 Where additional semantics apply to a function, the material is identified by use of the MON
362 margin legend.

363 MPR **Memory Protection**
364 The functionality described is optional. The functionality described is also an extension to the
365 ISO C standard.

366 Where applicable, functions are marked with the MPR margin legend in the SYNOPSIS section.
367 Where additional semantics apply to a function, the material is identified by use of the MPR
368 margin legend.

369 MSG **Message Passing**
370 The functionality described is optional. The functionality described is also an extension to the
371 ISO C standard.

372 Where applicable, functions are marked with the MSG margin legend in the SYNOPSIS section.
373 Where additional semantics apply to a function, the material is identified by use of the MSG
374 margin legend.

375 OB **Obsolescent**
376 The functionality described may be withdrawn in a future version of this volume of
377 IEEE Std. 1003.1-200x. Strictly Conforming POSIX Applications and Strictly Conforming XSI
378 Applications shall not use obsolescent features.

379 OF **Output Format Incompletely Specified**
380 The functionality described is an XSI extension. The format of the output produced by the utility
381 is not fully specified. It is therefore not possible to post-process this output in a consistent
382 fashion. Typical problems include unknown length of strings and unspecified field delimiters.

383 OH **Optional Header**
384 In the SYNOPSIS section of some interfaces in the System Interfaces volume of
385 IEEE Std. 1003.1-200x an included header is marked as in the following example:

```
386 OH      #include <sys/types.h>
387          #include <grp.h>
388          struct group *getgrnam(const char *name);
```

389 This indicates that the marked header is not required on XSI-conformant systems.

390 PIO **Prioritized Input and Output**

391 The functionality described is optional. The functionality described is also an extension to the
392 ISO C standard.

393 Where applicable, functions are marked with the PIO margin legend in the SYNOPSIS section.
394 Where additional semantics apply to a function, the material is identified by use of the PIO
395 margin legend.

396 PS **Process Scheduling**

397 The functionality described is optional. The functionality described is also an extension to the
398 ISO C standard.

399 Where applicable, functions are marked with the PS margin legend in the SYNOPSIS section.
400 Where additional semantics apply to a function, the material is identified by use of the PS
401 margin legend.

402 RTS **Realtime Signals Extension**

403 The functionality described is optional. The functionality described is also an extension to the
404 ISO C standard.

405 Where applicable, functions are marked with the RTS margin legend in the SYNOPSIS section.
406 Where additional semantics apply to a function, the material is identified by use of the RTS
407 margin legend.

408 SD **Software Development Utilities**

409 The functionality described is optional.

410 Where applicable, utilities are marked with the SD margin legend in the SYNOPSIS section.
411 Where additional semantics apply to a utility, the material is identified by use of the SD margin
412 legend.

413 SEM **Semaphores**

414 The functionality described is optional. The functionality described is also an extension to the
415 ISO C standard.

416 Where applicable, functions are marked with the SEM margin legend in the SYNOPSIS section.
417 Where additional semantics apply to a function, the material is identified by use of the SEM
418 margin legend.

419 SHM **Shared Memory Objects**

420 The functionality described is optional. The functionality described is also an extension to the
421 ISO C standard.

422 Where applicable, functions are marked with the SHM margin legend in the SYNOPSIS section.
423 Where additional semantics apply to a function, the material is identified by use of the SHM
424 margin legend.

425 SIO **Synchronized Input and Output**

426 The functionality described is optional. The functionality described is also an extension to the
427 ISO C standard.

428 Where applicable, functions are marked with the SIO margin legend in the SYNOPSIS section.
429 Where additional semantics apply to a function, the material is identified by use of the SIO
430 margin legend.

431	SPI	Spin Locks
432		The functionality described is optional. The functionality described is also an extension to the
433		ISO C standard.
434		Where applicable, functions are marked with the SPI margin legend in the SYNOPSIS section.
435		Where additional semantics apply to a function, the material is identified by use of the SPI
436		margin legend.
437	SPN	Spawn
438		The functionality described is optional. The functionality described is also an extension to the
439		ISO C standard.
440		Where applicable, functions are marked with the SPN margin legend in the SYNOPSIS section.
441		Where additional semantics apply to a function, the material is identified by use of the SPN
442		margin legend.
443	SS	Process Sporadic Server
444		The functionality described is optional. The functionality described is also an extension to the
445		ISO C standard.
446		Where applicable, functions are marked with the SS margin legend in the SYNOPSIS section.
447		Where additional semantics apply to a function, the material is identified by use of the SS
448		margin legend.
449	TCT	Thread CPU-Time Clocks
450		The functionality described is optional. The functionality described is also an extension to the
451		ISO C standard.
452		Where applicable, functions are marked with the TCT margin legend in the SYNOPSIS section.
453		Where additional semantics apply to a function, the material is identified by use of the TCT
454		margin legend.
455	THR	Threads
456		The functionality described is optional. The functionality described is also an extension to the
457		ISO C standard.
458		Where applicable, functions are marked with the THR margin legend in the SYNOPSIS section.
459		Where additional semantics apply to a function, the material is identified by use of the THR
460		margin legend.
461	TMO	Timeouts
462		The functionality described is optional. The functionality described is also an extension to the
463		ISO C standard.
464		Where applicable, functions are marked with the TMO margin legend in the SYNOPSIS section.
465		Where additional semantics apply to a function, the material is identified by use of the TMO
466		margin legend.
467	TMR	Timers
468		The functionality described is optional. The functionality described is also an extension to the
469		ISO C standard.
470		Where applicable, functions are marked with the TMR margin legend in the SYNOPSIS section.
471		Where additional semantics apply to a function, the material is identified by use of the TMR
472		margin legend.
473	TPI	Threads Priority Inheritance
474		The functionality described is optional. The functionality described is also an extension to the
475		ISO C standard.

476 Where applicable, functions are marked with the TPI margin legend in the SYNOPSIS section.
477 Where additional semantics apply to a function, the material is identified by use of the TPI
478 margin legend.

479 TPP **Thread Priority Protection**
480 The functionality described is optional. The functionality described is also an extension to the
481 ISO C standard.

482 Where applicable, functions are marked with the TPP margin legend in the SYNOPSIS section.
483 Where additional semantics apply to a function, the material is identified by use of the TPP
484 margin legend.

485 TPS **Thread Execution Scheduling**
486 The functionality described is optional. The functionality described is also an extension to the
487 ISO C standard.

488 Where applicable, functions are marked with the TPS margin legend for the SYNOPSIS section.
489 Where additional semantics apply to a function, the material is identified by use of the TPS
490 margin legend.

491 TRC **Trace**
492 The functionality described is optional. The functionality described is also an extension to the
493 ISO C standard.

494 Where applicable, functions are marked with the TRC margin legend in the SYNOPSIS section.
495 Where additional semantics apply to a function, the material is identified by use of the TRC
496 margin legend.

497 TEF **Trace Event Filter**
498 The functionality described is optional. The functionality described is also an extension to the
499 ISO C standard.

500 Where applicable, functions are marked with the TEF margin legend in the SYNOPSIS section.
501 Where additional semantics apply to a function, the material is identified by use of the TEF
502 margin legend.

503 TRL **Trace Log**
504 The functionality described is optional. The functionality described is also an extension to the
505 ISO C standard.

506 Where applicable, functions are marked with the TRL margin legend in the SYNOPSIS section.
507 Where additional semantics apply to a function, the material is identified by use of the TRL
508 margin legend.

509 TRI **Trace Inherit**
510 The functionality described is optional. The functionality described is also an extension to the
511 ISO C standard.

512 Where applicable, functions are marked with the TRI margin legend in the SYNOPSIS section.
513 Where additional semantics apply to a function, the material is identified by use of the TRI
514 margin legend.

515 TSA **Thread Stack Address Attribute**
516 The functionality described is optional. The functionality described is also an extension to the
517 ISO C standard.

518 Where applicable, functions are marked with the TPS margin legend for the SYNOPSIS section.
519 Where additional semantics apply to a function, the material is identified by use of the TSA
520 margin legend.

- 521 TSF **Thread-Safe Functions**
522 The functionality described is optional. The functionality described is also an extension to the
523 ISO C standard.
- 524 Where applicable, functions are marked with the TSF margin legend in the SYNOPSIS section.
525 Where additional semantics apply to a function, the material is identified by use of the TSF
526 margin legend.
- 527 TSH **Thread Process-Shared Synchronization**
528 The functionality described is optional. The functionality described is also an extension to the
529 ISO C standard.
- 530 Where applicable, functions are marked with the TSH margin legend in the SYNOPSIS section.
531 Where additional semantics apply to a function, the material is identified by use of the TSH
532 margin legend.
- 533 TSP **Thread Sporadic Server**
534 The functionality described is optional. The functionality described is also an extension to the
535 ISO C standard.
- 536 Where applicable, functions are marked with the TSP margin legend in the SYNOPSIS section.
537 Where additional semantics apply to a function, the material is identified by use of the TSP
538 margin legend.
- 539 TSS **Thread Stack Address Size**
540 The functionality described is optional. The functionality described is also an extension to the
541 ISO C standard.
- 542 Where applicable, functions are marked with the TSS margin legend in the SYNOPSIS section.
543 Where additional semantics apply to a function, the material is identified by use of the TSS
544 margin legend.
- 545 TYM **Typed Memory Objects**
546 The functionality described is optional. The functionality described is also an extension to the
547 ISO C standard.
- 548 Where applicable, functions are marked with the TYM margin legend in the SYNOPSIS section.
549 Where additional semantics apply to a function, the material is identified by use of the TYM
550 margin legend.
- 551 UN **Possibly Unsupportable Feature**
552 The functionality described is an XSI extension. It need not be possible to implement the
553 required functionality (as defined) on all conformant systems and the functionality need not be
554 present. This may, for example, be the case where the conformant system is hosted and the
555 underlying system provides the service in an alternative way.
- 556 UP **User Portability Utilities**
557 The functionality described is optional.
- 558 Where applicable, utilities are marked with the UP margin legend in the SYNOPSIS section.
559 Where additional semantics apply to a utility, the material is identified by use of the UP margin
560 legend.
- 561 XSI **Extension**
562 The functionality described is an XSI extension. Functionality marked XSI is also an extension to
563 the ISO C standard. Application writers may confidently make use of an extension on all
564 systems supporting the X/Open System Interfaces Extension.

565 If an entire SYNOPSIS section is shaded and marked with one XSI, all the functionality described
566 in that reference page is an extension. See the Base Definitions volume of IEEE Std. 1003.1-200x,
567 Section 3.441, XSI.

568 XSR **XSI STREAMS**

569 The functionality described is optional. The functionality described is also an extension to the
570 ISO C standard.

571 Where applicable, functions are marked with the XSR margin legend in the SYNOPSIS section.
572 Where additional semantics apply to a function, the material is identified by use of the XSR
573 margin legend.

574 **1.9 Utility Limits**

575 This section lists magnitude limitations imposed by a specific implementation. The braces
 576 notation, {LIMIT}, is used in this volume of IEEE Std. 1003.1-200x to indicate these values, but
 577 the braces are not part of the name.

578 **Table 1-1 Utility Limit Minimum Values**

Name	Description	Value
{POSIX2_BC_BASE_MAX}	The maximum <i>obase</i> value allowed by the <i>bc</i> utility.	99
{POSIX2_BC_DIM_MAX}	The maximum number of elements permitted in an array by the <i>bc</i> utility.	2048
{POSIX2_BC_SCALE_MAX}	The maximum <i>scale</i> value allowed by the <i>bc</i> utility.	99
{POSIX2_BC_STRING_MAX}	The maximum length of a string constant accepted by the <i>bc</i> utility.	1000
{POSIX2_COLL_WEIGHTS_MAX}	The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> order keyword in the locale definition file; see the border_start keyword in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3.2, <i>LC_COLLATE</i> .	2
{POSIX2_EXPR_NEST_MAX}	The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.	32
{POSIX2_LINE_MAX}	Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing newline.	2048
{POSIX2_RE_DUP_MAX}	The maximum number of repeated occurrences of a BRE permitted when using the interval notation $\{m,n\}$; see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3.6, BREs Matching Multiple Characters.	255
{POSIX2_VERSION}	This value indicates the version of the utilities in this volume of IEEE Std. 1003.1-200x that are provided by the implementation. It changes with each published version.	199209

610 The values specified in Table 1-1 represent the lowest values conforming implementations shall
 611 provide and, consequently, the largest values on which an application can rely without further
 612 enquiries, as described below. These values shall be accessible to applications via the *getconf*
 613 utility (see *getconf* (on page 2692)) and through the *sysconf()* function defined in the System
 614 Interfaces volume of IEEE Std. 1003.1-200x. The literal names shown in Table 1-1 apply only to
 615 the *getconf* utility; the high-level language binding describes the exact form of each name to be
 616 used by the interfaces in that binding.

617 Implementations may provide more liberal, or less restrictive, values than shown in Table 1-1.
 618 These possibly more liberal values are accessible using the symbols in Table 1-2 (on page 2221).

619 The *sysconf()* function defined in the System Interfaces volume of IEEE Std. 1003.1-200x or the
 620 *getconf* utility return the value of each symbol on each specific implementation. The value so

621 retrieved is the largest, or most liberal, value that is available throughout the session lifetime, as
 622 determined at session creation. The literal names shown in the table apply only to the *getconf*
 623 utility; the high-level language binding describes the exact form of each name to be used by the
 624 interfaces in that binding.

625 All numeric limits defined by the System Interfaces volume of IEEE Std. 1003.1-200x, such as
 626 {PATH_MAX}, also apply to this volume of IEEE Std. 1003.1-200x. All the utilities defined by this
 627 volume of IEEE Std. 1003.1-200x are implicitly limited by these values, unless otherwise noted in
 628 the utility descriptions.

629 It is not guaranteed that the application can actually reach the specified limit of an
 630 implementation in any given case, or at all, as a lack of virtual memory or other resources may
 631 prevent this. The limit value indicates only that the implementation does not specifically impose
 632 any arbitrary, more restrictive limit.

633 **Table 1-2** Symbolic Utility Limits

Name	Description	Minimum Value
{BC_BASE_MAX}	The maximum <i>obase</i> value allowed by the <i>bc</i> utility.	{POSIX2_BC_BASE_MAX}
{BC_DIM_MAX}	The maximum number of elements permitted in an array by the <i>bc</i> utility.	{POSIX2_BC_DIM_MAX}
{BC_SCALE_MAX}	The maximum <i>scale</i> value allowed by the <i>bc</i> utility.	{POSIX2_BC_SCALE_MAX}
{BC_STRING_MAX}	The maximum length of a string constant accepted by the <i>bc</i> utility.	{POSIX2_BC_STRING_MAX}
{COLL_WEIGHTS_MAX}	The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> order keyword in the locale definition file; see the order_start keyword in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3.2, <i>LC_COLLATE</i> .	{POSIX2_COLL_WEIGHTS_MAX}
{EXPR_NEST_MAX}	The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.	{POSIX2_EXPR_NEST_MAX}
{LINE_MAX}	Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the	{POSIX2_LINE_MAX}

669
670
671
672
673
674
675
676
677
678
679
680
681

Name	Description	Minimum Value
{RE_DUP_MAX}	trailing newline. The maximum number of repeated occurrences of a BRE permitted when using the interval notation $\{m,n\}$; see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3.6, BREs Matching Multiple Characters.	{POSIX2_RE_DUP_MAX}

682
683

The following value may be a constant within an implementation or may vary from one path name to another.

684
685
686

{POSIX2_SYMLINKS}

When referring to a directory, the system supports the creation of symbolic links within that directory; for non-directory files, the meaning of {POSIX2_SYMLINKS} is undefined.

687 1.10 Grammar Conventions

688 Portions of this volume of IEEE Std. 1003.1-200x are expressed in terms of a special grammar
689 notation. It is used to portray the complex syntax of certain program input. The grammar is
690 based on the syntax used by the *yacc* utility. However, it does not represent fully functional *yacc*
691 input, suitable for program use; the lexical processing and all semantic requirements are
692 described only in textual form. The grammar is not based on source used in any traditional
693 implementation and has not been tested with the semantic code that would normally be
694 required to accompany it. Furthermore, there is no implication that the partial *yacc* code
695 presented represents the most efficient, or only, means of supporting the complex syntax within
696 the utility. Implementations may use other programming languages or algorithms, as long as the
697 syntax supported is the same as that represented by the grammar.

698 The following typographical conventions are used in the grammar; they have no significance
699 except to aid in reading.

- 700 • The identifiers for the reserved words of the language are shown with a leading capital letter.
701 (These are terminals in the grammar; for example, **While**, **Case**.)
- 702 • The identifiers for terminals in the grammar are all named with uppercase letters and
703 underscores; for example, **NEWLINE**, **ASSIGN_OP**, **NAME**.
- 704 • The identifiers for non-terminals are all lowercase.

705 1.11 Utility Description Defaults

706 This section describes all of the subsections used within the utility descriptions, including:

- 707 • Intended usage of the section
- 708 • Global defaults that affect all the standard utilities
- 709 • The meanings of notations used in this volume of IEEE Std. 1003.1-200x that are specific to
- 710 individual utility sections

711 Integer variables and constants, including the values of operands and option-arguments, used
712 by the utilities listed in this volume of IEEE Std. 1003.1-200x shall be implemented as equivalent
713 to the ISO C standard **signed long** data type. Conversion between types shall be as described in
714 the ISO C standard. The evaluation of arithmetic expressions shall be equivalent to that
715 described in Section 6.3 of the ISO C standard.

716 NAME

717 This section gives the name or names of the utility and briefly states its purpose.

718 SYNOPSIS

719 The SYNOPSIS section summarizes the syntax of the calling sequence for the utility,
720 including options, option-arguments, and operands. Standards for utility naming are
721 described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility
722 Syntax Guidelines; for describing the utility's arguments in the Base Definitions volume
723 of IEEE Std. 1003.1-200x, Section 12.1, Utility Argument Syntax.

724 DESCRIPTION

725 The DESCRIPTION section describes the actions of the utility. If the utility has a very
726 complex set of subcommands or its own procedural language, an EXTENDED
727 DESCRIPTION section is also provided. Most explanations of optional functionality are
728 omitted here, as they are usually explained in the OPTIONS section.

729 Some utilities in this volume of IEEE Std. 1003.1-200x are described in terms of
730 functionality equivalent to the System Interfaces volume of IEEE Std. 1003.1-200x.
731 When specific functions are cited, the underlying operating system provides equivalent
732 functionality and all side effects associated with successful execution of the function.
733 The treatment of errors and intermediate results from the individual functions cited is
734 generally not specified by this volume of IEEE Std. 1003.1-200x. See the utility's EXIT
735 STATUS and CONSEQUENCES OF ERRORS sections for all actions associated with
736 errors encountered by the utility.

737 OPTIONS

738 The OPTIONS section describes the utility options and option-arguments, and how
739 they modify the actions of the utility. Standard utilities that have options either fully
740 comply with the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility
741 Syntax Guidelines or describe all deviations. Apparent disagreements between
742 functionality descriptions in the OPTIONS and DESCRIPTION (or EXTENDED
743 DESCRIPTION) sections are always resolved in favor of the OPTIONS section.

744 Each OPTIONS section that uses the phrase "The ... utility shall conform to the Utility
745 Syntax Guidelines ..." refers only to the use of the utility as specified by this volume of
746 IEEE Std. 1003.1-200x; implementation extensions should also conform to the
747 guidelines, but may allow exceptions for historical practice.

748 Unless otherwise stated in the utility description, when given an option unrecognized
749 by the implementation, or when a required option-argument is not provided, standard
750 utilities shall issue a diagnostic message to standard error and exit with a non-zero exit

751 status.

752 XSI All utilities in this volume of IEEE Std. 1003.1-200x shall be capable of processing
753 arguments using 8-bit transparency.

754 **Default Behavior:** When this section is listed as “None.”, it means that the
755 implementation need not support any options. Standard utilities that do not accept
756 options, but that do accept operands, shall recognize “—” as a first argument to be
757 discarded.

758 The requirement for recognizing “—” is because portable applications need a way to
759 shield their operands from any arbitrary options that the implementation may provide
760 as an extension. For example, if the standard utility *foo* is listed as taking no options,
761 and the application needed to give it a path name with a leading hyphen, it could safely
762 do it as:

```
763     foo -- -myfile
```

764 and avoid any problems with `-m` used as an extension.

765 OPERANDS

766 The OPERANDS section describes the utility operands, and how they affect the actions
767 of the utility. Apparent disagreements between functionality descriptions in the
768 OPERANDS and DESCRIPTION (or EXTENDED DESCRIPTION) sections shall be
769 resolved in favor of the OPERANDS section.

770 If an operand naming a file can be specified as ‘-’, which means to use the standard
771 input instead of a named file, this is explicitly stated in this section. Unless otherwise
772 stated, the use of multiple instances of ‘-’ to mean standard input in a single
773 command produces unspecified results.

774 Unless otherwise stated, the standard utilities that accept operands shall process those
775 operands in the order specified in the command line.

776 **Default Behavior:** When this section is listed as “None.”, it means that the
777 implementation need not support any operands.

778 STDIN

779 The STDIN section describes the standard input of the utility. This section is frequently
780 merely a reference to the following section, as many utilities treat standard input and
781 input files in the same manner. Unless otherwise stated, all restrictions described in the
782 INPUT FILES section shall apply to this section as well.

783 Use of a terminal for standard input can cause any of the standard utilities that read
784 standard input to stop when used in the background. For this reason, applications
785 should not use interactive features in scripts to be placed in the background.

786 The specified standard input format of the standard utilities shall not depend on the
787 existence or value of the environment variables defined in this volume of
788 IEEE Std. 1003.1-200x, except as provided by this volume of IEEE Std. 1003.1-200x.

789 **Default Behavior:** When this section is listed as “Not used.”, it means that the
790 standard input shall not be read when the utility is used as described by this volume of
791 IEEE Std. 1003.1-200x.

792 INPUT FILES

793 The INPUT FILES section describes the files, other than the standard input, used as
794 input by the utility. It includes files named as operands and option-arguments as well
795 as other files that are referred to, such as start-up and initialization files, databases, and

796 so on. Commonly-used files are generally described in one place and cross-referenced
797 by other utilities.

798 XSI All utilities in this volume of IEEE Std. 1003.1-200x shall be capable of processing input
799 files using 8-bit transparency.

800 When a standard utility reads a seekable input file and terminates without an error
801 before it reaches end-of-file, the utility shall ensure that the file offset in the open file
802 description is properly positioned just past the last byte processed by the utility. For
803 files that are not seekable, the state of the file offset in the open file description for that
804 file is unspecified. A portable application shall not assume that the following three
805 commands are equivalent:

```
806     tail -n +2 file
807     (sed -n 1q; cat) < file
808     cat file | (sed -n 1q; cat)
```

809 The second command is equivalent to the first only when the file is seekable. The third
810 command leaves the file offset in the open file description in an unspecified state. Other
811 utilities, such as *head*, *read*, and *sh*, have similar properties.

812 Some of the standard utilities, such as filters, process input files a line or a block at a
813 time and have no restrictions on the maximum input file size. Some utilities may have
814 size limitations that are not as obvious as file space or memory limitations. Such
815 limitations should reflect resource limitations of some sort, not arbitrary limits set by
816 implementors. Implementations shall document those utilities that are limited by
817 constraints other than file system space, available memory, and other limits specifically
818 cited by this volume of IEEE Std. 1003.1-200x, and identify what the constraint is and
819 indicate a way of estimating when the constraint would be reached. Similarly, some
820 utilities descend the directory tree (recursively). Implementations shall also document
821 any limits that they may have in descending the directory tree that are beyond limits
822 cited by this volume of IEEE Std. 1003.1-200x.

823 When an input file is described as a *text file*, the utility produces undefined results if
824 given input that is not from a text file, unless otherwise stated. Some utilities (for
825 example, *make*, *read*, *sh*) allow for continued input lines using an escaped <newline>
826 convention; unless otherwise stated, the utility need not be able to accumulate more
827 than {LINE_MAX} bytes from a set of multiple, continued input lines. Thus, for a
828 portable application the total of all the continued lines in a set cannot exceed
829 {LINE_MAX}. If a utility using the escaped <newline> convention detects an end-of-
830 file condition immediately after an escaped <newline>, the results are unspecified.

831 Record formats are described in a notation similar to that used by the C-language
832 function, *printf()*. See the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5,
833 File Format Notation for a description of this notation. The format description is
834 intended to be sufficiently rigorous to allow other applications to generate these input
835 files. However, since <blank> characters can legitimately be included in some of the
836 fields described by the standard utilities, particularly in locales other than the POSIX
837 locale, this intent is not always realized.

838 **Default Behavior:** When this section is listed as “None.”, it means that no input files
839 are required to be supplied when the utility is used as described by this volume of
840 IEEE Std. 1003.1-200x.

841 ENVIRONMENT VARIABLES

842 The ENVIRONMENT VARIABLES section lists what variables affect the utility’s
843 execution.

844 The entire manner in which environment variables described in this volume of
 845 IEEE Std. 1003.1-200x affect the behavior of each utility is described in the
 846 ENVIRONMENT VARIABLES section for that utility, in conjunction with the global
 847 XSI effects of the *LANG*, *LC_ALL*, and *NLSPATH* environment variables described in the
 848 Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.
 849 The existence or value of environment variables described in this volume of
 850 IEEE Std. 1003.1-200x shall not otherwise affect the specified behavior of the standard
 851 utilities. Any effects of the existence or value of environment variables not described by
 852 this volume of IEEE Std. 1003.1-200x upon the standard utilities are unspecified.

853 For those standard utilities that use environment variables as a means for selecting a
 854 utility to execute (such as *CC* in *make*), the string provided to the utility is subjected to
 855 the path search described for *PATH* in the Base Definitions volume of
 856 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.

857 XSI All utilities in this volume of IEEE Std. 1003.1-200x shall be capable of processing
 858 environment variable names and values using 8-bit transparency.

859 **Default Behavior:** When this section is listed as “None.”, it means that the behavior of
 860 the utility is not directly affected by environment variables described by this volume of
 861 IEEE Std. 1003.1-200x when the utility is used as described by this volume of
 862 IEEE Std. 1003.1-200x.

863 ASYNCHRONOUS EVENTS

864 The ASYNCHRONOUS EVENTS section lists how the utility reacts to such events as
 865 signals and what signals are caught.

866 **Default Behavior:** When this section is listed as “Default.”, or it refers to “the standard
 867 action for all other signals; see Section 1.11 (on page 2224)” it means that the action
 868 taken as a result of the signal shall be one of the following:

- 869 1. The action is that inherited from the parent according to the rules of inheritance
 870 of signal actions defined in the System Interfaces volume of IEEE Std. 1003.1-200x.
- 871 2. When no action has been taken to change the default, the default action is that
 872 specified by the System Interfaces volume of IEEE Std. 1003.1-200x.
- 873 3. The result of the utility’s execution is as if default actions had been taken.

874 A utility is permitted to catch a signal, perform some additional processing (such as
 875 deleting temporary files), restore the default signal action (or action inherited from the
 876 parent process), and resignal itself.

877 STDOUT

878 The STDOUT section describes the standard output of the utility. This section is
 879 frequently merely a reference to the following section, OUTPUT FILES, because many
 880 utilities treat standard output and output files in the same manner.

881 Use of a terminal for standard output may cause any of the standard utilities that write
 882 standard output to stop when used in the background. For this reason, applications
 883 should not use interactive features in scripts to be placed in the background.

884 Record formats are described in a notation similar to that used by the C-language
 885 function, *printf()*. See the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5,
 886 File Format Notation for a description of this notation.

887 The specified standard output of the standard utilities shall not depend on the
 888 existence or value of the environment variables defined in this volume of
 889 IEEE Std. 1003.1-200x, except as provided by this volume of IEEE Std. 1003.1-200x.

890 Some of the standard utilities describe their output using the verb *display*, defined in
891 the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.135, Display. Output
892 described in the STDOUT sections of such utilities may be produced using means other
893 than standard output. When standard output is directed to a terminal, the output
894 described shall be written directly to the terminal. Otherwise, the results are undefined.

895 **Default Behavior:** When this section is listed as “Not used.”, it means that the
896 standard output shall not be written when the utility is used as described by this
897 volume of IEEE Std. 1003.1-200x.

898 **STDERR**

899 The STDERR section describes the standard error output of the utility. Only those
900 messages that are purposely sent by the utility are described.

901 Use of a terminal for standard error may cause any of the standard utilities that write
902 standard error output to stop when used in the background. For this reason,
903 applications should not use interactive features in scripts to be placed in the
904 background.

905 The format of diagnostic messages for most utilities is unspecified, but the language
906 and cultural conventions of diagnostic and informative messages whose format is
907 unspecified by this volume of IEEE Std. 1003.1-200x should be affected by the setting of
908 XSI *LC_MESSAGES* and *NLSPATH*.

909 The specified standard error output of standard utilities shall not depend on the
910 existence or value of the environment variables defined in this volume of
911 IEEE Std. 1003.1-200x, except as provided by this volume of IEEE Std. 1003.1-200x.

912 **Default Behavior:** When this section is listed as “Used only for diagnostic messages.”,
913 it means that, unless otherwise stated, the diagnostic messages shall be sent to the
914 standard error only when the exit status is non-zero and the utility is used as described
915 by this volume of IEEE Std. 1003.1-200x.

916 When this section is listed as “Not used.”, it means that the standard error shall not be
917 used when the utility is used as described in this volume of IEEE Std. 1003.1-200x.

918 **OUTPUT FILES**

919 The OUTPUT FILES section describes the files created or modified by the utility.
920 Temporary or system files that are created for internal usage by this utility or other
921 parts of the implementation (for example, spool, log, and audit files) are not described
922 in this, or any, section. The utilities creating such files and the names of such files are
923 unspecified. If applications are written to use temporary or intermediate files, they
924 should use the *TMPDIR* environment variable, if it is set and represents an accessible
925 directory, to select the location of temporary files.

926 Implementations shall ensure that temporary files, when used by the standard utilities,
927 are named so that different utilities or multiple instances of the same utility can operate
928 simultaneously without regard to their working directories, or any other process
929 characteristic other than process ID. There are two exceptions to this rule:

- 930 1. Resources for temporary files other than the name space (for example, disk space,
931 available directory entries, or number of processes allowed) are not guaranteed.
- 932 2. Certain standard utilities generate output files that are intended as input for other
933 utilities (for example, *lex* generates *lex.yy.c*), and these cannot have unique
934 names. These cases are explicitly identified in the descriptions of the respective
935 utilities.

936 Any temporary file created by the implementation shall be removed by the
937 implementation upon a utility's successful exit, exit because of errors, or before
938 termination by any of the SIGHUP, SIGINT, or SIGTERM signals, unless specified
939 otherwise by the utility description.

940 Receipt of the SIGQUIT signal should generally cause termination (unless in some
941 debugging mode) that would bypass any attempted recovery actions.

942 Record formats are described in a notation similar to that used by the C-language
943 function, *printf()*; see the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5,
944 File Format Notation for a description of this notation.

945 **Default Behavior:** When this section is listed as “None.”, it means that no files are
946 created or modified as a consequence of direct action on the part of the utility when the
947 utility is used as described by this volume of IEEE Std. 1003.1-200x. However, the
948 utility may create or modify system files, such as log files, that are outside the utility's
949 normal execution environment.

950 EXTENDED DESCRIPTION

951 The EXTENDED DESCRIPTION section provides a place for describing the actions of
952 very complicated utilities, such as text editors or language processors, which typically
953 have elaborate command languages.

954 **Default Behavior:** When this section is listed as “None.”, no further description is
955 necessary.

956 EXIT STATUS

957 The EXIT STATUS section describes the values the utility shall return to the calling
958 program, or shell, and the conditions that cause these values to be returned. Usually,
959 utilities return zero for successful completion and values greater than zero for various
960 error conditions. If specific numeric values are listed in this section, the system shall
961 use those values for the errors described. In some cases, status values are listed more
962 loosely, such as >0. A portable application shall not rely on any specific value in the
963 range shown and shall be prepared to receive any value in the range.

964 For example, a utility may list zero as a successful return, 1 as a failure for a specific
965 reason, and >1 as “an error occurred”. In this case, unspecified conditions may cause a
966 2 or 3, or other value, to be returned. A portable application should be written so that it
967 tests for successful exit status values (zero in this case), rather than relying upon the
968 single specific error value listed in this volume of IEEE Std. 1003.1-200x. In that way, it
969 has maximum portability, even on implementations with extensions.

970 Unspecified error conditions may be represented by specific values not listed in this
971 volume of IEEE Std. 1003.1-200x.

972 CONSEQUENCES OF ERRORS

973 The CONSEQUENCES OF ERRORS section describes the effects on the environment,
974 file systems, process state, and so on, when error conditions occur. It does not describe
975 error messages produced or exit status values used.

976 The many reasons for failure of a utility are generally not specified by the utility
977 descriptions. Utilities may terminate prematurely if they encounter: invalid usage of
978 options, arguments, or environment variables; invalid usage of the complex syntaxes
979 expressed in EXTENDED DESCRIPTION sections; difficulties accessing, creating,
980 reading, or writing files; or difficulties associated with the privileges of the process.

981 The following shall apply to each utility, unless otherwise stated:

982 • If the requested action cannot be performed on an operand representing a file,
983 directory, user, process, and so on, the utility shall issue a diagnostic message to
984 standard error and continue processing the next operand in sequence, but the final
985 exit status shall be returned as non-zero.

986 For a utility that recursively traverses a file hierarchy (such as *find* or *chown -R*), if
987 the requested action cannot be performed on a file or directory encountered in the
988 hierarchy, the utility shall issue a diagnostic message to standard error and continue
989 processing the remaining files in the hierarchy, but the final exit status shall be
990 returned as non-zero.

991 • If the requested action characterized by an option or option-argument cannot be
992 performed, the utility shall issue a diagnostic message to standard error and the exit
993 status returned shall be non-zero.

994 • When an unrecoverable error condition is encountered, the utility shall exit with a
995 non-zero exit status.

996 • A diagnostic message shall be written to standard error whenever an error
997 condition occurs.

998 When a utility encounters an error condition several actions are possible, depending on
999 the severity of the error and the state of the utility. Included in the possible actions of
1000 various utilities are: deletion of temporary or intermediate work files; deletion of
1001 incomplete files; validity checking of the file system or directory.

1002 **Default Behavior:** When this section is listed as “Default.”, it means that any changes
1003 to the environment are unspecified.

1004 APPLICATION USAGE

1005 This section is non-normative.

1006 The APPLICATION USAGE section gives advice to the application programmer or user
1007 about the way the utility should be used.

1008 EXAMPLES

1009 This section is non-normative.

1010 The EXAMPLES section gives one or more examples of usage, where appropriate. In
1011 the event of conflict between an example and a normative part of the specification, the
1012 normative material is to be taken as correct.

1013 In all examples, quoting has been used, showing how sample commands (utility names
1014 combined with arguments) could be passed correctly to a shell (see *sh*) or as a string to
1015 the *system()* function defined in the System Interfaces volume of IEEE Std. 1003.1-200x.
1016 Such quoting would not be used if the utility is invoked using one of the *exec* functions
1017 defined in the System Interfaces volume of IEEE Std. 1003.1-200x.

1018 RATIONALE

1019 This section is non-normative.

1020 This section contains historical information concerning the contents of this volume of
1021 IEEE Std. 1003.1-200x and why features were included or discarded by the standard
1022 developers.

1023 FUTURE DIRECTIONS

1024 This section is non-normative.

1025 The FUTURE DIRECTIONS section should be used as a guide to current thinking; there
1026 is not necessarily a commitment to implement all of these future directions in their

1027 entirety.

1028 **SEE ALSO**

1029 This section is non-normative.

1030 The SEE ALSO section lists related entries.

1031 **CHANGE HISTORY**

1032 This section is non-normative.

1033 The CHANGE HISTORY section shows the derivation of the description used by this
1034 volume of IEEE Std. 1003.1-200x and lists the functional differences between Issues 4
1035 and 6.

1036 Certain of the standard utilities describe how they can invoke other utilities or applications, such
1037 as by passing a command string to the command interpreter. The external influences (STDIN,
1038 ENVIRONMENT VARIABLES, and so on) and external effects (STDOUT, CONSEQUENCES OF
1039 ERRORS, and so on) of such invoked utilities are not described in the section concerning the
1040 standard utility that invokes them.

1041 1.12 Considerations for Utilities in Support of Files of Arbitrary Size

1042 The following utilities support files of any size up to the maximum that can be created by the
1043 implementation. This support includes correct writing of file size-related values (such as file
1044 sizes and offsets, line numbers, and block counts) and correct interpretation of command line
1045 arguments that contain such values.

1046	<i>basename</i>	Return non-directory portion of path name.
1047	<i>cat</i>	Concatenate and print files.
1048	<i>cd</i>	Change working directory.
1049	<i>chgrp</i>	Change file group ownership.
1050	<i>chmod</i>	Change file modes.
1051	<i>chown</i>	Change file ownership.
1052	<i>cksum</i>	Write file checksums and sizes.
1053	<i>cmp</i>	Compare two files.
1054	<i>cp</i>	Copy files.
1055	<i>dd</i>	Convert and copy a file.
1056	<i>df</i>	Report free disk space.
1057	<i>dirname</i>	Return directory portion of path name.
1058	<i>du</i>	Estimate file space usage.
1059	<i>find</i>	Find files.
1060	<i>ln</i>	Link files.
1061	<i>ls</i>	List directory contents.
1062	<i>mkdir</i>	Make directories.
1063	<i>mv</i>	Move files.
1064	<i>pathchk</i>	Check path names.
1065	<i>pwd</i>	Return working directory name.
1066	<i>rm</i>	Remove directory entries.
1067	<i>rmdir</i>	Remove directories.
1068	<i>sh</i>	Shell, the standard command language interpreter.
1069	<i>sum</i>	Print checksum and block or byte count of a file.
1070	<i>test</i>	Evaluate expression.
1071	<i>touch</i>	Change file access and modification times.
1072	<i>ulimit</i>	Set or report file size limit.

1073 Exceptions to the requirement that utilities support files of any size up to the maximum are as
1074 follows:

- 1075 1. Uses of files as command scripts, or for configuration or control, are exempt. For example,
1076 it is not required that *sh* be able to read an arbitrarily large **.profile**.

- 1077 2. Shell input and output redirection are exempt. For example, it is not required that the
1078 redirections *sum < file* or *echo foo > file* succeed for an arbitrarily large existing file.

Shell Command Language

1080

1081 This chapter contains the definition of the Shell Command Language.

1082 2.1 Shell Introduction

1083 The shell is a command language interpreter. This chapter describes the syntax of that command
 1084 language as it is used by the *sh* utility and the *system()* and *popen()* functions defined in the
 1085 System Interfaces volume of IEEE Std. 1003.1-200x.

1086 The shell operates according to the following general overview of operations. The specific
 1087 details are included in the cited sections of this chapter.

- 1088 1. The shell reads its input from a file (see *sh*), from the `-c` option or from the *system()* and
 1089 *popen()* functions defined in the System Interfaces volume of IEEE Std. 1003.1-200x. If the
 1090 first line of a file of shell commands starts with the characters "#!", the results are
 1091 XSI unspecified. On XSI-conformant systems, if the first two characters of a file are "#!", it
 1092 shall behave as described for executable scripts in Section 2.10 (on page 2265).
- 1093 2. The shell breaks the input into tokens: words and operators; see Section 2.3 (on page 2238).
- 1094 3. The shell parses the input into simple commands (see Section 2.9.1 (on page 2256)) and
 1095 compound commands (see Section 2.9.4 (on page 2261)).
- 1096 4. The shell performs various expansions (separately) on different parts of each command,
 1097 resulting in a list of path names and fields to be treated as a command and arguments; see
 1098 Section 2.6 (on page 2244).
- 1099 5. The shell performs redirection (see Section 2.7 (on page 2251)) and removes redirection
 1100 operators and their operands from the parameter list.
- 1101 6. The shell executes a function (see Section 2.9.5 (on page 2263)), built-in (see Section 2.15
 1102 (on page 2276)), executable file, or script, giving the names of the arguments as positional
 1103 parameters numbered 1 to *n*, and the name of the command (or in the case of a function
 1104 within a script, the name of the script) as the positional parameter numbered 0 (see Section
 1105 2.9.1.1 (on page 2257)).
- 1106 7. The shell optionally waits for the command to complete and collects the exit status (see
 1107 Section 2.8.2 (on page 2255)).

1108 2.2 Quoting

1109 Quoting is used to remove the special meaning of certain characters or words to the shell.
 1110 Quoting can be used to preserve the literal meaning of the special characters in the next
 1111 paragraph, prevent reserved words from being recognized as such, and prevent parameter
 1112 expansion and command substitution within here-document processing (see Section 2.7.4 (on
 1113 page 2252)).

1114 The application shall quote the following characters if they are to represent themselves:

1115 | & ; < > () \$ ' \ " ' <space> <tab> <newline>

1116 and the following may need to be quoted under certain circumstances. That is, these characters
 1117 may be special depending on conditions described elsewhere in this volume of
 1118 IEEE Std. 1003.1-200x:

1119 * ? [# ~ = %

1120 The various quoting mechanisms are the escape character, single-quotes, and double-quotes.
 1121 The here-document represents another form of quoting; see Section 2.7.4 (on page 2252).

1122 2.2.1 Escape Character (Backslash)

1123 A backslash that is not quoted shall preserve the literal value of the following character, with the
 1124 exception of a <newline> character. If a <newline> character follows the backslash, the shell
 1125 shall interpret this as line continuation. The backslash and <newline> characters shall be
 1126 removed before splitting the input into tokens. Since the escaped <newline> character is
 1127 removed entirely from the input and is not replaced by any white space, it cannot serve as a
 1128 token separator.

1129 2.2.2 Single-Quotes

1130 Enclosing characters in single-quotes (' ') shall preserve the literal value of each character
 1131 within the single-quotes. A single-quote cannot occur within single-quotes.

1132 2.2.3 Double-Quotes

1133 Enclosing characters in double-quotes (" ") shall preserve the literal value of all characters
 1134 within the double-quotes, with the exception of the characters dollar sign, backquote, and
 1135 backslash, as follows:

1136 \$ The dollar sign shall retain its special meaning introducing parameter expansion (see
 1137 Section 2.6.2 (on page 2245)), a form of command substitution (see Section 2.6.3 (on page
 1138 2247)), and arithmetic expansion (see Section 2.6.4 (on page 2248)).

1139 The input characters within the quoted string that are also enclosed between "\$(" and the
 1140 matching ')' is not affected by the double-quotes, but rather shall define that command
 1141 whose output replaces the "\$(...)" when the word is expanded. The tokenizing rules in
 1142 Section 2.3 (on page 2238) shall be applied recursively to find the matching ')'.
 1143

1144 Within the string of characters from an enclosed "\${" to the matching '}', an even number
 1145 of unescaped double-quotes or single-quotes, if any, shall occur. A preceding backslash
 1146 character shall be used to escape a literal '{' or '}'. The rule in Section 2.6.2 (on page
 2245) shall be used to determine the matching '}'.

1147 ` The backquote shall retain its special meaning introducing the other form of command
 1148 substitution (see Section 2.6.3 (on page 2247)). The portion of the quoted string from the
 1149 initial backquote and the characters up to the next backquote that is not preceded by a

1150 backslash, having escape characters removed, defines that command whose output replaces
1151 "`\ . . \`" when the word is expanded. Either of the following cases produces undefined
1152 results:

1153 • A single-quoted or double-quoted string that begins, but does not end, within the
1154 "`\ . . \`" sequence

1155 • A "`\ . . \`" sequence that begins, but does not end, within the same double-quoted
1156 string

1157 \
1158 The backslash shall retain its special meaning as an escape character (see Section 2.2.1 (on
page 2236)) only when followed by one of the following characters when considered special:

1159 \$ \ " \
 <newline>

1160 The application shall ensure that a double-quote is preceded by a backslash to be included
1161 within double-quotes. The parameter '@' has special meaning inside double-quotes and is
1162 described in Section 2.5.2 (on page 2241).

1163 2.3 Token Recognition

1164 The shell reads its input in terms of lines from a file, from a terminal in the case of an interactive
 1165 shell, or from a string in the case of *sh -c* or *system()*. The input lines can be of unlimited length.
 1166 These lines are parsed using two major modes: ordinary token recognition and processing of
 1167 here-documents.

1168 When an **io_here** token has been recognized by the grammar (see Section 2.11 (on page 2266)),
 1169 one or more of the subsequent lines immediately following the next **NEWLINE** token form the
 1170 body of one or more here-documents and shall be parsed according to the rules of Section 2.7.4
 1171 (on page 2252).

1172 When it is not processing an **io_here**, the shell shall break its input into tokens by applying the
 1173 first applicable rule below to the next character in its input. The token shall be from the current
 1174 position in the input until a token is delimited according to one of the rules below; the characters
 1175 forming the token are exactly those in the input, including any quoting characters. If it is
 1176 indicated that a token is delimited, and no characters have been included in a token, processing
 1177 shall continue until an actual token is delimited.

- 1178 1. If the end of input is recognized, the current token shall be delimited. If there is no current
 1179 token, the end-of-input indicator shall be returned as the token.
- 1180 2. If the previous character was used as part of an operator and the current character is not
 1181 quoted and can be used with the current characters to form an operator, it shall be used as
 1182 part of that (operator) token.

1183 On some systems, the symbol "`((`" is a control operator; its use produces unspecified
 1184 results. Applications that wish to have nested subshells, such as:

```
1185     ((echo Hello);(echo World))
```

1186 shall separate the "`((`" characters into two tokens by including white space between them.
 1187 Some systems may treat these as invalid arithmetic expressions instead of subshells.

1188 Certain combinations of characters are invalid in portable scripts, as shown in the
 1189 grammar, and that some systems have assigned these combinations (such as "`|&`") as
 1190 valid control operators. Portable scripts cannot rely on receiving errors in all cases where
 1191 this volume of IEEE Std. 1003.1-200x indicates that a syntax is invalid.

- 1192 3. If the previous character was used as part of an operator and the current character cannot
 1193 be used with the current characters to form an operator, the operator containing the
 1194 previous character shall be delimited.
- 1195 4. If the current character is backslash, single-quote, or double-quote (`'\'`, `'\''`, or `'\"'`)
 1196 and it is not quoted, it shall affect quoting for subsequent characters up to the end of the
 1197 quoted text. The rules for quoting are as described in Section 2.2 (on page 2236). During
 1198 token recognition no substitutions shall be actually performed, and the result token shall
 1199 contain exactly the characters that appear in the input (except for `<newline>` character
 1200 joining), unmodified, including any embedded or enclosing quotes or substitution
 1201 operators, between the quote mark and the end of the quoted text. The token shall not be
 1202 delimited by the end of the quoted field.
- 1203 5. If the current character is an unquoted `'$'` or `'\''`, the shell shall identify the start of any
 1204 candidates for parameter expansion (Section 2.6.2 (on page 2245)), command substitution
 1205 (Section 2.6.3 (on page 2247)), or arithmetic expansion (Section 2.6.4 (on page 2248)) from
 1206 their introductory unquoted character sequences: `'$'` or `"${"`, `"$(` or `'\''`, and `"$((`,
 1207 respectively. The shell shall read sufficient input to determine the end of the unit to be
 1208 expanded (as explained in the cited sections). While processing the characters, if instances

- 1209 of expansions or quoting are found nested within the substitution, the shell shall
 1210 recursively process them in the manner specified for the construct that is found. The
 1211 characters found from the beginning of the substitution to its end, allowing for any
 1212 recursion necessary to recognize embedded constructs, shall be included unmodified in the
 1213 result token, including any embedded or enclosing substitution operators or quotes. The
 1214 token shall not be delimited by the end of the substitution.
- 1215 6. If the current character is not quoted and can be used as the first character of a new
 1216 operator, the current token (if any) shall be delimited. The current character shall be used
 1217 as the beginning of the next (operator) token.
 - 1218 7. If the current character is an unquoted <newline> character, the current token shall be
 1219 delimited.
 - 1220 8. If the current character is an unquoted <blank> character, any token containing the
 1221 previous character is delimited and the current character shall be discarded.
 - 1222 9. If the previous character was part of a word, the current character shall be appended to
 1223 that word.
 - 1224 10. If the current character is a '#', it and all subsequent characters up to, but excluding, the
 1225 next <newline> character shall be discarded as a comment. The <newline> character that
 1226 ends the line is not considered part of the comment.
 - 1227 11. The current character is used as the start of a new word.
- 1228 Once a token is delimited, it is categorized as required by the grammar in Section 2.11 (on page
 1229 2266).

1230 2.3.1 Alias Substitution

1231 UP XSI The processing of aliases shall be supported on all XSI-conformant systems or if the system
 1232 supports the User Portability Utilities option (and the rest of this section is not further shaded for
 1233 these options).

1234 After a token has been delimited, but before applying the grammatical rules in Section 2.11 (on
 1235 page 2266), a resulting word that is identified to be the command name word of a simple
 1236 command shall be examined to determine whether it is an unquoted, valid alias name. However,
 1237 reserved words in correct grammatical context shall not be candidates for alias substitution. A
 1238 valid alias name (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.10, Alias
 1239 Name) shall be one that has been defined by the *alias* utility and not subsequently undefined
 1240 using *unalias*. Implementations also may provide predefined valid aliases that are in effect when
 1241 the shell is invoked. To prevent infinite loops in recursive aliasing, if the shell is not currently
 1242 processing an alias of the same name, the word shall be replaced by the value of the alias;
 1243 otherwise, it shall not be replaced.

1244 If the value of the alias replacing the word ends in a <blank> character, the shell shall check the
 1245 next command word for alias substitution; this process shall continue until a word is found that
 1246 is not a valid alias or an alias value does not end in a <blank> character.

1247 When used as specified by this volume of IEEE Std. 1003.1-200x, alias definitions shall not be
 1248 inherited by separate invocations of the shell or by the utility execution environments invoked
 1249 by the shell; see Section 2.13 (on page 2273).

1250 **2.4 Reserved Words**

1251 Reserved words are words that have special meaning to the shell; see Section 2.9 (on page 2256).
1252 The following words shall be recognized as reserved words:

1253	!	do	esac	in
1254	{	done	fi	then
1255	}	elif	for	until
1256	case	else	if	while

1257 This recognition shall only occur when none of the characters is quoted and when the word is
1258 used as:

- 1259 • The first word of a command
- 1260 • The first word following one of the reserved words other than **case**, **for**, or **in**
- 1261 • The third word in a **case** or **for** command (only **in** is valid in this case)

1262 See the grammar in Section 2.11 (on page 2266).

1263 The following words may be recognized as reserved words on some systems (when none of the
1264 characters are quoted), causing unspecified results:

1265	[[]]	function	select
------	-----------	-----------	-----------------	---------------

1266 Words that are the concatenation of a name and a colon (':') are reserved; their use produces
1267 unspecified results. This reservation is to allow future implementations that support named
1268 labels for flow control.

1269 **2.5 Parameters and Variables**

1270 A parameter can be denoted by a name, a number, or one of the special characters listed in
1271 Section 2.5.2. A variable is a parameter denoted by a name.

1272 A parameter is set if it has an assigned value (null is a valid value). Once a variable is set, it can
1273 only be unset by using the *unset* special built-in command.

1274 **2.5.1 Positional Parameters**

1275 A positional parameter is a parameter denoted by the decimal value represented by one or more
1276 digits, other than the single digit 0. The digits denoting the positional parameters shall always be
1277 interpreted as a decimal value, even if there is a leading zero. When a positional parameter with
1278 more than one digit is specified, the application shall enclose the digits in braces (see Section
1279 2.6.2 (on page 2245)). Positional parameters are initially assigned when the shell is invoked (see
1280 *sh*), temporarily replaced when a shell function is invoked (see Section 2.9.5 (on page 2263)), and
1281 can be reassigned with the *set* special built-in command.

1282 **2.5.2 Special Parameters**

1283 Listed below are the special parameters and the values to which they shall expand. Only the
1284 values of the special parameters are listed; see Section 2.6 (on page 2244) for a detailed summary
1285 of all the stages involved in expanding words.

1286 @ Expands to the positional parameters, starting from one. When the expansion occurs within
1287 double-quotes, and where field splitting (see Section 2.6.5 (on page 2249)) is performed,
1288 each positional parameter expands as a separate field, with the provision that the expansion
1289 of the first parameter is still joined with the beginning part of the original word (assuming
1290 that the expanded parameter was embedded within a word), and the expansion of the last
1291 parameter is still joined with the last part of the original word. If there are no positional
1292 parameters, the expansion of '@' generates zero fields, even when '@' is double-quoted.

1293 * Expands to the positional parameters, starting from one. When the expansion occurs within
1294 a double-quoted string (see Section 2.2.3 (on page 2236)), it expands to a single field with the
1295 value of each parameter separated by the first character of the *IFS* variable, or by a <space>
1296 character if *IFS* is unset. If *IFS* is set to a null string, this is not equivalent to unsetting it; its
1297 first character does not exist, so the parameter values are concatenated.

1298 # Expands to the decimal number of positional parameters. The command name (parameter
1299 0) is not counted in the number given by '#' because it is a special parameter, not a
1300 positional parameter.

1301 ? Expands to the decimal exit status of the most recent pipeline (see Section 2.9.2 (on page
1302 2258)).

1303 – (Hyphen.) Expands to the current option flags (the single-letter option names concatenated
1304 into a string) as specified on invocation by the *set* special built-in command or implicitly by
1305 the shell.

1306 \$ Expands to the decimal process ID of the invoked shell. In a subshell (see Section 2.13 (on
1307 page 2273)), '\$' shall expand to the same value as that of the current shell.

1308 ! Expands to the decimal process ID of the most recent background command (see Section
1309 2.9.3 (on page 2259)) executed from the current shell. (For example, background commands
1310 executed from subshells do not affect the value of "\$!" in the current shell environment.)
1311 For a pipeline, the process ID is that of the last command in the pipeline.

1312 0 (Zero.) Expands to the name of the shell or shell script. See *sh* (on page 3060) for a detailed
1313 description of how this name is derived.

1314 See the description of the *IFS* variable in Section 2.5.3.

1315 2.5.3 Shell Variables

1316 Variables shall be initialized from the environment (as defined by the Base Definitions volume of
1317 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables and the *exec* function in the System
1318 Interfaces volume of IEEE Std. 1003.1-200x) and can be given new values with variable
1319 assignment commands. If a variable is initialized from the environment, it shall be marked for
1320 export immediately; see the *export* special built-in. New variables can be defined and initialized
1321 with variable assignments, with the *read* or *getopts* utilities, with the *name* parameter in a *for*
1322 loop, with the $\${name=word}$ expansion, or with other mechanisms provided as implementation
1323 extensions.

1324 The following variables shall affect the execution of the shell.

1325 *ENV* This variable, when and only when an interactive shell is invoked, shall be
1326 subjected to parameter expansion (see Section 2.6.2 (on page 2245)) by the
1327 shell and the resulting value shall be used as a path name of a file containing
1328 shell commands to execute in the current environment. The file need not be
1329 executable. If the expanded value of *ENV* is not an absolute path name, the
1330 results are unspecified. *ENV* shall be ignored if the user's real and effective
1331 user IDs or real and effective group IDs are different.

1332 UP XSI The processing of the *ENV* shell variable shall be supported on all XSI-
1333 conformant systems or if the system supports the User Portability Utilities
1334 option.

1335 *HOME* This variable shall be interpreted as the path name of the user's home
1336 directory. The contents of *HOME* are used in tilde expansion (see Section 2.6.1
1337 (on page 2244)).

1338 *IFS* (Input Field Separators.) A string treated as a list of characters that is used for
1339 field splitting and to split lines into fields with the *read* command. If *IFS* is not
1340 set, the shell shall behave as if the value of *IFS* were the <space>, <tab>, and
1341 <newline> characters; see Section 2.6.5 (on page 2249).

1342 *LANG* This variable shall provide a default value for the internationalization
1343 variables that are unset or null. If *LANG* is unset or null, the corresponding
1344 value from the implementation-defined default locale is used. If any of the
1345 internationalization variables contains an invalid setting, the utility behaves as
1346 if none of the variables had been defined.

1347 *LC_ALL* This variable shall provide a default value for the *LC_** variables, as described
1348 in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8,
1349 Environment Variables.

1350 *LC_COLLATE* This variable shall determine the behavior of range expressions, equivalence
1351 classes, and multi-character collating elements within pattern matching.

1352 *LC_CTYPE* This variable shall determine the interpretation of sequences of bytes of text
1353 data as characters (for example, single-byte as opposed to multi-byte
1354 characters), which characters are defined as letters (character class **alpha**) and
1355 <blank> characters (character class **blank**), and the behavior of character
1356 classes within pattern matching. Changing the value of *LC_CTYPE* after the
1357 shell has started shall not affect the lexical processing of shell commands in

1358		the current shell execution environment or its subshells. Invoking a shell script or performing <i>exec sh</i> subjects the new shell to the changes in <i>LC_CTYPE</i> .
1359		
1360		
1361	<i>LC_MESSAGES</i>	This variable shall determine the language in which messages should be written.
1362		
1363	<i>LINENO</i>	This variable shall be set by the shell to a decimal number representing the current sequential line number (numbered starting with 1) within a script or function before it executes each command. If the user unsets or resets <i>LINENO</i> , the variable may lose its special meaning for the life of the shell. If the shell is not currently executing a script or function, the value of <i>LINENO</i> is unspecified. This volume of IEEE Std. 1003.1-200x specifies the effects of the variable only for systems supporting the User Portability Utilities option.
1364		
1365		
1366		
1367		
1368		
1369		
1370	XSI <i>NLSPATH</i>	This variable shall determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
1371		
1372	<i>PATH</i>	This variable represents a string formatted as described in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables, used to effect command interpretation; see Section 2.9.1.1 (on page 2257).
1373		
1374		
1375	<i>PPID</i>	This variable shall be set by the shell to the decimal process ID of the process that invoked this shell. In a subshell (see Section 2.13 (on page 2273)), <i>PPID</i> shall be set to the same value as that of the parent of the current shell. For example, <i>echo\$PPID</i> and (<i>echo\$PPID</i>) would produce the same value. This volume of IEEE Std. 1003.1-200x specifies the effects of the variable only for systems supporting the User Portability Utilities option.
1376		
1377		
1378		
1379		
1380		
1381	<i>PS1</i>	Each time an interactive shell is ready to read a command, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value shall be "\$ ". For users who have specific additional implementation-defined privileges, the default may be another, implementation-defined value. (Historically, the superuser has had a prompt of '#'.) The shell shall replace each instance of the character '!' in <i>PS1</i> with the history file number of the next command to be typed. Escaping the '!' with another '!' (that is, "!!") shall place the literal character '!' in the prompt. This volume of IEEE Std. 1003.1-200x specifies the effects of the variable only for systems supporting the User Portability Utilities option.
1382		
1383		
1384		
1385		
1386		
1387		
1388		
1389		
1390		
1391	<i>PS2</i>	Each time the user enters a <newline> character prior to completing a command line in an interactive shell, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value is "> ". This volume of IEEE Std. 1003.1-200x specifies the effects of the variable only for systems supporting the User Portability Utilities option.
1392		
1393		
1394		
1395		
1396	<i>PS4</i>	When an execution trace (<i>set -x</i>) is being performed in an interactive shell, before each line in the execution trace, the value of this variable shall be subjected to parameter expansion and written to standard error. The default value is "+ ". This volume of IEEE Std. 1003.1-200x specifies the effects of the variable only for systems supporting the User Portability Utilities option.
1397		
1398		
1399		
1400		
1401	<i>PWD</i>	This variable shall be set by the shell to be an absolute path name of the current working directory, containing no components of type symbolic link, no components that are dot, and no components that are dot-dot when the shell is initialized. If an application sets or unsets the value of <i>PWD</i> , the behaviors of the <i>cd</i> and <i>pwd</i> utilities are unspecified.
1402		
1403		
1404		
1405		

1406 2.6 Word Expansions

1407 This section describes the various expansions that are performed on words. Not all expansions
1408 are performed on every word, as explained in the following sections.

1409 Tilde expansions, parameter expansions, command substitutions, arithmetic expansions, and
1410 quote removals that occur within a single word expand to a single field. It is only field splitting
1411 or path name expansion that can create multiple fields from a single word. The single exception
1412 to this rule is the expansion of the special parameter '@' within double-quotes, as described in
1413 Section 2.5.2 (on page 2241).

1414 The order of word expansion shall be as follows:

- 1415 1. Tilde expansion (see Section 2.6.1), parameter expansion (see Section 2.6.2 (on page 2245)),
1416 command substitution (see Section 2.6.3 (on page 2247)), and arithmetic expansion (see
1417 Section 2.6.4 (on page 2248)) shall be performed, beginning to end. See item 5 in Section 2.3
1418 (on page 2238).
- 1419 2. Field splitting (see Section 2.6.5 (on page 2249)) shall be performed on the portions of the
1420 fields generated by step 1, unless *IFS* is null.
- 1421 3. Path name expansion (see Section 2.6.6 (on page 2249)) shall be performed, unless *set -f* is
1422 in effect.
- 1423 4. Quote removal (see Section 2.6.7 (on page 2250)) shall always be performed last.

1424 The expansions described in this section shall occur in the same shell environment as that in
1425 which the command is executed.

1426 If the complete expansion appropriate for a word results in an empty field, that empty field shall
1427 be deleted from the list of fields that form the completely expanded command, unless the
1428 original word contained single-quote or double-quote characters.

1429 The '\$' character is used to introduce parameter expansion, command substitution, or
1430 arithmetic evaluation. If an unquoted '\$' is followed by a character that is either not numeric,
1431 the name of one of the special parameters (see Section 2.5.2 (on page 2241)), a valid first
1432 character of a variable name, a left curly brace ('{') or a left parenthesis, the result is
1433 unspecified.

1434 2.6.1 Tilde Expansion

1435 A *tilde-prefix* consists of an unquoted tilde character at the beginning of a word, followed by all
1436 of the characters preceding the first unquoted slash in the word, or all the characters in the word
1437 if there is no slash. In an assignment (see the Base Definitions volume of IEEE Std. 1003.1-200x,
1438 Section 4.16, Variable Assignment), multiple tilde-prefixes can be used: at the beginning of the
1439 word (that is, following the equal sign of the assignment), following any unquoted colon, or
1440 both. A tilde-prefix in an assignment is terminated by the first unquoted colon or slash. If none
1441 of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the
1442 tilde are treated as a possible login name from the user database. A portable login name cannot
1443 contain characters outside the set given in the description of the *LOGNAME* environment
1444 variable in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 8.3, Other Environment
1445 Variables. If the login name is null (that is, the tilde-prefix contains only the tilde), the tilde-
1446 prefix is replaced by the value of the variable *HOME*. If *HOME* is unset, the results are
1447 unspecified. Otherwise, the tilde-prefix is replaced by a path name of the initial working
1448 directory associated with the login name obtained using the *getpwnam()* function as defined in
1449 the System Interfaces volume of IEEE Std. 1003.1-200x. If the system does not recognize the login
1450 name, the results are undefined.

1451 **2.6.2 Parameter Expansion**

1452 The format for parameter expansion is as follows:

1453 $\${expression}$

1454 where *expression* consists of all characters until the matching '}'. Any '}' escaped by a
 1455 backslash or within a quoted string, and characters in embedded arithmetic expansions,
 1456 command substitutions, and variable expansions, shall not be examined in determining the
 1457 matching '}'.

1458 The simplest form for parameter expansion is:

1459 $\${parameter}$ 1460 The value, if any, of *parameter* shall be substituted.

1461 The parameter name or symbol can be enclosed in braces, which are optional except for
 1462 positional parameters with more than one digit or when *parameter* is followed by a character that
 1463 could be interpreted as part of the name. The matching closing brace shall be determined by
 1464 counting brace levels, skipping over enclosed quoted strings, and command substitutions.

1465 If the parameter name or symbol is not enclosed in braces, the expansion shall use the longest
 1466 valid name (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.232, Name),
 1467 whether or not the symbol represented by that name exists.

1468 If a parameter expansion occurs inside double-quotes:

- 1469 • Path name expansion shall not be performed on the results of the expansion.
- 1470 • Field splitting shall not be performed on the results of the expansion, with the exception of
 1471 '@'; see Section 2.5.2 (on page 2241).

1472 In addition, a parameter expansion can be modified by using one of the following formats. In
 1473 each case that a value of *word* is needed (based on the state of *parameter*, as described below),
 1474 *word* shall be subjected to tilde expansion, parameter expansion, command substitution, and
 1475 arithmetic expansion. If *word* is not needed, it shall not be expanded. The '}' character that
 1476 delimits the following parameter expansion modifications shall be determined as described
 1477 previously in this section and in Section 2.2.3 (on page 2236). (For example, $\{\mathbf{foo-bar}\mathbf{xyz}$
 1478 would result in the expansion of **foo** followed by the string **xyz** if **foo** is set, else the string
 1479 "barxyz").

1480 $\${parameter}:-word$ **Use Default Values.** If *parameter* is unset or null, the expansion of *word*
 1481 shall be substituted; otherwise, the value of *parameter* shall be substituted.

1482 $\${parameter}:=word$ **Assign Default Values.** If *parameter* is unset or null, the expansion of
 1483 *word* shall be assigned to *parameter*. In all cases, the final value of
 1484 *parameter* shall be substituted. Only variables, not positional parameters
 1485 or special parameters, can be assigned in this way.

1486 $\${parameter}?:[word]$ **Indicate Error if Null or Unset.** If *parameter* is unset or null, the
 1487 expansion of *word* (or a message indicating it is unset if *word* is omitted)
 1488 shall be written to standard error and the shell exits with a non-zero exit
 1489 status. Otherwise, the value of *parameter* shall be substituted. An
 1490 interactive shell need not exit.

1491 $\${parameter}:+word$ **Use Alternative Value.** If *parameter* is unset or null, null shall be
 1492 substituted; otherwise, the expansion of *word* shall be substituted.

1493 In the parameter expansions shown previously, use of the colon in the format results in a test for
 1494 a parameter that is unset or null; omission of the colon results in a test for a parameter that is

1495 only unset. The following table summarizes the effect of the colon:

	<i>parameter</i> Set and Not Null	<i>parameter</i> Set But Null	<i>parameter</i> Unset
1496 <i>S{parameter:-word}</i>	substitute <i>parameter</i>	substitute <i>word</i>	substitute <i>word</i>
1497 <i>S{parameter-word}</i>	substitute <i>parameter</i>	substitute null	substitute <i>word</i>
1498 <i>S{parameter:=word}</i>	substitute <i>parameter</i>	assign <i>word</i>	assign <i>word</i>
1499 <i>S{parameter=word}</i>	substitute <i>parameter</i>	substitute <i>parameter</i>	assign null
1500 <i>S{parameter?word}</i>	substitute <i>parameter</i>	error, exit	error, exit
1501 <i>S{parameter?word}</i>	substitute <i>parameter</i>	substitute null	error, exit
1502 <i>S{parameter+word}</i>	substitute <i>word</i>	substitute null	substitute null
1503 <i>S{parameter+word}</i>	substitute <i>word</i>	substitute <i>word</i>	substitute null

1506 In all cases shown with “substitute”, the expression is replaced with the value shown. In all
1507 cases shown with “assign”, *parameter* is assigned that value, which also replaces the expression.

1508 ***S{#parameter}*** **String Length.** The length in characters of the value of *parameter* shall be
1509 substituted. If *parameter* is '*' or '@', the result of the expansion is
1510 unspecified.

1511 The following four varieties of parameter expansion provide for substring processing. In each
1512 case, pattern matching notation (see Section 2.14 (on page 2274)), rather than regular expression
1513 notation, shall be used to evaluate the patterns. If *parameter* is '*' or '@', the result of the
1514 expansion is unspecified. Enclosing the full parameter expansion string in double-quotes shall
1515 not cause the following four varieties of pattern characters to be quoted, whereas quoting
1516 characters within the braces shall have this effect.

1517 ***S{parameter%word}*** **Remove Smallest Suffix Pattern.** The *word* is expanded to produce a
1518 pattern. The parameter expansion then results in *parameter*, with the
1519 smallest portion of the suffix matched by the *pattern* deleted.

1520 ***S{parameter%%word}*** **Remove Largest Suffix Pattern.** The *word* shall be expanded to produce a
1521 pattern. The parameter expansion then results in *parameter*, with the
1522 largest portion of the suffix matched by the *pattern* deleted.

1523 ***S{parameter#word}*** **Remove Smallest Prefix Pattern.** The *word* shall be expanded to produce
1524 a pattern. The parameter expansion then results in *parameter*, with the
1525 smallest portion of the prefix matched by the *pattern* deleted.

1526 ***S{parameter##word}*** **Remove Largest Prefix Pattern.** The *word* shall be expanded to produce a
1527 pattern. The parameter expansion then results in *parameter*, with the
1528 largest portion of the prefix matched by the *pattern* deleted.

1529 Examples

1530 ***S{parameter:-word}***
1531 In this example, *ls* is executed only if *x* is null or unset. (The *S{ls}* command substitution
1532 notation is explained in Section 2.6.3 (on page 2247).)

```
1533     ${x:-$(ls)}
```

```
1534 S{parameter:=word}  
1535     unset X  
1536     echo ${X:=abc}  
1537     abc
```

```
1538 S{parameter:?word}  
1539     unset posix
```

```
1540     echo ${posix:?}
1541     sh: posix: parameter null or not set
```

```
1542     ${parameter:+word}
1543     set a b c
1544     echo ${3:+posix}
1545     posix
```

```
1546     ${#parameter}
1547     HOME=/usr/posix
1548     echo ${#HOME}
1549     10
```

```
1550     ${parameter%word}
1551     x=file.c
1552     echo ${x%.c}.o
1553     file.o
```

```
1554     ${parameter%%word}
1555     x=posix/src/std
1556     echo ${x%%/*}
1557     posix
```

```
1558     ${parameter#word}
1559     x=$HOME/src/cmd
1560     echo ${x#$HOME}
1561     /src/cmd
```

```
1562     ${parameter##word}
1563     x=/one/two/three
1564     echo ${x##*/}
1565     three
```

1566 The double-quoting of patterns is different depending on where the double-quotes are placed:

1567 "\$ {x#*} " The asterisk is a pattern character. |

1568 \${x#"*" } The literal asterisk is quoted and not special. |

1569 2.6.3 Command Substitution

1570 Command substitution allows the output of a command to be substituted in place of the
1571 command name itself. Command substitution shall occur when the command is enclosed as
1572 follows:

```
1573     $( command )
```

1574 or (backquoted version):

```
1575     ` command `
```

1576 The shell shall expand the command substitution by executing *command* in a subshell
1577 environment (see Section 2.13 (on page 2273)) and replacing the command substitution (the text
1578 of *command* plus the enclosing "\$ () " or backquotes) with the standard output of the command,
1579 removing sequences of one or more <newline> characters at the end of the substitution.
1580 Embedded <newline> characters before the end of the output shall not be removed; however,
1581 they may be treated as field delimiters and eliminated during field splitting, depending on the
1582 value of *IFS* and quoting that is in effect.

1583 Within the backquoted style of command substitution, backslash shall retain its literal meaning,
 1584 except when followed by: '\$', '`', or '\\` (dollar sign, backquote, backslash). The search for
 1585 the matching backquote shall be satisfied by the first backquote found without a preceding
 1586 backslash; during this search, if a non-escaped backquote is encountered within a shell
 1587 comment, a here-document, an embedded command substitution of the \$(*command*) form, or a
 1588 quoted string, undefined results occur. A single-quoted or double-quoted string that begins, but
 1589 does not end, within the "`...`" sequence produces undefined results.

1590 With the \$(*command*) form, all characters following the open parenthesis to the matching closing
 1591 parenthesis constitute the *command*. Any valid shell script can be used for *command*, except:

- 1592 • A script consisting solely of redirections produces unspecified results
- 1593 • See the restriction on single subshells described below

1594 The results of command substitution shall not be processed for further tilde expansion,
 1595 parameter expansion, command substitution, or arithmetic expansion. If a command
 1596 substitution occurs inside double-quotes, it shall not be performed on the results of the
 1597 substitution.

1598 Command substitution can be nested. To specify nesting within the backquoted version, the
 1599 application shall precede the inner backquotes with backslashes, for example:

```
1600 \ `command` \ `
```

1601 If the command substitution consists of a single subshell, such as:

```
1602 $( (command) )
```

1603 a portable application shall separate the "\$(" and "((" into two tokens (that is, separate them
 1604 with white space). This is required to avoid any ambiguities with arithmetic expansion.

1605 2.6.4 Arithmetic Expansion

1606 Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and
 1607 substituting its value. The format for arithmetic expansion shall be as follows:

```
1608 $((expression))
```

1609 The expression shall be treated as if it were in double-quotes, except that a double-quote inside
 1610 the expression is not treated specially. The shell expands all tokens in the expression for
 1611 parameter expansion, command substitution, and quote removal.

1612 Next, the shell shall treat this as an arithmetic expression and substitutes the value of the
 1613 expression. The arithmetic expression shall be processed according to the rules of the ISO C
 1614 standard, with the following exceptions:

- 1615 • Only integer arithmetic is required.
- 1616 • The *sizeof*() operator and the prefix and postfix "++" and "--" operators are not required.
- 1617 • Selection, iteration, and jump statements are not supported.

1618 As an extension, the shell may recognize arithmetic expressions beyond those listed. If the
 1619 expression is invalid, the expansion fails and the shell shall write a message to standard error
 1620 indicating the failure.

1621 **Examples**

1622 A simple example using arithmetic expansion:

```

1623     # repeat a command 100 times
1624     x=100
1625     while [ $x -gt 0 ]
1626     do
1627         command
1628         x=$((x-1))
1629     done

```

1630 **2.6.5 Field Splitting**

1631 After parameter expansion (Section 2.6.2 (on page 2245)), command substitution (Section 2.6.3
 1632 (on page 2247)), and arithmetic expansion (Section 2.6.4 (on page 2248)), the shell shall scan the
 1633 results of expansions and substitutions that did not occur in double-quotes for field splitting and
 1634 multiple fields can result.

1635 The shell shall treat each character of the *IFS* as a delimiter and uses the delimiters to split the
 1636 results of parameter expansion and command substitution into fields.

1637 1. If the value of *IFS* is a <space>, <tab>, and <newline> character, or if it is unset, any
 1638 sequence of <space>, <tab>, or <newline> characters at the beginning or end of the input
 1639 shall be ignored and any sequence of those characters within the input shall delimit a field.
 1640 For example, the input:

```
1641     <newline><space><tab>foo<tab><tab>bar<space>
```

1642 yields two fields, **foo** and **bar**.

1643 2. If the value of *IFS* is null, no field splitting shall be performed.

1644 3. Otherwise, the following rules shall be applied in sequence. The term “*IFS* white space” is
 1645 used to mean any sequence (zero or more instances) of white space characters that are in
 1646 the *IFS* value (for example, if *IFS* contains <space>/<comma>/<tab>, any sequence of
 1647 <space> and <tab> characters is considered *IFS* white space).

1648 a. *IFS* white space shall be ignored at the beginning and end of the input.

1649 b. Each occurrence in the input of an *IFS* character that is not *IFS* white space, along
 1650 with any adjacent *IFS* white space, shall delimit a field, as described previously.

1651 c. Non-zero-length *IFS* white space shall delimit a field.

1652 **2.6.6 Path Name Expansion**

1653 After field splitting, if *set -f* is not in effect, each field in the resulting command line shall be
 1654 expanded using the algorithm described in Section 2.14 (on page 2274), qualified by the rules in
 1655 Section 2.14.3 (on page 2275).

1656 **2.6.7 Quote Removal**

1657 The quote characters: '\', '\'', and '"' (backslash, single-quote, double-quote) that were |
1658 present in the original word shall be removed unless they have themselves been quoted.

1659 **2.7 Redirection**

1660 Redirection is used to open and close files for the current shell execution environment (see
1661 Section 2.13 (on page 2273)) or for any command. *Redirection operators* can be used with numbers
1662 representing file descriptors (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section
1663 3.167, File Descriptor) as described below.

1664 The overall format used for redirection is:

```
1665     [n]redir-op word
```

1666 The number *n* is an optional decimal number designating the file descriptor number; the
1667 application shall ensure it is delimited from any preceding text and immediately precede the
1668 redirection operator *redir-op*. If *n* is quoted, the number shall not be recognized as part of the
1669 redirection expression. For example:

```
1670     echo \2>a
```

1671 writes the character 2 into file a. If any part of *redir-op* is quoted, no redirection expression is
1672 recognized. For example:

```
1673     echo 2\>a
```

1674 writes the characters 2>a to standard output. The optional number, redirection operator, and
1675 *word* shall not appear in the arguments provided to the command to be executed (if any).

1676 Open files are represented by decimal numbers starting with zero. The largest possible value is
1677 implementation-defined; however, all implementations shall support at least 0 to 9, inclusive, for
1678 use by the application. These numbers are called *file descriptors*. The values 0, 1, and 2 have
1679 special meaning and conventional uses and are implied by certain redirection operations; they
1680 are referred to as *standard input*, *standard output*, and *standard error*, respectively. Programs
1681 usually take their input from standard input, and write output on standard output. Error
1682 messages are usually written on standard error. The redirection operators can be preceded by
1683 one or more digits (with no intervening <blank> characters allowed) to designate the file
1684 descriptor number.

1685 If the redirection operator is "<<" or "<<-", the word that follows the redirection operator shall
1686 be subjected to quote removal; it is unspecified whether any of the other expansions occur. For
1687 the other redirection operators, the word that follows the redirection operator shall be subjected
1688 to tilde expansion, parameter expansion, command substitution, arithmetic expansion, and
1689 quote removal. Path name expansion shall not be performed on the word by a non-interactive
1690 shell; an interactive shell may perform it, but does do so only when the expansion would result
1691 in one word.

1692 If more than one redirection operator is specified with a command, the order of evaluation is
1693 from beginning to end.

1694 A failure to open or create a file shall cause a redirection to fail.

1695 2.7.1 Redirecting Input

1696 Input redirection shall cause the file whose name results from the expansion of *word* to be
 1697 opened for reading on the designated file descriptor, or standard input if the file descriptor is not
 1698 specified.

1699 The general format for redirecting input is:

1700 `[n]<word`

1701 where the optional *n* represents the file descriptor number. If the number is omitted, the
 1702 redirection shall refer to standard input (file descriptor 0).

1703 2.7.2 Redirecting Output

1704 The two general formats for redirecting output are:

1705 `[n]>word`

1706 `[n]>|word`

1707 where the optional *n* represents the file descriptor number. If the number is omitted, the
 1708 redirection shall refer to standard output (file descriptor 1).

1709 Output redirection using the '*>*' format shall fail if the *noclobber* option is set (see the
 1710 description of *set -C*) and the file named by the expansion of *word* exists and is a regular file.
 1711 Otherwise, redirection using the '*>*' or "*>|*" formats shall cause the file whose name results
 1712 from the expansion of *word* to be created and opened for output on the designated file
 1713 descriptor, or standard output if none is specified. If the file does not exist, it shall be created;
 1714 otherwise, it shall be truncated to be an empty file after being opened.

1715 2.7.3 Appending Redirected Output

1716 Appended output redirection shall cause the file whose name results from the expansion of
 1717 *word* to be opened for output on the designated file descriptor. The file is opened as if the *open()*
 1718 function as defined in the System Interfaces volume of IEEE Std. 1003.1-200x was called with the
 1719 *O_APPEND* flag. If the file does not exist, it shall be created.

1720 The general format for appending redirected output is as follows:

1721 `[n]>>word`

1722 where the optional *n* represents the file descriptor number. If the number is omitted, the
 1723 redirection refers to standard output (file descriptor 1).

1724 2.7.4 Here-Document

1725 The redirection operators "*<<*" and "*<<-*" both allow redirection of lines contained in a shell
 1726 input file, known as a *here-document*, to the input of a command.

1727 The here-document shall be treated as a single word that begins after the next *<newline>*
 1728 character and continues until there is a line containing only the delimiter, with no trailing
 1729 *<blank>* characters. Then the next here-document starts, if there is one. The format is as follows:

1730 `[n]<<word`

1731 `here-document`

1732 `delimiter`

1733 where the optional *n* represents the file descriptor number. If the number is omitted, the here-
 1734 document refers to standard output (file descriptor 0).

1735 If any character in *word* is quoted, the delimiter shall be formed by performing quote removal on
 1736 *word*, and the here-document lines are not expanded. Otherwise, the delimiter shall be the *word*
 1737 itself.

1738 If no characters in *word* are quoted, all lines of the here-document shall be expanded for
 1739 parameter expansion, command substitution, and arithmetic expansion. In this case, the
 1740 backslash in the input behaves as the backslash inside double-quotes (see Section 2.2.3 (on page
 1741 2236)). However, the double-quote character (' " ') shall not be treated specially within a here-
 1742 document, except when the double-quote appears within "\$ () ", "` ` ", or "\$ { } ".

1743 If the redirection symbol is "<<-", all leading tab characters shall be stripped from input lines
 1744 and the line containing the trailing delimiter. If more than one "<<" or "<<-" operator is
 1745 specified on a line, the here-document associated with the first operator shall be supplied first by
 1746 the application and shall be read first by the shell.

1747 **Examples**

1748 An example of a here-document follows:

```
1749     cat <<eof1; cat <<eof2
1750     Hi ,
1751     eof1
1752     Helene.
1753     eof2
```

1754 **2.7.5 Duplicating an Input File Descriptor**

1755 The redirection operator:

```
1756     [n]<&word
```

1757 is used to duplicate one input file descriptor from another, or to close one. If *word* evaluates to
 1758 one or more digits, the file descriptor denoted by *n*, or standard input if *n* is not specified, shall
 1759 be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent
 1760 a file descriptor already open for input, a redirection error shall result; see Section 2.8.1 (on page
 1761 2255). If *word* evaluates to '- ', file descriptor *n*, or standard input if *n* is not specified, shall be
 1762 closed. If *word* evaluates to something else, the behavior is unspecified.

1763 **2.7.6 Duplicating an Output File Descriptor**

1764 The redirection operator:

```
1765     [n]>&word
```

1766 is used to duplicate one output file descriptor from another, or to close one. If *word* evaluates to
 1767 one or more digits, the file descriptor denoted by *n*, or standard output if *n* is not specified, shall
 1768 be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent
 1769 a file descriptor already open for output, a redirection error shall result; see Section 2.8.1 (on
 1770 page 2255). If *word* evaluates to '- ', file descriptor *n*, or standard output if *n* is not specified, is
 1771 closed. If *word* evaluates to something else, the behavior is unspecified.

1772 2.7.7 Open File Descriptors for Reading and Writing

1773 The redirection operator:

1774 `[n]<>word`

1775 shall cause the file whose name is the expansion of *word* to be opened for both reading and
1776 writing on the file descriptor denoted by *n*, or standard input if *n* is not specified. If the file does
1777 not exist, it shall be created.

1778 **2.8 Exit Status and Errors**1779 **2.8.1 Consequences of Shell Errors**

1780 For a non-interactive shell, an error condition encountered by a special built-in (see Section 2.15
1781 (on page 2276)) or other type of utility shall cause the shell to write a diagnostic message to
1782 standard error and exit as shown in the following table:

	Error	Special Built-In	Other Utilities
1783			
1784	Shell language syntax error	Shall exit	Shall exit
1785	Utility syntax error (option or operand error)	Shall exit	Shall not exit
1786	Redirection error	Shall exit	Shall not exit
1787	Variable assignment error	Shall exit	Shall not exit
1788	Expansion error	Shall exit	Shall exit
1789	Command not found	N/A	May exit
1790	Dot script not found	Shall exit	N/A

1791 An expansion error is one that occurs when the shell expansions defined in Section 2.6 (on page
1792 2244) are carried out (for example, "\$ {x!y}", because '!' is not a valid operator); an
1793 implementation may treat these as syntax errors if it is able to detect them during tokenization,
1794 rather than during expansion.

1795 If any of the errors shown as "shall exit" or "(may) exit" occur in a subshell, the subshell shall
1796 (or may exit) with a non-zero status, but the script containing the subshell shall not exit because
1797 of the error.

1798 In all of the cases shown in the table, an interactive shell shall write a diagnostic message to
1799 standard error without exiting.

1800 **2.8.2 Exit Status for Commands**

1801 Each command has an exit status that can influence the behavior of other shell commands. The
1802 exit status of commands that are not utilities is documented in this section. The exit status of the
1803 standard utilities is documented in their respective sections.

1804 If a command is not found, the exit status shall be 127. If the command name is found, but it is
1805 not an executable utility, the exit status shall be 126. Applications that invoke utilities without
1806 using the shell should use these exit status values to report similar errors.

1807 If a command fails during word expansion or redirection, its exit status shall be greater than
1808 zero.

1809 Internally, for purposes of deciding whether a command exits with a non-zero exit status, the
1810 shell shall recognize the entire status value retrieved for the command by the equivalent of the
1811 *wait()* function WEXITSTATUS macro (as defined in the System Interfaces volume of
1812 IEEE Std. 1003.1-200x). When reporting the exit status with the special parameter '?', the shell
1813 shall report the full eight bits of exit status available. The exit status of a command that
1814 terminated because it received a signal shall be reported as greater than 128.

1815 2.9 Shell Commands

1816 This section describes the basic structure of shell commands. The following command
1817 descriptions each describe a format of the command that is only used to aid the reader in
1818 recognizing the command type, and does not formally represent the syntax. Each description
1819 discusses the semantics of the command; for a formal definition of the command language,
1820 consult Section 2.11 (on page 2266).

1821 A *command* is one of the following:

- 1822 • *Simple command* (see Section 2.9.1)
- 1823 • *Pipeline* (see Section 2.9.2 (on page 2258))
- 1824 • *List or compound-list* (see Section 2.9.3 (on page 2259))
- 1825 • *Compound command* (see Section 2.9.4 (on page 2261))
- 1826 • *Function definition* (see Section 2.9.5 (on page 2263))

1827 Unless otherwise stated, the exit status of a command is that of the last simple command
1828 executed by the command. There is no limit on the size of any shell command other than that
1829 imposed by the underlying system (memory constraints, {ARG_MAX}, and so on).

1830 2.9.1 Simple Commands

1831 A *simple command* is a sequence of optional variable assignments and redirections, in any
1832 sequence, optionally followed by words and redirections, terminated by a control operator.

1833 When a given simple command is required to be executed (that is, when any conditional
1834 construct such as an AND-OR list or a **case** statement has not bypassed the simple command),
1835 the following expansions, assignments, and redirections are all performed from the beginning of
1836 the command text to the end:

- 1837 1. The words that are recognized as variable assignments or redirections according to Section
1838 2.11.2 (on page 2266) are saved for processing in steps 3 and 4.
- 1839 2. The words that are not variable assignments or redirections shall be expanded. If any fields
1840 remain following their expansion, the first field shall be considered the command name
1841 and remaining fields are the arguments for the command.
- 1842 3. Redirections shall be performed as described in Section 2.7 (on page 2251).
- 1843 4. Each variable assignment shall be expanded for tilde expansion, parameter expansion,
1844 command substitution, arithmetic expansion, and quote removal prior to assigning the
1845 value.

1846 In the preceding list, the order of steps 3 and 4 may be reversed for the processing of special
1847 built-in utilities; see Section 2.15 (on page 2276).

1848 If no command name results, variable assignments shall affect the current execution
1849 environment. Otherwise, the variable assignments shall be exported for the execution
1850 environment of the command and shall not affect the current execution environment (except for
1851 special built-ins). If any of the variable assignments attempt to assign a value to a read-only
1852 variable, a variable assignment error occurs. See Section 2.8.1 (on page 2255) for the
1853 consequences of these errors.

1854 If there is no command name, any redirections shall be performed in a subshell environment; it
1855 is unspecified whether this subshell environment is the same one as that used for a command
1856 substitution within the command. (To affect the current execution environment, see the *exec* (on
1857 page 2287) special built-in.) If any of the redirections performed in the current shell execution

1858 environment fail, the command shall immediately fail with an exit status greater than zero, and
 1859 the shell shall write an error message indicating the failure. See Section 2.8.1 (on page 2255) for
 1860 the consequences of these failures on interactive and non-interactive shells.

1861 If there is a command name, execution shall continue as described in Section 2.9.1.1. If there is
 1862 no command name, but the command contained a command substitution, the command shall
 1863 complete with the exit status of the last command substitution performed. Otherwise, the
 1864 command shall complete with a zero exit status.

1865 2.9.1.1 Command Search and Execution

1866 If a simple command results in a command name and an optional list of arguments, the
 1867 following actions shall be performed:

1868 1. If the command name does not contain any slashes, the first successful step in the
 1869 following sequence shall occur:

1870 a. If the command name matches the name of a special built-in utility, that special
 1871 built-in utility shall be invoked.

1872 b. If the command name matches the name of a function known to this shell, the
 1873 function shall be invoked as described in Section 2.9.5 (on page 2263). If the
 1874 implementation has provided a standard utility in the form of a function, it shall not
 1875 be recognized at this point. It shall be invoked in conjunction with the path search in
 1876 step 1d.

1877 c. If the command name matches the name of a utility listed in the following table, that
 1878 utility shall be invoked.

1879	<i>alias</i>	<i>false</i>	<i>jobs</i>	<i>true</i>
1880	<i>bg</i>	<i>fc</i>	<i>kill</i>	<i>umask</i>
1881	<i>cd</i>	<i>fg</i>	<i>newgrp</i>	<i>unalias</i>
1882	<i>command</i>	<i>getopts</i>	<i>read</i>	<i>wait</i>

1883 d. Otherwise, the command is searched for using the *PATH* environment variable as
 1884 described in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8,
 1885 Environment Variables:

1886 i. If the search is successful:

1887 a. If the system has implemented the utility as a regular built-in or as a shell
 1888 function, it shall be invoked at this point in the path search.

1889 b. Otherwise, the shell executes the utility in a separate utility environment
 1890 (see Section 2.13 (on page 2273)) with actions equivalent to calling the
 1891 *execve()* function as defined in the System Interfaces volume of
 1892 IEEE Std. 1003.1-200x with the *path* argument set to the path name
 1893 resulting from the search, *arg0* set to the command name, and the
 1894 remaining arguments set to the operands, if any.

1895 If the *execve()* function fails due to an error equivalent to the [ENOEXEC]
 1896 error defined in the System Interfaces volume of IEEE Std. 1003.1-200x,
 1897 the shell shall execute a command equivalent to having a shell invoked
 1898 with the command name as its first operand, along with any remaining
 1899 arguments passed along. If the executable file is not a text file, the shell
 1900 may bypass this command execution, write an error message, and return
 1901 an exit status of 126.

1902 Once a utility has been searched for and found (either as a result of this specific
 1903 search or as part of an unspecified shell start-up activity), an implementation
 1904 may remember its location and need not search for the utility again unless the
 1905 *PATH* variable has been the subject of an assignment. If the remembered
 1906 location fails for a subsequent invocation, the shell shall repeat the search to
 1907 find the new location for the utility, if any.

1908 ii. If the search is unsuccessful, the command shall fail with an exit status of 127
 1909 and the shell shall write an error message.

1910 2. If the command name contains at least one slash, the shell shall execute the utility in a
 1911 separate utility environment with actions equivalent to calling the *execve()* function
 1912 defined in the System Interfaces volume of IEEE Std. 1003.1-200x with the *path* and *arg0*
 1913 arguments set to the command name, and the remaining arguments set to the operands, if
 1914 any.

1915 If the *execve()* function fails due to an error equivalent to the [ENOEXEC] error, the shell
 1916 shall execute a command equivalent to having a shell invoked with the command name as
 1917 its first operand, along with any remaining arguments passed along. If the executable file is
 1918 not a text file, the shell may bypass this command execution, write an error message, and
 1919 return an exit status of 126.

1920 2.9.2 Pipelines

1921 A *pipeline* is a sequence of one or more commands separated by the control operator '*|*'. The
 1922 standard output of all but the last command shall be connected to the standard input of the next
 1923 command.

1924 The format for a pipeline is:

```
1925                   [!] command1 [ | command2 ... ]
```

1926 The standard output of *command1* shall be connected to the standard input of *command2*. The
 1927 standard input, standard output, or both of a command shall be considered to be assigned by the
 1928 pipeline before any redirection specified by redirection operators that are part of the command
 1929 (see Section 2.7 (on page 2251)).

1930 If the pipeline is not in the background (see Section 2.9.3.1 (on page 2259)), the shell shall wait for
 1931 the last command specified in the pipeline to complete, and may also wait for all commands to
 1932 complete.

1933 Exit Status

1934 If the reserved word *!* does not precede the pipeline, the exit status shall be the exit status of the
 1935 last command specified in the pipeline. Otherwise, the exit status shall be the logical NOT of the
 1936 exit status of the last command. That is, if the last command returns zero, the exit status shall be
 1937 1; if the last command returns greater than zero, the exit status shall be zero.

1938 **2.9.3 Lists**

1939 An *AND-OR list* is a sequence of one or more pipelines separated by the operators "&&" and
 1940 "||".

1941 A *list* is a sequence of one or more AND-OR lists separated by the operators ';' and '&' and
 1942 optionally terminated by ';', '&', or <newline>.

1943 The operators "&&" and "||" shall have equal precedence and are evaluated from beginning to
 1944 end. For example, both of the following commands write solely **bar** to standard output:

```
1945     false && echo foo || echo bar
1946     true  || echo foo && echo bar
```

1947 A ';' or <newline> character terminator shall cause the preceding AND-OR list to be executed
 1948 sequentially; an '&' shall cause asynchronous execution of the preceding AND-OR list.

1949 The term *compound-list* is derived from the grammar in Section 2.11 (on page 2266); it is
 1950 equivalent to a sequence of *lists*, separated by <newline> characters, that can be preceded or
 1951 followed by an arbitrary number of <newline> characters.

1952 **Examples**

1953 The following is an example that illustrates <newline> characters in compound-lists:

```
1954     while
1955         # a couple of <newline>s
1956         # a list
1957         date && who || ls; cat file
1958         # a couple of <newline>s
1959         # another list
1960         wc file > output & true
1961     do
1962         # 2 lists
1963         ls
1964         cat file
1965     done
```

1966 **2.9.3.1 Asynchronous Lists**

1967 If a command is terminated by the control operator ampersand ('&'), the shell shall execute the
 1968 command asynchronously in a subshell. This means that the shell shall not wait for the
 1969 command to finish before executing the next command.

1970 The format for running a command in the background is:

```
1971     command1 & [command2 & ... ]
```

1972 The standard input for an asynchronous list, before any explicit redirections are performed, shall
 1973 be considered to be assigned to a file that has the same properties as **/dev/null**. If it is an
 1974 interactive shell, this need not happen. In all cases, explicit redirection of standard input shall
 1975 override this activity.

1976 When an element of an asynchronous list (the portion of the list ended by an ampersand, such as
 1977 *command1*, above) is started by the shell, the process ID of the last command in the asynchronous
 1978 list element shall become known in the current shell execution environment; see Section 2.13 (on
 1979 page 2273). This process ID shall remain known until:

- 1980 1. The command terminates and the application waits for the process ID.
1981 2. Another asynchronous list invoked before "\$!" (corresponding to the previous
1982 asynchronous list) is expanded in the current execution environment.

1983 The implementation need not retain more than the {CHILD_MAX} most recent entries in its list
1984 of known process IDs in the current shell execution environment.

1985 **Exit Status**

1986 The exit status of an asynchronous list shall be zero.

1987 **2.9.3.2 Sequential Lists**

1988 Commands that are separated by a semicolon (;) shall be executed sequentially.

1989 The format for executing commands sequentially shall be:

1990 `command1 [; command2] . . .`

1991 Each command shall be expanded and executed in the order specified.

1992 **Exit Status**

1993 The exit status of a sequential list shall be the exit status of the last command in the list.

1994 **2.9.3.3 AND Lists**

1995 The control operator "&&" denotes an AND list. The format shall be:

1996 `command1 [&& command2] . . .`

1997 First *command1* shall be executed. If its exit status is zero, *command2* shall be executed, and so on,
1998 until a command has a non-zero exit status or there are no more commands left to execute. The
1999 commands are expanded only if they are executed.

2000 **Exit Status**

2001 The exit status of an AND list shall be the exit status of the last command that is executed in the
2002 list.

2003 **2.9.3.4 OR Lists**

2004 The control operator " || " denotes an OR List. The format shall be:

2005 `command1 [|| command2] . . .`

2006 First, *command1* shall be executed. If its exit status is non-zero, *command2* shall be executed, and
2007 so on, until a command has a zero exit status or there are no more commands left to execute.

2008 **Exit Status**

2009 The exit status of an OR list shall be the exit status of the last command that is executed in the
2010 list.

2011 **2.9.4 Compound Commands**

2012 The shell has several programming constructs that are *compound commands*, which provide
 2013 control flow for commands. Each of these compound commands has a reserved word or control
 2014 operator at the beginning, and a corresponding terminator reserved word or operator at the end.
 2015 In addition, each can be followed by redirections on the same line as the terminator. Each
 2016 redirection shall apply to all the commands within the compound command that do not
 2017 explicitly override that redirection.

2018 **2.9.4.1 Grouping Commands**

2019 The format for grouping commands is as follows:

2020 (*compound-list*) Execute *compound-list* in a subshell environment; see Section 2.13 (on page
 2021 2273). Variable assignments and built-in commands that affect the
 2022 environment shall not remain in effect after the list finishes.

2023 { *compound-list*; } Execute *compound-list* in the current process environment. The semicolon
 2024 shown here is an example of a control operator delimiting the } reserved
 2025 word. Other delimiters are possible, as shown in Section 2.11 (on page
 2026 2266); a <newline> character is frequently used.

2027 **Exit Status**

2028 The exit status of a grouping command shall be the exit status of *list*.

2029 **2.9.4.2 For Loop**

2030 The **for** loop executes a sequence of commands for each member in a list of *items*. The **for** loop
 2031 requires that the reserved words **do** and **done** be used to delimit the sequence of commands.

2032 The format for the **for** loop is as follows:

```
2033     for name [ in [word ... ] ]
2034     do
2035         compound-list
2036     done
```

2037 First, the list of words following **in** shall be expanded to generate a list of items. Then, the
 2038 variable *name* shall be set to each item, in turn, and the *compound-list* executed each time. If no
 2039 items result from the expansion, the *compound-list* shall not be executed. Omitting:

```
2040     in word...
```

2041 is equivalent to:

```
2042     in "$@"
```

2043 **Exit Status**

2044 The exit status of a **for** command shall be the exit status of the last command that executes. If
 2045 there are no items, the exit status shall be zero.

2046 2.9.4.3 *Case Conditional Construct*

2047 The conditional construct **case** shall execute the *compound-list* corresponding to the first one of
 2048 several *patterns* (see Section 2.14 (on page 2274)) that is matched by the string resulting from the
 2049 tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote
 2050 removal of the given word. The reserved word **in** shall denote the beginning of the patterns to be
 2051 matched. Multiple patterns with the same *compound-list* shall be delimited by the '|' symbol.
 2052 The control operator ')' terminates a list of patterns corresponding to a given action. The
 2053 *compound-list* for each list of patterns, with the possible exception of the last, shall be terminated
 2054 with ";". The **case** construct terminates with the reserved word **esac** (**case** reversed).

2055 The format for the **case** construct is as follows:

```
2056     case word in
2057         [( ]pattern1) compound-list;;
2058         [( ]pattern[ | pattern] ... ) compound-list;;] ...
2059         [( ]pattern[ | pattern] ... ) compound-list]
2060     esac
```

2061 The ";" is optional for the last *compound-list*.

2062 In order from the beginning to the end of the **case** statement, each *pattern* that labels a
 2063 *compound-list* shall be subjected to tilde expansion, parameter expansion, command substitution,
 2064 and arithmetic expansion, and the result of these expansions shall be compared against the
 2065 expansion of *word*, according to the rules described in Section 2.14 (on page 2274) (which also
 2066 describes the effect of quoting parts of the pattern). After the first match, no more patterns shall
 2067 be expanded, and the *compound-list* shall be executed. The order of expansion and comparison of
 2068 multiple *patterns* that label a *compound-list* statement is unspecified.

2069 **Exit Status**

2070 The exit status of **case** shall be zero if no patterns are matched. Otherwise, the exit status shall be
 2071 the exit status of the last command executed in the *compound-list*.

2072 2.9.4.4 *If Conditional Construct*

2073 The **if** command shall execute a *compound-list* and use its exit status to determine whether to
 2074 execute another *compound-list*.

2075 The format for the **if** construct is as follows:

```
2076     if compound-list
2077     then
2078         compound-list
2079     [elif compound-list
2080     then
2081         compound-list] ...
2082     [else
2083         compound-list]
```

2084 The **if** *compound-list* shall be executed; if its exit status is zero, the **then** *compound-list* shall be
 2085 executed and the command shall complete. Otherwise, each **elif** *compound-list* shall be executed,
 2086 in turn, and if its exit status is zero, the **then** *compound-list* shall be executed and the command
 2087 shall complete. Otherwise, the **else** *compound-list* shall be executed.

2088 **Exit Status**

2089 The exit status of the **if** command shall be the exit status of the **then** or **else** *compound-list* that
 2090 was executed, or zero, if none was executed.

2091 2.9.4.5 *While Loop*

2092 The **while** loop shall continuously execute one *compound-list* as long as another *compound-list* has
 2093 a zero exit status.

2094 The format of the **while** loop is as follows:

```
2095            while compound-list-1
2096            do
2097                compound-list-2
2098            done
```

2099 The *compound-list-1* shall be executed, and if it has a non-zero exit status, the **while** command
 2100 shall complete. Otherwise, the *compound-list-2* shall be executed, and the process shall repeat.

2101 **Exit Status**

2102 The exit status of the **while** loop shall be the exit status of the last *compound-list-2* executed, or
 2103 zero if none was executed.

2104 2.9.4.6 *Until Loop*

2105 The **until** loop shall continuously execute one *compound-list* as long as another *compound-list* has
 2106 a non-zero exit status.

2107 The format of the **until** loop is as follows:

```
2108            until compound-list-1
2109            do
2110                compound-list-2
2111            done
```

2112 The *compound-list-1* shall be executed, and if it has a zero exit status, the **until** command
 2113 completes. Otherwise, the *compound-list-2* shall be executed, and the process repeats.

2114 **Exit Status**

2115 The exit status of the **until** loop shall be the exit status of the last *compound-list-2* executed, or
 2116 zero if none was executed.

2117 2.9.5 **Function Definition Command**

2118 A function is a user-defined name that is used as a simple command to call a compound
 2119 command with new positional parameters. A function is defined with a *function definition*
 2120 *command*.

2121 The format of a function definition command is as follows:

```
2122            fname() compound-command[io-redirect ...]
```

2123 The function is named *fname*; the application shall ensure that it is a name (see the Base
 2124 Definitions volume of IEEE Std. 1003.1-200x, Section 3.232, Name). An implementation may
 2125 allow other characters in a function name as an extension. The implementation shall maintain
 2126 separate name spaces for functions and variables.

2127 The argument *compound-command* represents a compound command, as described in Section
2128 2.9.4 (on page 2261).

2129 When the function is declared, none of the expansions in Section 2.6 (on page 2244) shall be
2130 performed on the text in *compound-command* or *io-redirect*; all expansions shall be performed as
2131 normal each time the function is called. Similarly, the optional *io-redirect* redirections and any
2132 variable assignments within *compound-command* shall be performed during the execution of the
2133 function itself, not the function definition. See Section 2.8.1 (on page 2255) for the consequences
2134 of failures of these operations on interactive and non-interactive shells.

2135 When a function is executed, it shall have the syntax-error and variable-assignment properties
2136 described for special built-in utilities in the enumerated list at the beginning of Section 2.15 (on
2137 page 2276).

2138 The *compound-command* shall be executed whenever the function name is specified as the name
2139 of a simple command (see Section 2.9.1.1 (on page 2257)). The operands to the command
2140 temporarily shall become the positional parameters during the execution of the *compound-*
2141 *command*; the special parameter '# ' also shall be changed to reflect the number of operands. The
2142 special parameter 0 shall be unchanged. When the function completes, the values of the
2143 positional parameters and the special parameter '# ' shall be restored to the values they had
2144 before the function was executed. If the special built-in *return* is executed in the *compound-*
2145 *command*, the function completes and execution shall resume with the next command after the
2146 function call.

2147 **Exit Status**

2148 The exit status of a function definition shall be zero if the function was declared successfully;
2149 otherwise, it shall be greater than zero. The exit status of a function invocation shall be the exit
2150 status of the last command executed by the function.

2.10 Executable Script

2152 XSI XSI-Conformant systems shall support executable scripts. A successful call to a function of the
2153 *exec* family with an executable script as the first parameter shall result in a new process, where
2154 the process image that is started is that of the interpreter. The path name of the interpreter
2155 follows the "#!" characters.

2156 If the executable script has a first line:

```
2157 #! interpreter [arg]
```

2158 then the interpreter shall be called with an argument array consisting of an unspecified zero'th
2159 argument, followed by *arg* (if present), followed by a path name for the script, followed by the
2160 arguments following the zero'th argument in the *exec* call of the script.

2161 No shell operations (as described in Section 2.1 (on page 2235)) shall be performed on the first
2162 line of an executable script.

2163 The behavior shall be unspecified if the first line of the executable script does not meet all of the
2164 following criteria:

2165 1. The first line shall be in one of the formats below:

```
2166 "#!%s\n" interpreter
```

```
2167 "#!<delta>%s\n" interpreter
```

```
2168 "#!%s<delta>%s\n" interpreter arg
```

```
2169 "#!<delta>%s<delta>%s\n" interpreter arg
```

2170 2. The *interpreter* argument shall be an absolute path name of an executable file other than an
2171 executable script.

2172 3. The *interpreter* argument and the *arg* argument, if present, shall not contain any quoting
2173 characters.

2174 4. The *interpreter* argument and the *arg* argument, if present, shall not contain any white-
2175 space characters.

2176 5. The length of the first line shall be no longer than 80 bytes.
2177

2178 2.11 Shell Grammar

2179 The following grammar defines the Shell Command Language. This formal syntax shall take
2180 precedence over the preceding text syntax description.

2181 2.11.1 Shell Grammar Lexical Conventions

2182 The input language to the shell must be first recognized at the character level. The resulting
2183 tokens shall be classified by their immediate context according to the following rules (applied in
2184 order). These rules are used to determine what a “token” is that is subject to parsing at the
2185 token level. The rules for token recognition in Section 2.3 (on page 2238) shall apply.

- 2186 1. A <newline> character shall be returned as the token identifier **NEWLINE**.
- 2187 2. If the token is an operator, the token identifier for that operator shall result.
- 2188 3. If the string consists solely of digits and the delimiter character is one of '<' or '>', the
2189 token identifier **IO_NUMBER** shall be returned.
- 2190 4. Otherwise, the token identifier **TOKEN** results.

2191 Further distinction on **TOKEN** is context-dependent. It may be that the same **TOKEN** yields
2192 **WORD**, a **NAME**, an **ASSIGNMENT**, or one of the reserved words below, dependent upon the
2193 context. Some of the productions in the grammar below are annotated with a rule number from
2194 the following list. When a **TOKEN** is seen where one of those annotated productions could be
2195 used to reduce the symbol, the applicable rule shall be applied to convert the token identifier
2196 type of the **TOKEN** to a token identifier acceptable at that point in the grammar. The reduction
2197 shall then proceed based upon the token identifier type yielded by the rule applied. When more
2198 than one rule applies, the highest numbered rule shall apply (which in turn may refer to another
2199 rule). (Note that except in rule 7, the presence of an '=' in the token has no effect.)

2200 The **WORD** tokens shall have the word expansion rules applied to them immediately before the
2201 associated command is executed, not at the time the command is parsed.

2202 2.11.2 Shell Grammar Rules

- 2203 1. [Command Name]

2204 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word
2205 shall result. Otherwise, the token **WORD** shall be returned. Also, if the parser is in any
2206 state where only a reserved word could be the next correct token, proceed as above. This
2207 rule applies rather narrowly: when a compound list is terminated by some clear delimiter
2208 (such as the closing **fi** of an inner **if_clause**) then it would apply; where the compound list
2209 might continue (as in after a ';'), rule 7a (and consequently the first sentence of this rule)
2210 would apply. In many instances the two conditions are identical, but this part of this rule
2211 does not give license to treating a **WORD** as a reserved word unless it is in a place where a
2212 reserved word shall appear.

2213 **Note:** Because at this point quote marks are retained in the token, quoted strings
2214 cannot be recognized as reserved words. This rule also implies that reserved
2215 words are not recognized except in certain positions in the input, such as after a
2216 <newline> character or semicolon; the grammar presumes that if the reserved
2217 word is intended, it is properly delimited by the user, and does not attempt to
2218 reflect that requirement directly. Also note that line joining is done before
2219 tokenization, as described in Section 2.2.1 (on page 2236), so escaped
2220 <newline>s are already removed at this point.

- 2221 Rule 1 is not directly referenced in the grammar, but is referred to by other rules, or applies
2222 globally.
- 2223 2. [Redirection to or from file name]
- 2224 The expansions specified in Section 2.7 (on page 2251) shall occur. As specified there,
2225 exactly one field can result (or the result is unspecified), and there are additional
2226 requirements on path name expansion.
- 2227 3. [Redirection from here-document]
- 2228 Quote removal shall be applied to the word to determine the delimiter that is used to find
2229 the end of the here-document that begins after the next <newline> character.
- 2230 4. [Case statement termination]
- 2231 When the **TOKEN** is exactly the reserved word **esac**, the token identifier for **esac** shall
2232 result. Otherwise, the token **WORD** shall be returned.
- 2233 5. [**NAME** in **for**]
- 2234 When the **TOKEN** meets the requirements for a name (see the Base Definitions volume of
2235 IEEE Std. 1003.1-200x, Section 3.232, Name), the token identifier **NAME** shall result. |
2236 Otherwise, the token **WORD** shall be returned.
- 2237 6. [Third word of **for** and **case**]
- 2238 When the **TOKEN** is exactly the reserved word **in**, the token identifier for **in** shall result.
2239 Otherwise, the token **WORD** shall be returned. (As indicated in the grammar, a *linebreak*
2240 precedes the token **in**. If <newline> characters are present at the indicated location, it is
2241 the token after them that is treated in this fashion.)
- 2242 7. [Assignment preceding command name]
- 2243 a. [When the first word]
- 2244 If the **TOKEN** does not contain the character '=' , rule 1 is applied. Otherwise, 7b
2245 shall be applied.
- 2246 b. [Not the first word]
- 2247 If the **TOKEN** contains the equal sign character:
- 2248 — If it begins with '=' , the token **WORD** shall be returned.
- 2249 — If all the characters preceding '=' form a valid name (see the Base Definitions |
2250 volume of IEEE Std. 1003.1-200x, Section 3.232, Name), the token |
2251 **ASSIGNMENT_WORD** shall be returned. (Quoted characters cannot participate
2252 in forming a valid name.)
- 2253 — Otherwise, it is unspecified whether it is **ASSIGNMENT_WORD** or **WORD** that
2254 is returned.
- 2255 Assignment to the **NAME** shall occur as specified in Section 2.9.1 (on page 2256).
- 2256 8. [**NAME** in function]
- 2257 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word
2258 shall result. Otherwise, when the **TOKEN** meets the requirements for a name, the token
2259 identifier **NAME** shall result. Otherwise, rule 7 applies.
- 2260 9. [Body of function]

```

2261         Word expansion and assignment shall never occur, even when required by the rules above,
2262         when this rule is being parsed. Each TOKEN that might either be expanded or have
2263         assignment applied to it shall instead be returned as a single WORD consisting only of
2264         characters that are exactly the token described in Section 2.3 (on page 2238).

2265         /* -----
2266         The grammar symbols
2267         ----- */

2268         %token WORD
2269         %token ASSIGNMENT_WORD
2270         %token NAME
2271         %token NEWLINE
2272         %token IO_NUMBER

2273         /* The following are the operators mentioned above. */

2274         %token AND_IF OR_IF DSEMI
2275         /* '&&' '|'| ';' */

2276         %token DLESS DGREAT LESSAND GREATAND LESSGREAT DLESSDASH
2277         /* '<<' '>>' '<&' '>&' '<>' '<<-' */

2278         %token CLOBBER
2279         /* '>|' */

2280         /* The following are the reserved words. */

2281         %token If Then Else Elif Fi Do Done
2282         /* 'if' 'then' 'else' 'elif' 'fi' 'do' 'done' */

2283         %token Case Esac While Until For
2284         /* 'case' 'esac' 'while' 'until' 'for' */

2285         /* These are reserved words, not operator tokens, and are
2286         recognized when reserved words are recognized. */

2287         %token Lbrace Rbrace Bang
2288         /* '{' '}' '!' */

2289         %token In
2290         /* 'in' */

2291         /* -----
2292         The Grammar
2293         ----- */

2294         %start complete_command
2295         %%
2296         complete_command : list separator
2297                         | list
2298                         ;
2299         list              : list separator_op and_or
2300                         |                               and_or
2301                         ;
2302         and_or            :                               pipeline
2303                         | and_or AND_IF linebreak pipeline
2304                         | and_or OR_IF linebreak pipeline
2305                         ;

```

```

2306     pipeline      :      pipe_sequence
2307                   | Bang pipe_sequence
2308                   ;
2309     pipe_sequence  :                                  command
2310                   | pipe_sequence '|' linebreak command
2311                   ;
2312     command        : simple_command
2313                   | compound_command
2314                   | compound_command redirect_list
2315                   | function_definition
2316                   ;
2317     compound_command : brace_group
2318                   | subshell
2319                   | for_clause
2320                   | case_clause
2321                   | if_clause
2322                   | while_clause
2323                   | until_clause
2324                   ;
2325     subshell       : '(' compound_list ')'
2326                   ;
2327     compound_list  :      term
2328                   | newline_list term
2329                   |      term separator
2330                   | newline_list term separator
2331                   ;
2332     term           : term separator and_or
2333                   |      and_or
2334                   ;
2335     for_clause     : For name linebreak      do_group
2336                   | For name linebreak in  sequential_sep do_group
2337                   | For name linebreak in wordlist sequential_sep do_group
2338                   ;
2339     name           : NAME /* Apply rule 5 */
2340                   ;
2341     in             : In /* Apply rule 6 */
2342                   ;
2343     wordlist       : wordlist WORD
2344                   |      WORD
2345                   ;
2346     case_clause    : Case WORD linebreak in linebreak case_list Esac
2347                   | Case WORD linebreak in linebreak case_list_ns Esac
2348                   | Case WORD linebreak in linebreak Esac
2349                   ;
2350     case_list_ns   : case_list case_item_ns
2351                   | case_item_ns
2352                   ;
2353     case_list      : case_list case_item
2354                   | case_item
2355                   ;
2356     case_item_ns   : pattern ')' linebreak linebreak
2357                   | pattern ')' compound_list linebreak

```

```

2358         | '(' pattern ')' linebreak linebreak
2359         | '(' pattern ')' compound_list linebreak
2360         ;
2361     case_item      : pattern ')' linebreak DSEMI linebreak
2362         | pattern ')' compound_list linebreak
2363         | '(' pattern ')' linebreak linebreak
2364         | '(' pattern ')' compound_list linebreak
2365         ;
2366     pattern       :          WORD          /* Apply rule 4 */
2367         | pattern '|' WORD          /* Do not apply rule (4) */
2368         ;
2369     if_clause     : If compound_list Then compound_list else_part Fi
2370         | If compound_list Then compound_list          Fi
2371         ;
2372     else_part     : Elif compound_list Then else_part
2373         | Else compound_list
2374         ;
2375     while_clause  : While compound_list do_group
2376         ;
2377     until_clause  : Until compound_list do_group
2378         ;
2379     function_definition : fname '(' ')' linebreak function_body
2380         ;
2381     function_body : compound_command          /* Apply rule 9 */
2382         | compound_command redirect_list /* Apply rule 9 */
2383         ;
2384     fname         : NAME          /* Apply rule 8 */
2385         ;
2386     brace_group   : Lbrace compound_list Rbrace
2387         ;
2388     do_group      : Do compound_list Done
2389         ;
2390     simple_command : cmd_prefix cmd_word cmd_suffix
2391         | cmd_prefix cmd_word
2392         | cmd_prefix
2393         | cmd_name cmd_suffix
2394         | cmd_name
2395         ;
2396     cmd_name      : WORD          /* Apply rule 7a */
2397         ;
2398     cmd_word      : WORD          /* Apply rule 7b */
2399         ;
2400     cmd_prefix    :          io_redirect
2401         | cmd_prefix io_redirect
2402         |          ASSIGNMENT_WORD
2403         | cmd_prefix ASSIGNMENT_WORD
2404         ;
2405     cmd_suffix    :          io_redirect
2406         | cmd_suffix io_redirect
2407         |          WORD
2408         | cmd_suffix WORD
2409         ;

```

```

2410     redirect_list      :           io_redirect
2411                          | redirect_list io_redirect
2412                          ;
2413     io_redirect         :           io_file
2414                          | IO_NUMBER io_file
2415                          |           io_here
2416                          | IO_NUMBER io_here
2417                          ;
2418     io_file             : '<'      filename
2419                          | LESSAND filename
2420                          | '>'      filename
2421                          | GREATAND filename
2422                          | DGREAT  filename
2423                          | LESSGREAT filename
2424                          | CLOBBER  filename
2425                          ;
2426     filename            : WORD           /* Apply rule 2 */
2427                          ;
2428     io_here              : DLESS      here_end
2429                          | DLESSDASH here_end
2430                          ;
2431     here_end             : WORD           /* Apply rule 3 */
2432                          ;
2433     newline_list        :           NEWLINE
2434                          | newline_list NEWLINE
2435                          ;
2436     linebreak           : newline_list
2437                          | /* empty */
2438                          ;
2439     separator_op        : '&'
2440                          | ';'
2441                          ;
2442     separator            : separator_op linebreak
2443                          | newline_list
2444                          ;
2445     sequential_sep      : ';' linebreak
2446                          | newline_list
2447                          ;

```

2448 2.12 Signals and Error Handling

2449 When a command is in an asynchronous list, the shell shall prevent SIGQUIT and SIGINT
2450 signals from the keyboard from interrupting the command. Otherwise, signals shall have the
2451 values inherited by the shell from its parent (see also the *trap* (on page 2307) special built-in).

2452 When a signal for which a trap has been set is received while the shell is waiting for the
2453 completion of a utility executing a foreground command, the trap associated with that signal
2454 shall not be executed until after the foreground command has completed. When the shell is
2455 waiting, by means of the *wait* utility, for asynchronous commands to complete, the reception of a
2456 signal for which a trap has been set shall cause the *wait* utility to return immediately with an exit
2457 status >128, immediately after which the trap associated with that signal shall be taken.

2458 If multiple signals are pending for the shell for which there are associated trap actions, the order
2459 of execution of trap actions is unspecified.

2460 2.13 Shell Execution Environment

2461 A shell execution environment consists of the following:

- 2462 • Open files inherited upon invocation of the shell, plus open files controlled by *exec*
- 2463 • Working directory as set by *cd*
- 2464 • File creation mask set by *umask*
- 2465 • Current traps set by *trap*
- 2466 • Shell parameters that are set by variable assignment (see the *set* (on page 2297) special built-
- 2467 in) or from the System Interfaces volume of IEEE Std. 1003.1-200x environment inherited by
- 2468 the shell when it begins (see the *export* (on page 2291) special built-in)
- 2469 • Shell functions; see Section 2.9.5 (on page 2263)
- 2470 • Options turned on at invocation or by *set*
- 2471 • Process IDs of the last commands in asynchronous lists known to this shell environment; see
- 2472 Section 2.9.3.1 (on page 2259)
- 2473 • Shell aliases; see Section 2.3.1 (on page 2239)

2474 Utilities other than the special built-ins (see Section 2.15 (on page 2276)) shall be invoked in a
2475 separate environment that consists of the following. The initial value of these objects shall be the
2476 same as that for the parent shell, except as noted below.

- 2477 • Open files inherited on invocation of the shell, open files controlled by the *exec* special built-
- 2478 in plus any modifications, and additions specified by any redirections to the utility
- 2479 • Current working directory
- 2480 • File creation mask
- 2481 • If the utility is a shell script, traps caught by the shell shall be set to the default values and
- 2482 traps ignored by the shell shall be set to be ignored by the utility; if the utility is not a shell
- 2483 script, the trap actions (default or ignore) shall be mapped into the appropriate signal
- 2484 handling actions for the utility
- 2485 • Variables with the *export* attribute, along with those explicitly exported for the duration of the
- 2486 command, shall be passed to the utility as System Interfaces volume of IEEE Std. 1003.1-200x
- 2487 environment variables

2488 The environment of the shell process shall not be changed by the utility unless explicitly
2489 specified by the utility description (for example, *cd* and *umask*).

2490 A subshell environment shall be created as a duplicate of the shell environment, except that
2491 signal traps set by that shell environment shall be set to the default values. Changes made to the
2492 subshell environment shall not affect the shell environment. Command substitution, commands
2493 that are grouped with parentheses, and asynchronous lists shall be executed in a subshell
2494 environment. Additionally, each command of a multi-command pipeline is in a subshell
2495 environment; as an extension, however, any or all commands in a pipeline may be executed in
2496 the current environment. All other commands shall be executed in the current shell
2497 environment.

2498 2.14 Pattern Matching Notation

2499 The pattern matching notation described in this section is used to specify patterns for matching
2500 strings in the shell. Historically, pattern matching notation is related to, but slightly different
2501 from, the regular expression notation described in the Base Definitions volume of
2502 IEEE Std. 1003.1-200x, Chapter 9, Regular Expressions. For this reason, the description of the
2503 rules for this pattern matching notation are based on the description of regular expression
2504 notation, modified to include backslash escape processing.

2505 2.14.1 Patterns Matching a Single Character

2506 The following *patterns matching a single character* match a single character: *ordinary characters*,
2507 *special pattern characters*, and *pattern bracket expressions*. The pattern bracket expression also shall
2508 match a single collating element. A backslash character shall escape the following character. The
2509 escaping backslash shall be discarded.

2510 An ordinary character is a pattern that shall match itself. It can be any character in the supported
2511 character set except for NUL, those special shell characters in Section 2.2 (on page 2236) that
2512 require quoting, and the following three special pattern characters. Matching shall be based on
2513 the bit pattern used for encoding the character, not on the graphic representation of the
2514 character. If any character (ordinary, shell special, or pattern special) is quoted, that pattern shall
2515 match the character itself. The shell special characters always require quoting.

2516 When unquoted and outside a bracket expression, the following three characters shall have
2517 special meaning in the specification of patterns:

- 2518 ? A question-mark is a pattern that shall match any character.
- 2519 * An asterisk is a pattern that shall match multiple characters, as described in Section 2.14.2.
- 2520 [The open bracket shall introduce a pattern bracket expression.

2521 The description of basic regular expression bracket expressions in the Base Definitions volume
2522 of IEEE Std. 1003.1-200x, Section 9.3.5, RE Bracket Expression shall also apply to the pattern
2523 bracket expression, except that the exclamation mark character ('!') shall replace the
2524 circumflex character ('^') in its role in a *non-matching list* in the regular expression notation. A
2525 bracket expression starting with an unquoted circumflex character produces unspecified results.

2526 When pattern matching is used where shell quote removal is not performed (such as in the
2527 argument to the *find name* primary when *find* is being called using one of the *exec* functions as
2528 defined in the System Interfaces volume of IEEE Std. 1003.1-200x, or in the *pattern* argument to
2529 the *fnmatch()* function), special characters can be escaped to remove their special meaning by
2530 preceding them with a backslash character. This escaping backslash is discarded. The sequence
2531 "\\\" represents one literal backslash. All of the requirements and effects of quoting on ordinary,
2532 shell special, and special pattern characters shall apply to escaping in this context.

2533 2.14.2 Patterns Matching Multiple Characters

2534 The following rules are used to construct *patterns matching multiple characters* from *patterns*
2535 *matching a single character*:

- 2536 1. The asterisk ('*') is a pattern that shall match any string, including the null string.
- 2537 2. The concatenation of *patterns matching a single character* is a valid pattern that shall match
2538 the concatenation of the single characters or collating elements matched by each of the
2539 concatenated patterns.

- 2540 3. The concatenation of one or more *patterns matching a single character* with one or more
 2541 asterisks is a valid pattern. In such patterns, each asterisk shall match a string of zero or
 2542 more characters, matching the greatest possible number of characters that still allows the
 2543 remainder of the pattern to match the string.

2544 2.14.3 Patterns Used for File Name Expansion

2545 The rules described so far in Section 2.14.1 (on page 2274) and Section 2.14.2 (on page 2274) are
 2546 qualified by the following rules that apply when pattern matching notation is used for file name
 2547 expansion:

- 2548 1. The application shall ensure that the slash character in a path name is explicitly matched
 2549 by using one or more slashes in the pattern; it cannot be matched by the asterisk or
 2550 question-mark special characters or by a bracket expression. Slashes in the pattern are
 2551 identified before bracket expressions; thus, a slash cannot be included in a pattern bracket
 2552 expression used for file name expansion. If a slash character is found following an
 2553 unescaped open square bracket character before a corresponding closing square bracket is
 2554 found, the open bracket is treated as an ordinary character. For example, the pattern
 2555 "a[b/c]d" does not match such path names as **abd** or **a/d**. It only matches a path name
 2556 of literally **a[b/c]d**.
- 2557 2. If a file name begins with a period ('.'), the application shall ensure that the period is
 2558 explicitly matched by using a period as the first character of the pattern or immediately
 2559 following a slash character. The leading period shall not be matched by:

- 2560 • The asterisk or question-mark special characters
- 2561 • A bracket expression containing a non-matching list, such as "[!a]", a range
 2562 expression, such as "[%-0]", or a character class expression, such as "[[:punct:]]"

2563 It is unspecified whether an explicit period in a bracket expression matching list, such as
 2564 "[.abc]", can match a leading period in a file name.

- 2565 3. Specified patterns are matched against existing file names and path names, as appropriate.
 2566 Each component that contains a pattern character requires read permission in the directory
 2567 containing that component. Any component, except the last, that does not contain a
 2568 pattern character requires search permission. For example, given the pattern:

```
2569 /foo/bar/x*/bam
```

2570 search permission is needed for directories / and **foo**, search and read permissions are
 2571 needed for directory **bar**, and search permission is needed for each **x*** directory. If the
 2572 pattern matches any existing file names or path names, the pattern shall be replaced with
 2573 those file names and path names, sorted according to the collating sequence in effect in the
 2574 current locale. If the pattern contains an invalid bracket expression or does not match any
 2575 existing file names or path names, the pattern string shall be left unchanged.

2576 **2.15 Special Built-In Utilities**

2577 The following *special built-in* utilities shall be supported in the shell command language. The
2578 output of each command, if any, shall be written to standard output, subject to the normal
2579 redirection and piping possible with all commands.

2580 The term *built-in* implies that the shell can execute the utility directly and does not need to
2581 search for it. An implementation can choose to make any utility a built-in; however, the special
2582 built-in utilities described here differ from regular built-in utilities in two respects:

- 2583 1. A syntax error in a special built-in utility may cause a shell executing that utility to abort,
2584 while a syntax error in a regular built-in utility shall not cause a shell executing that utility
2585 to abort. (See Section 2.8.1 (on page 2255) for the consequences of errors on interactive and
2586 non-interactive shells.) If a special built-in utility encountering a syntax error does not
2587 abort the shell, its exit value shall be non-zero.
- 2588 2. Variable assignments specified with special built-in utilities remain in effect after the
2589 built-in completes; this shall not be the case with a regular built-in or other utility.

2590 The special built-in utilities in this section need not be provided in a manner accessible via the
2591 *exec* family of functions defined in the System Interfaces volume of IEEE Std. 1003.1-200x.

2592 Some of the special built-ins are described as conforming to the Base Definitions volume of |
2593 IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines. For those that are not, the |
2594 requirement in Section 1.11 (on page 2224) that "—" be recognized as a first argument to be |
2595 discarded does not apply and a portable application shall not use that argument.

2596 **NAME**

2597 break — exit from for, while, or until loop

2598 **SYNOPSIS**2599 break [*n*]2600 **DESCRIPTION**

2601 The *break* utility shall exit from the smallest enclosing **for**, **while**, or **until** loop, if any; or from the
2602 *n*th enclosing loop if *n* is specified. The value of *n* is an unsigned decimal integer greater than or
2603 equal to 1. The default shall be equivalent to *n*=1. If *n* is greater than the number of enclosing
2604 loops, the last enclosing loop shall be exited from. Execution shall continue with the command
2605 immediately following the loop.

2606 **OPTIONS**

2607 None.

2608 **OPERANDS**

2609 None.

2610 **STDIN**

2611 None.

2612 **INPUT FILES**

2613 None.

2614 **ENVIRONMENT VARIABLES**

2615 None.

2616 **ASYNCHRONOUS EVENTS**

2617 None.

2618 **STDOUT**

2619 None.

2620 **STDERR**

2621 None.

2622 **OUTPUT FILES**

2623 None.

2624 **EXTENDED DESCRIPTION**

2625 None.

2626 **EXIT STATUS**

2627 0 Successful completion.

2628 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.2629 **CONSEQUENCES OF ERRORS**

2630 None.

2631 **APPLICATION USAGE**

2632 None.

2633 **EXAMPLES**

```
2634       for i in * do
2635           if test -d "$i" then break fi done
```

2636 **RATIONALE**

2637 In early proposals, consideration was given to expanding the syntax of *break* and *continue* to refer
2638 to a label associated with the appropriate loop as a preferable alternative to the *n* method.
2639 However, this volume of IEEE Std. 1003.1-200x does reserve the namespace of command names
2640 ending with a colon. It is anticipated that a future implementation could take advantage of this
2641 and provide something like:

```
2642       outofloop: for i in a b c d e
2643           do
2644               for j in 0 1 2 3 4 5 6 7 8 9
2645               do
2646                   if test -r "${i}${j}"
2647                   then break outofloop
2648                   fi
2649               done
2650           done
```

2651 and that this might be standardized after implementation experience is achieved.

2652 **FUTURE DIRECTIONS**

2653 None.

2654 **SEE ALSO**

2655 Section 2.15 (on page 2276)

2656 **CHANGE HISTORY**

2657 None.

2658 **NAME**
2659 colon — null utility

2660 **SYNOPSIS**
2661 : [*argument* ...]

2662 **DESCRIPTION**
2663 This utility shall only expand command *arguments*. It is used when a command is needed, as in
2664 the *then* condition of an **if** command, but nothing is to be done by the command.

2665 **OPTIONS**
2666 None.

2667 **OPERANDS**
2668 None.

2669 **STDIN**
2670 None.

2671 **INPUT FILES**
2672 None.

2673 **ENVIRONMENT VARIABLES**
2674 None.

2675 **ASYNCHRONOUS EVENTS**
2676 None.

2677 **STDOUT**
2678 None.

2679 **STDERR**
2680 None.

2681 **OUTPUT FILES**
2682 None.

2683 **EXTENDED DESCRIPTION**
2684 None.

2685 **EXIT STATUS**
2686 Zero.

2687 **CONSEQUENCES OF ERRORS**
2688 None.

2689 **APPLICATION USAGE**
2690 None.

2691 **EXAMPLES**
2692 : \${X=abc}
2693 if false
2694 then :
2695 else echo \$X
2696 fi
2697 **abc**

2698 As with any of the special built-ins, the null utility can also have variable assignments and
2699 redirections associated with it, such as:

2700 `x=y : > z`

2701 which sets variable `x` to the value `y` (so that it persists after the null utility completes) and creates
2702 or truncates file `z`.

2703 **RATIONALE**

2704 None.

2705 **FUTURE DIRECTIONS**

2706 None.

2707 **SEE ALSO**

2708 Section 2.15 (on page 2276)

2709 **CHANGE HISTORY**

2710 None.

2711 **NAME**
2712 continue — continue for, while, or until loop

2713 **SYNOPSIS**
2714 continue [*n*]

2715 **DESCRIPTION**
2716 The *continue* utility shall return to the top of the smallest enclosing **for**, **while**, or **until** loop, or to
2717 the top of the *n*th enclosing loop, if *n* is specified. This involves repeating the condition list of a
2718 **while** or **until** loop or performing the next assignment of a **for** loop, and reexecuting the loop if
2719 appropriate.
2720 The value of *n* is a decimal integer greater than or equal to 1. The default is equivalent to *n*=1. If
2721 *n* is greater than the number of enclosing loops, the last enclosing loop shall be used.

2722 **OPTIONS**
2723 None.

2724 **OPERANDS**
2725 None.

2726 **STDIN**
2727 None.

2728 **INPUT FILES**
2729 None.

2730 **ENVIRONMENT VARIABLES**
2731 None.

2732 **ASYNCHRONOUS EVENTS**
2733 None.

2734 **STDOUT**
2735 None.

2736 **STDERR**
2737 None.

2738 **OUTPUT FILES**
2739 None.

2740 **EXTENDED DESCRIPTION**
2741 None.

2742 **EXIT STATUS**
2743 0 Successful completion.
2744 >0 The *n* value was not an unsigned decimal integer greater than or equal to 1.

2745 **CONSEQUENCES OF ERRORS**
2746 None.

2747 **APPLICATION USAGE**

2748 None.

2749 **EXAMPLES**

```
2750       for i in *
2751       do
2752           if test -d "$i"
2753           then continue
2754           fi
2755           echo "\"$i\" \" is not a directory.
2756       done
```

2757 **RATIONALE**

2758 None.

2759 **FUTURE DIRECTIONS**

2760 None.

2761 **SEE ALSO**

2762 Section 2.15 (on page 2276)

2763 **CHANGE HISTORY**

2764 None.

2765 **NAME**
2766 dot — execute commands in current environment

2767 **SYNOPSIS**
2768 . *file*

2769 **DESCRIPTION**
2770 The shell shall execute commands from the *file* in the current environment.
2771 If *file* does not contain a slash, the shell shall use the search path specified by *PATH* to find the
2772 directory containing *file*. Unlike normal command search, however, the file searched for by the
2773 *dot* utility need not be executable. If no readable file is found, a non-interactive shell shall abort;
2774 an interactive shell shall write a diagnostic message to standard error, but this condition shall
2775 not be considered a syntax error.

2776 **OPTIONS**
2777 None.

2778 **OPERANDS**
2779 None.

2780 **STDIN**
2781 None.

2782 **INPUT FILES**
2783 None.

2784 **ENVIRONMENT VARIABLES**
2785 None.

2786 **ASYNCHRONOUS EVENTS**
2787 None.

2788 **STDOUT**
2789 None.

2790 **STDERR**
2791 None.

2792 **OUTPUT FILES**
2793 None.

2794 **EXTENDED DESCRIPTION**
2795 None.

2796 **EXIT STATUS**
2797 Returns the value of the last command executed, or a zero exit status if no command is executed.

2798 **CONSEQUENCES OF ERRORS**
2799 None.

2800 **APPLICATION USAGE**
2801 None.

2802 **EXAMPLES**
2803 cat foobar
2804 **foo=hello bar=world**
2805 . foobar
2806 echo \$foo \$bar
2807 **hello world**

2808 RATIONALE

2809 Some older implementations searched the current directory for the *file*, even if the value of *PATH*
2810 disallowed it. This behavior was omitted from this volume of IEEE Std. 1003.1-200x due to
2811 concerns about introducing the susceptibility to trojan horses that the user might be trying to
2812 avoid by leaving *dot* out of *PATH*.

2813 The KornShell version of *dot* takes optional arguments that are set to the positional parameters.
2814 This is a valid extension that allows a *dot* script to behave identically to a function.

2815 FUTURE DIRECTIONS

2816 None.

2817 SEE ALSO

2818 Section 2.15 (on page 2276)

2819 CHANGE HISTORY

2820 None.

2821 **NAME**
2822 eval — construct command by concatenating arguments

2823 **SYNOPSIS**
2824 eval [*argument* ...]

2825 **DESCRIPTION**
2826 The *eval* utility shall construct a command by concatenating *arguments* together, separating each
2827 with a <space> character. The constructed command shall be read and executed by the shell.

2828 **OPTIONS**
2829 None.

2830 **OPERANDS**
2831 None.

2832 **STDIN**
2833 None.

2834 **INPUT FILES**
2835 None.

2836 **ENVIRONMENT VARIABLES**
2837 None.

2838 **ASYNCHRONOUS EVENTS**
2839 None.

2840 **STDOUT**
2841 None.

2842 **STDERR**
2843 None.

2844 **OUTPUT FILES**
2845 None.

2846 **EXTENDED DESCRIPTION**
2847 None.

2848 **EXIT STATUS**
2849 If there are no *arguments*, or only null arguments, *eval* shall return a zero exit status; otherwise, it
2850 shall return the exit status of the command defined by the string of concatenated *arguments*
2851 separated by spaces.

2852 **CONSEQUENCES OF ERRORS**
2853 None.

2854 **APPLICATION USAGE**
2855 None.

2856 **EXAMPLES**
2857 foo=10 x=foo
2858 y=' '\$x
2859 echo \$y
2860 **\$foo**
2861 eval y=' '\$x
2862 echo \$y
2863 **10**

2864 **RATIONALE**

2865 None.

2866 **FUTURE DIRECTIONS**

2867 None.

2868 **SEE ALSO**

2869 Section 2.15 (on page 2276)

2870 **CHANGE HISTORY**

2871 None.

2872 **NAME**

2873 exec — execute commands and open, close, or copy file descriptors

2874 **SYNOPSIS**

2875 exec [*command* [*argument* ...]]

2876 **DESCRIPTION**

2877 The *exec* utility shall open, close, and/or copy file descriptors as specified by any redirections as
2878 part of the command.

2879 If *exec* is specified without *command* or *arguments*, and any file descriptors with numbers greater
2880 than 2 are opened with associated redirection statements, it is unspecified whether those file
2881 descriptors remain open when the shell invokes another utility. Scripts concerned that child
2882 shells could misuse open file descriptors can always close them explicitly, as shown in one of the
2883 following examples.

2884 If *exec* is specified with *command*, it shall replace the shell with *command* without creating a new
2885 process. If *arguments* are specified, they shall be arguments to *command*. Redirection affects the
2886 current shell execution environment.

2887 **OPTIONS**

2888 None.

2889 **OPERANDS**

2890 None.

2891 **STDIN**

2892 None.

2893 **INPUT FILES**

2894 None.

2895 **ENVIRONMENT VARIABLES**

2896 None.

2897 **ASYNCHRONOUS EVENTS**

2898 None.

2899 **STDOUT**

2900 None.

2901 **STDERR**

2902 None.

2903 **OUTPUT FILES**

2904 None.

2905 **EXTENDED DESCRIPTION**

2906 None.

2907 **EXIT STATUS**

2908 If *command* is specified, *exec* shall not return to the shell; rather, the exit status of the process shall
2909 be the exit status of the program implementing *command*, which overlaid the shell. If *command* is
2910 not found, the exit status shall be 127. If *command* is found, but it is not an executable utility, the
2911 exit status shall be 126. If a redirection error occurs (see Section 2.8.1 (on page 2255)), the shell
2912 shall exit with a value in the range 1–125. Otherwise, *exec* shall return a zero exit status.

2913 **CONSEQUENCES OF ERRORS**

2914 None.

2915 **APPLICATION USAGE**

2916 None.

2917 **EXAMPLES**2918 Open *readfile* as file descriptor 3 for reading:2919 `exec 3< readfile`2920 Open *writefile* as file descriptor 4 for writing:2921 `exec 4> writefile`

2922 Make file descriptor 5 a copy of file descriptor 0:

2923 `exec 5<&0`

2924 Close file descriptor 3:

2925 `exec 3<&-`2926 Cat the file **maggie** by replacing the current shell with the *cat* utility:2927 `exec cat maggie`2928 **RATIONALE**

2929 Most historical implementations were not conformant in that:

2930 `foo=bar exec cmd`2931 did not pass **foo** to **cmd**.2932 **FUTURE DIRECTIONS**

2933 None.

2934 **SEE ALSO**

2935 Section 2.15 (on page 2276)

2936 **CHANGE HISTORY**

2937 None.

2938 **NAME**
2939 exit — cause the shell to exit

2940 **SYNOPSIS**
2941 exit [*n*]

2942 **DESCRIPTION**
2943 The *exit* utility shall cause the shell to exit with the exit status specified by the unsigned decimal
2944 integer *n*. If *n* is specified, but its value is not between 0 and 255 inclusively, the exit status is
2945 undefined.

2946 A *trap* on **EXIT** shall be executed before the shell terminates, except when the *exit* utility is
2947 invoked in that *trap* itself, in which case the shell shall exit immediately.

2948 **OPTIONS**
2949 None.

2950 **OPERANDS**
2951 None.

2952 **STDIN**
2953 None.

2954 **INPUT FILES**
2955 None.

2956 **ENVIRONMENT VARIABLES**
2957 None.

2958 **ASYNCHRONOUS EVENTS**
2959 None.

2960 **STDOUT**
2961 None.

2962 **STDERR**
2963 None.

2964 **OUTPUT FILES**
2965 None.

2966 **EXTENDED DESCRIPTION**
2967 None.

2968 **EXIT STATUS**
2969 The exit status shall be *n*, if specified. Otherwise, the value shall be the exit value of the last
2970 command executed, or zero if no command was executed. When *exit* is executed in a *trap* action,
2971 the last command is considered to be the command that executed immediately preceding the
2972 *trap* action.

2973 **CONSEQUENCES OF ERRORS**
2974 None.

2975 **APPLICATION USAGE**
2976 None.

2977 **EXAMPLES**
2978 Exit with a *true* value:
2979 exit 0

2980 Exit with a *false* value:

2981 `exit 1`

2982 **RATIONALE**

2983 As explained in other sections, certain exit status values have been reserved for special uses and
2984 should be used by applications only for those purposes:

2985 126 A file to be executed was found, but it was not an executable utility.

2986 127 A utility to be executed was not found.

2987 >128 A command was interrupted by a signal.

2988 **FUTURE DIRECTIONS**

2989 None.

2990 **SEE ALSO**

2991 Section 2.15 (on page 2276)

2992 **CHANGE HISTORY**

2993 None.

2994 **NAME**

2995 export — set export attribute for variables

2996 **SYNOPSIS**2997 export name[=*word*]. . .

2998 export -p

2999 **DESCRIPTION**3000 The shell shall give the export attribute to the variables corresponding to the specified *names*,
3001 which shall cause them to be in the environment of subsequently executed commands.3002 The *export* special built-in shall support the Base Definitions volume of IEEE Std. 1003.1-200x,
3003 Section 12.2, Utility Syntax Guidelines.3004 When **-p** is specified, *export* shall write to the standard output the names and values of all
3005 exported variables, in the following format:

3006 "export %s=%s\n", <name>, <value>

3007 The shell shall format the output, including the proper use of quoting, so that it is suitable for
3008 reinput to the shell as commands that achieve the same exporting results.

3009 When no arguments are given, the results are unspecified.

3010 **OPTIONS**

3011 None.

3012 **OPERANDS**

3013 None.

3014 **STDIN**

3015 None.

3016 **INPUT FILES**

3017 None.

3018 **ENVIRONMENT VARIABLES**

3019 None.

3020 **ASYNCHRONOUS EVENTS**

3021 None.

3022 **STDOUT**

3023 None.

3024 **STDERR**

3025 None.

3026 **OUTPUT FILES**

3027 None.

3028 **EXTENDED DESCRIPTION**

3029 None.

3030 **EXIT STATUS**

3031 Zero.

3032 **CONSEQUENCES OF ERRORS**

3033 None.

3034 **APPLICATION USAGE**

3035 None.

3036 **EXAMPLES**3037 Export *PWD* and *HOME* variables:3038 `export PWD HOME`3039 Set and export the *PATH* variable:3040 `export PATH=/local/bin:$PATH`

3041 Save and restore all exported variables:

3042 `export -p > temp-file`3043 `unset a lot of variables`3044 `... processing`3045 `. temp-file`3046 **RATIONALE**

3047 Some historical shells use the no-argument case as the functional equivalent of what is required
3048 here with **-p**. This feature was left unspecified because it is not historical practice in all shells,
3049 and some scripts may rely on the now-unspecified results on their implementations. Attempts to
3050 specify the **-p** output as the default case were unsuccessful in achieving consensus. The **-p**
3051 option was added to allow portable access to the values that can be saved and then later restored
3052 using; for example, a *dot* script.

3053 **FUTURE DIRECTIONS**

3054 None.

3055 **SEE ALSO**

3056 Section 2.15 (on page 2276)

3057 **CHANGE HISTORY**

3058 None.

3059 **NAME**

3060 readonly — set read-only attribute for variables

3061 **SYNOPSIS**3062 readonly name[=*word*]...

3063 readonly -p

3064 **DESCRIPTION**

3065 The variables whose *names* are specified shall be given the *readonly* attribute. The values of
3066 variables with the *readonly* attribute cannot be changed by subsequent assignment, nor can those
3067 variables be unset by the *unset* utility.

3068 The *readonly* special built-in shall support the Base Definitions volume of IEEE Std. 1003.1-200x,
3069 Section 12.2, Utility Syntax Guidelines.

3070 When **-p** is specified, *readonly* writes to the standard output the names and values of all read-
3071 only variables, in the following format:

3072 "readonly %s=%s\n", <name>, <value>

3073 The shell shall format the output, including the proper use of quoting, so that it is suitable for
3074 reinput to the shell as commands that achieve the same attribute-setting results.

3075 When no arguments are given, the results are unspecified.

3076 **OPTIONS**

3077 None.

3078 **OPERANDS**

3079 None.

3080 **STDIN**

3081 None.

3082 **INPUT FILES**

3083 None.

3084 **ENVIRONMENT VARIABLES**

3085 None.

3086 **ASYNCHRONOUS EVENTS**

3087 None.

3088 **STDOUT**

3089 None.

3090 **STDERR**

3091 None.

3092 **OUTPUT FILES**

3093 None.

3094 **EXTENDED DESCRIPTION**

3095 None.

3096 **EXIT STATUS**

3097 Zero.

3098 **CONSEQUENCES OF ERRORS**

3099 None.

3100 **APPLICATION USAGE**

3101 None.

3102 **EXAMPLES**3103 `readonly HOME PWD`3104 **RATIONALE**

3105 Some historical shells preserve the read-only attribute across separate invocations. This volume
3106 of IEEE Std. 1003.1-200x allows this behavior, but does not require it.

3107 The `-p` option allows portable access to the values that can be saved and then later restored
3108 using; for example, a *dot* script. Also see the RATIONALE for *export* (on page 2291) for a
3109 description of the no-argument and `-p` output cases and a related example.

3110 Read-only functions were considered, but they were omitted as not being historical practice or
3111 particularly useful. Furthermore, functions must not be *readonly* across invocations to preclude
3112 *spoofing* (spoofing is the term for the practice of creating a program that acts like a well-known
3113 utility with the intent of subverting the real intent of the user) of administrative or security-
3114 relevant (or security-conscious) shell scripts.

3115 **FUTURE DIRECTIONS**

3116 None.

3117 **SEE ALSO**

3118 Section 2.15 (on page 2276)

3119 **CHANGE HISTORY**

3120 None.

3121 **NAME**

3122 return — return from a function

3123 **SYNOPSIS**3124 return [*n*]3125 **DESCRIPTION**3126 The *return* utility shall cause the shell to stop executing the current function or *dot* script. If the
3127 shell is not currently executing a function or *dot* script, the results are unspecified.3128 **OPTIONS**

3129 None.

3130 **OPERANDS**

3131 None.

3132 **STDIN**

3133 None.

3134 **INPUT FILES**

3135 None.

3136 **ENVIRONMENT VARIABLES**

3137 None.

3138 **ASYNCHRONOUS EVENTS**

3139 None.

3140 **STDOUT**

3141 None.

3142 **STDERR**

3143 None.

3144 **OUTPUT FILES**

3145 None.

3146 **EXTENDED DESCRIPTION**

3147 None.

3148 **EXIT STATUS**3149 The value of the special parameter '?' shall be set to *n*, an unsigned decimal integer, or to the
3150 exit status of the last command executed if *n* is not specified. If the value of *n* is greater than 255,
3151 the results are undefined. When *return* is executed in a *trap* action, the last command is
3152 considered to be the command that executed immediately preceding the *trap* action.3153 **CONSEQUENCES OF ERRORS**

3154 None.

3155 **APPLICATION USAGE**

3156 None.

3157 **EXAMPLES**

3158 None.

3159 **RATIONALE**3160 The behavior of *return* when not in a function or *dot* script differs between the System V shell
3161 and the KornShell. In the System V shell this is an error, whereas in the KornShell, the effect is
3162 the same as *exit*.

3163 The results of returning a number greater than 255 are undefined because of differing practices
3164 in the various historical implementations. Some shells AND out all but the low-order 8 bits;
3165 others allow larger values, but not of unlimited size.

3166 See the discussion of appropriate exit status values under *exit* (on page 2289).

3167 **FUTURE DIRECTIONS**

3168 None.

3169 **SEE ALSO**

3170 Section 2.15 (on page 2276)

3171 **CHANGE HISTORY**

3172 None.

3173 **NAME**

3174 set — set or unset options and positional parameters

3175 **SYNOPSIS**3176 xSI set [-abCefmnuvx] [-h] [-o *option*] [*argument...*]3177 xSI set [+abCefmnuvx] [+h] [+o *option*] [*argument...*]3178 set — [*argument...*]

3179 set -o

3180 set +o

3181 **DESCRIPTION**

3182 If no options or *arguments* are specified, *set* shall write the names and values of all shell variables
 3183 in the collation sequence of the current locale. Each *name* shall start on a separate line, using the
 3184 format:

3185 "%s=%s\n", <*name*>, <*value*>

3186 The *value* string shall be written with appropriate quoting so that it is suitable for reinput to the
 3187 shell, setting or resetting, as far as possible, the variables that are currently set. Read-only
 3188 variables cannot be reset; see the description of shell quoting in Section 2.2 (on page 2236).

3189 When options are specified, they shall set or unset attributes of the shell, as described below.
 3190 When *arguments* are specified, they cause positional parameters to be set or unset, as described
 3191 below. Setting or unsetting attributes and positional parameters are not necessarily related
 3192 actions, but they can be combined in a single invocation of *set*.

3193 The *set* special built-in shall support the Base Definitions volume of IEEE Std. 1003.1-200x,
 3194 Section 12.2, Utility Syntax Guidelines except that options can be specified with either a leading
 3195 hyphen (meaning enable the option) or plus sign (meaning disable it).

3196 Implementations shall support the options in the following list in both their hyphen and plus-
 3197 sign forms. These options can also be specified as options to *sh*.

3198 **-a** When this option is on, the export attribute shall be set for each variable to which an
 3199 assignment is performed; see the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 3200 4.16, Variable Assignment. If the assignment precedes a utility name in a command, the
 3201 export attribute shall not persist in the current execution environment after the utility
 3202 completes, with the exception that preceding one of the special built-in utilities causes the
 3203 export attribute to persist after the built-in has completed. If the assignment does not
 3204 precede a utility name in the command, or if the assignment is a result of the operation of
 3205 the *getopts* or *read* utilities, the export attribute shall persist until the variable is unset.

3206 **-b** This option is supported if the system supports the User Portability Utilities option. It shall
 3207 cause the shell to notify the user asynchronously of background job completions. The
 3208 following message is written to standard error:

3209 "[%d]%c %s%s\n", <*job-number*>, <*current*>, <*status*>, <*job-name*>

3210 where the fields shall be as follows:

3211 <*current*> The character '+' identifies the job that would be used as a default for
 3212 the *fg* or *bg* utilities; this job can also be specified using the *job_id* "%+" or
 3213 "%%". The character '-' identifies the job that would become the default
 3214 if the current default job were to exit; this job can also be specified using
 3215 the *job_id* "%-". For other jobs, this field is a <space> character. At most
 3216 one job can be identified with '+' and at most one job can be identified

- 3217 with '-'. If there is any suspended job, then the current job shall be a
 3218 suspended job. If there are at least two suspended jobs, then the previous
 3219 job also shall be a suspended job.
- 3220 <job-number> A number that can be used to identify the process group to the *wait*, *fg*, *bg*,
 3221 and *kill* utilities. Using these utilities, the job can be identified by
 3222 prefixing the job number with '%'.
 3223 <status> Unspecified.
 3224 <job-name> Unspecified.
- 3225 When the shell notifies the user a job has been completed, it may remove the job's process
 3226 ID from the list of those known in the current shell execution environment; see Section
 3227 2.9.3.1 (on page 2259). Asynchronous notification shall not be enabled by default.
- 3228 -C (Uppercase C.) Prevent existing files from being overwritten by the shell's '>' redirection
 3229 operator (see Section 2.7.2 (on page 2252)); the ">|" redirection operator shall override this
 3230 *noclobber* option for an individual file.
- 3231 -e When this option is on, if a simple command fails for any of the reasons listed in Section
 3232 2.8.1 (on page 2255) or returns an exit status value >0, and is not part of the compound list
 3233 following a **while**, **until**, or **if** keyword, and is not a part of an AND or OR list, and is not a
 3234 pipeline preceded by the ! reserved word, then the shell shall immediately exit.
- 3235 -f The shell shall disable path name expansion.
- 3236 XSI -h Locate and remember utilities invoked by functions as those functions are defined (the
 3237 utilities are normally located when the function is executed).
- 3238 -m This option is supported if the system supports the User Portability Utilities option. All jobs
 3239 shall be run in their own process groups. Immediately before the shell issues a prompt after
 3240 completion of the background job, a message reporting the exit status of the background job
 3241 shall be written to standard error. If a foreground job stops, the shell shall write a message
 3242 to standard error to that effect, formatted as described by the *jobs* utility. In addition, if a job
 3243 changes status other than exiting (for example, if it stops for input or output or is stopped
 3244 by a SIGSTOP signal), the shell shall write a similar message immediately prior to writing
 3245 the next prompt. This option is enabled by default for interactive shells.
- 3246 -n The shell shall read commands but does not execute them; this can be used to check for
 3247 shell script syntax errors. An interactive shell may ignore this option.
- 3248 -o Write the current settings of the options to standard output in an unspecified format.
- 3249 +o Write the current option settings to standard output in a format that is suitable for reinput
 3250 to the shell as commands that achieve the same options settings.
- 3251 -o *option*
 3252 This option is supported if the system supports the User Portability Utilities option. It shall
 3253 set various options, many of which shall be equivalent to the single option letters. The
 3254 following values of *option* shall be supported:
- 3255 *allexport* Equivalent to -a.
 3256 *errexit* Equivalent to -e.
 3257 *ignoreeof* Prevent an interactive shell from exiting on end-of-file. This setting prevents
 3258 accidental logouts when <control>-D is entered. A user shall explicitly *exit* to
 3259 leave the interactive shell.

3260	<i>monitor</i>	Equivalent to -m . This option is supported if the system supports the User Portability Utilities option.
3261		
3262	<i>noclobber</i>	Equivalent to -C (uppercase C).
3263	<i>noglob</i>	Equivalent to -f .
3264	<i>noexec</i>	Equivalent to -n .
3265	<i>nolog</i>	Prevent the entry of function definitions into the command history; see
3266		Command History List (on page 3064).
3267	<i>notify</i>	Equivalent to -b .
3268	<i>nounset</i>	Equivalent to -u .
3269	<i>verbose</i>	Equivalent to -v .
3270	<i>vi</i>	Allow shell command line editing using the built-in <i>vi</i> editor. Enabling <i>vi</i>
3271		mode shall disable any other command line editing mode provided as an
3272		implementation extension.
3273		It need not be possible to set <i>vi</i> mode on for certain block-mode terminals.
3274	<i>xtrace</i>	Equivalent to -x .
3275	-u	The shell writes a message to standard error when it tries to expand a variable that is not set
3276		and immediately exit. An interactive shell shall not exit.
3277	-v	The shell writes its input to standard error as it is read.
3278	-x	The shell writes to standard error a trace for each command after it expands the command
3279		and before it executes it. It is unspecified whether the command that turns tracing off is
3280		traced.
3281		The default for all these options is off (unset) unless the shell was invoked with them on; see <i>sh</i> .
3282		The remaining arguments shall be assigned in order to the positional parameters. The special
3283		parameter ' # ' shall be set to reflect the number of positional parameters. All positional
3284		parameters shall be unset before any new values are assigned.
3285		The special argument " --- " immediately following the <i>set</i> command name can be used to delimit
3286		the arguments if the first argument begins with ' + ' or ' - ', or to prevent inadvertent listing of
3287		all shell variables when there are no arguments. The command <i>set--</i> without <i>argument</i> shall
3288		unset all positional parameters and set the special parameter ' # ' to zero.
3289	OPTIONS	
3290		None.
3291	OPERANDS	
3292		None.
3293	STDIN	
3294		None.
3295	INPUT FILES	
3296		None.
3297	ENVIRONMENT VARIABLES	
3298		None.

3299 **ASYNCHRONOUS EVENTS**

3300 None.

3301 **STDOUT**

3302 None.

3303 **STDERR**

3304 None.

3305 **OUTPUT FILES**

3306 None.

3307 **EXTENDED DESCRIPTION**

3308 None.

3309 **EXIT STATUS**

3310 Zero.

3311 **CONSEQUENCES OF ERRORS**

3312 None.

3313 **APPLICATION USAGE**

3314 None.

3315 **EXAMPLES**

3316 Write out all variables and their values:

3317 `set`

3318 Set \$1, \$2, and \$3 and set "\$#" to 3:

3319 `set c a b`3320 Turn on the `-x` and `-v` options:3321 `set -xv`

3322 Unset all positional parameters:

3323 `set --`3324 Set \$1 to the value of `-x`, even if `x` begins with `'-'` or `'+'`:3325 `set -- "$x"`3326 Set the positional parameters to the expansion of `x`, even if `x` expands with a leading `'-'` or `'+'`:3327 `set -- $x`3328 **RATIONALE**

3329 The `set --` form is listed specifically in the SYNOPSIS even though this usage is implied by the
3330 Utility Syntax Guidelines. The explanation of this feature removes any ambiguity about whether
3331 the `set --` form might be misinterpreted as being equivalent to `set` without any options or
3332 arguments. The functionality of this form has been adopted from the KornShell. In System V, `set`
3333 `--` only unsets parameters if there is at least one argument; the only way to unset all parameters
3334 is to use *shift*. Using the KornShell version should not affect System V scripts because there
3335 should be no reason to issue it without arguments deliberately; if it were issued as, for example:

3336 `set -- "$@"`

3337 and there were in fact no arguments resulting from `"$@"`, unsetting the parameters would have
3338 no result.

3339 The *set* + form in early proposals was omitted as being an unnecessary duplication of *set* alone
3340 and not widespread historical practice.

3341 The *noclobber* option was changed to allow *set -C* as well as the *set -o noclobber* option. The
3342 single-letter version was added so that the historical "\$-" paradigm would not be broken; see
3343 Section 2.5.2 (on page 2241).

3344 The *-h* flag is related to command name hashing and is only required on XSI-conformant
3345 systems.

3346 The following *set* flags were omitted intentionally with the following rationale:

3347 **-k** The *-k* flag was originally added by the author of the Bourne shell to make it easier for
3348 users of pre-release versions of the shell. In early versions of the Bourne shell the construct
3349 *set name=value*, had to be used to assign values to shell variables. The problem with *-k* is
3350 that the behavior affects parsing, virtually precluding writing any compilers. To explain the
3351 behavior of *-k*, it is necessary to describe the parsing algorithm, which is implementation-
3352 defined. For example:

```
3353     set -k; echo name=value
```

3354 and:

```
3355     set x--k  
3356     echo name=value
```

3357 behave differently. The interaction with functions is even more complex. What is more, the
3358 *-k* flag is never needed, since the command line could have been reordered.

3359 **-t** The *-t* flag is hard to specify and almost never used. The only known use could be done
3360 with here-documents. Moreover, the behavior with *ksh* and *sh* differs. The reference page
3361 says that it exits after reading and executing one command. What is one command? If the
3362 input is *date;date*, *sh* executes both *date* commands while *ksh* does only the first.

3363 Consideration was given to rewriting *set* to simplify its confusing syntax. A specific suggestion
3364 was that the *unset* utility should be used to unset options instead of using the non-*getopt()*-able
3365 *+option* syntax. However, the conclusion was reached that the historical practice of using *+option*
3366 was satisfactory and that there was no compelling reason to modify such widespread historical
3367 practice.

3368 The *-o* option was adopted from the KornShell to address user needs. In addition to its generally
3369 friendly interface, *-o* is needed to provide the *vi* command line editing mode, for which
3370 historical practice yields no single-letter option name. (Although it might have been possible to
3371 invent such a letter, it was recognized that other editing modes would be developed and *-o*
3372 provides ample name space for describing such extensions.)

3373 Historical implementations are inconsistent in the format used for *-o* option status reporting.
3374 The *+o* format without an option-argument was added to allow portable access to the options
3375 that can be saved and then later restored using, for instance, a dot script.

3376 Historically, *sh* did trace the command *set +x*, but *ksh* did not.

3377 The *ignoreeof* setting prevents accidental logouts when the end-of-file character (typically
3378 <control>-D) is entered. A user shall explicitly *exit* to leave the interactive shell.

3379 The *set -m* option was added to apply only to the UPE because it applies primarily to interactive
3380 use, not shell script applications.

3381 The ability to do asynchronous notification became available in the 1988 version of the
3382 KornShell. To have it occur, the user had to issue the command:

```
3383     trap "jobs -n" CLD
```

3384 The C shell provides two different levels of an asynchronous notification capability. The
3385 environment variable *notify* is analogous to what is done in *set -b* or *set -o notify*. When set, it
3386 notifies the user immediately of background job completions. When unset, this capability is
3387 turned off.

3388 The other notification ability comes through the built-in utility *notify*. The syntax is:

```
3389     notify [%job ... ]
```

3390 By issuing *notify* with no operands, it causes the C shell to notify the user asynchronously when
3391 the state of the current job changes. If given operands, *notify* asynchronously informs the user of
3392 changes in the states of the specified jobs.

3393 To add asynchronous notification to the POSIX shell, neither the KornShell extensions to *trap*,
3394 nor the C shell *notify* environment variable seemed appropriate (*notify* is not a proper POSIX
3395 environment variable name).

3396 The *set -b* option was selected as a compromise.

3397 The *notify* built-in was considered to have more functionality than was required for simple
3398 asynchronous notification.

3399 **FUTURE DIRECTIONS**

3400 None.

3401 **SEE ALSO**

3402 Section 2.15 (on page 2276)

3403 **CHANGE HISTORY**

3404 **Issue 6**

3405 The obsolescent *set* command name followed by *'-'* has been removed.

3406 The following new requirements on POSIX implementations derive from alignment with the
3407 Single UNIX Specification:

- 3408 • The *nolog* option is added to *set -o*.

3409 **NAME**

3410 shift — shift positional parameters

3411 **SYNOPSIS**3412 shift [*n*]3413 **DESCRIPTION**

3414 The positional parameters shall be shifted. Positional parameter 1 shall be assigned the value of
3415 parameter (1+*n*), parameter 2 shall be assigned the value of parameter (2+*n*), and so on. The
3416 parameters represented by the numbers "\$#" down to "\$#-*n*+1" shall be unset, and the
3417 parameter '#' is updated to reflect the new number of positional parameters.

3418 The value *n* shall be an unsigned decimal integer less than or equal to the value of the special
3419 parameter '#'. If *n* is not given, it shall be assumed to be 1. If *n* is 0, the positional and special
3420 parameters are not changed.

3421 **OPTIONS**

3422 None.

3423 **OPERANDS**

3424 None.

3425 **STDIN**

3426 None.

3427 **INPUT FILES**

3428 None.

3429 **ENVIRONMENT VARIABLES**

3430 None.

3431 **ASYNCHRONOUS EVENTS**

3432 None.

3433 **STDOUT**

3434 None.

3435 **STDERR**

3436 None.

3437 **OUTPUT FILES**

3438 None.

3439 **EXTENDED DESCRIPTION**

3440 None.

3441 **EXIT STATUS**3442 The exit status is >0 if *n*>\$#; otherwise, it is zero.3443 **CONSEQUENCES OF ERRORS**

3444 None.

3445 **APPLICATION USAGE**

3446 None.

3447 **EXAMPLES**

3448 \$ set a b c d e

3449 \$ shift 2

3450 \$ echo \$*

3451 c d e

3452 **RATIONALE**

3453 None.

3454 **FUTURE DIRECTIONS**

3455 None.

3456 **SEE ALSO**

3457 Section 2.15 (on page 2276)

3458 **CHANGE HISTORY**

3459 None.

3460 **NAME**

3461 times — write process times

3462 **SYNOPSIS**

3463 times

3464 **DESCRIPTION**3465 Write the accumulated user and system times for the shell and for all of its child processes, in the
3466 following POSIX locale format:

```
3467     "%dm%fs %dm%fs\n%dm%fs %dm%fs\n", <shell user minutes>,  
3468     <shell user seconds>, <shell system minutes>,  
3469     <shell system seconds>, <children user minutes>,  
3470     <children user seconds>, <children system minutes>,  
3471     <children system seconds>
```

3472 The four pairs of times correspond to the members of the `<sys/times.h>` `tms` structure (defined
3473 in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 13, Headers) as returned by
3474 `times()`: `tms_utime`, `tms_stime`, `tms_cutime`, and `tms_cstime`, respectively.3475 **OPTIONS**

3476 None.

3477 **OPERANDS**

3478 None.

3479 **STDIN**

3480 None.

3481 **INPUT FILES**

3482 None.

3483 **ENVIRONMENT VARIABLES**

3484 None.

3485 **ASYNCHRONOUS EVENTS**

3486 None.

3487 **STDOUT**

3488 None.

3489 **STDERR**

3490 None.

3491 **OUTPUT FILES**

3492 None.

3493 **EXTENDED DESCRIPTION**

3494 None.

3495 **EXIT STATUS**

3496 Zero.

3497 **CONSEQUENCES OF ERRORS**

3498 None.

3499 **APPLICATION USAGE**

3500 None.

3501 **EXAMPLES**3502 `$ times`3503 `0m0.43s 0m1.11s`3504 `8m44.18s 1m43.23s`3505 **RATIONALE**3506 The *times* special built-in from the Single UNIX Specification is now required for all conforming
3507 shells.3508 **FUTURE DIRECTIONS**

3509 None.

3510 **SEE ALSO**

3511 Section 2.15 (on page 2276)

3512 **CHANGE HISTORY**

3513 None.

3514 **NAME**

3515 trap — trap signals

3516 **SYNOPSIS**3517 trap [*action condition ...*]3518 **DESCRIPTION**

3519 If *action* is '-', the shell shall reset each *condition* to the default value. If *action* is null (" "), the
 3520 shell shall ignore each specified *condition* if it arises. Otherwise, the argument *action* shall be read
 3521 and executed by the shell when one of the corresponding conditions arises. The action of *trap*
 3522 shall override a previous action (either default action or one explicitly set). The value of "\$?"
 3523 after the *trap* action completes shall be the value it had before *trap* was invoked.

3524 The condition can be EXIT, 0 (equivalent to EXIT), or a signal specified using a symbolic name,
 3525 without the SIG prefix, as listed in the tables of signal names in the <signal.h> header defined in
 3526 the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 13, Headers; for example, HUP,
 3527 INT, QUIT, TERM. Implementations may permit lowercase signal names or names with the SIG
 3528 prefix as an extension. Setting a trap for SIGKILL or SIGSTOP produces undefined results.

3529 The environment in which the shell executes a *trap* on EXIT shall be identical to the environment
 3530 immediately after the last command executed before the *trap* on EXIT was taken.

3531 Each time *trap* is invoked, the *action* argument shall be processed in a manner equivalent to:

3532 eval "\$action"

3533 Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, although
 3534 no error need be reported when attempting to do so. An interactive shell may reset or catch
 3535 signals ignored on entry. Traps shall remain in place for a given shell until explicitly changed
 3536 with another *trap* command.

3537 When a subshell is entered, traps that are not being ignored are set to the default actions. This
 3538 does not imply that the *trap* command cannot be used within the subshell to set new traps.

3539 The *trap* command with no arguments shall write to standard output a list of commands
 3540 associated with each condition. The format shall be:

3541 "trap — %s %s ... \n", <action>, <condition> ...

3542 The shell shall format the output, including the proper use of quoting, so that it is suitable for
 3543 reinput to the shell as commands that achieve the same trapping results. For example:

3544 save_traps=\$(trap)

3545 ...

3546 eval "\$save_traps"

3547 XSI the following signal names:
 3548

3549

3550

3551 XSI

3552 XSI

3553 XSI

3554 XSI

3555 XSI

3556 XSI

3557 XSI

Signal Number	Signal Name
1	SIGHUP
2	SIGINT
3	SIGQUIT
6	SIGABRT
9	SIGKILL
14	SIGALRM
15	SIGTERM

3558

3559

The *trap* special built-in shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

3560 **OPTIONS**

3561 None.

3562 **OPERANDS**

3563 None.

3564 **STDIN**

3565 None.

3566 **INPUT FILES**

3567 None.

3568 **ENVIRONMENT VARIABLES**

3569 None.

3570 **ASYNCHRONOUS EVENTS**

3571 None.

3572 **STDOUT**

3573 None.

3574 **STDERR**

3575 None.

3576 **OUTPUT FILES**

3577 None.

3578 **EXTENDED DESCRIPTION**

3579 None.

3580 **EXIT STATUS**

3581 XSI If the trap name or number is invalid, a non-zero exit status shall be returned; otherwise, zero

3582 XSI shall be returned. For both interactive and non-interactive shells, invalid signal names or

3583 numbers shall not be considered a syntax error and do not cause the shell to abort.

3584 **CONSEQUENCES OF ERRORS**

3585 None.

3586 **APPLICATION USAGE**

3587 None.

3588 **EXAMPLES**

3589 Write out a list of all traps and actions:

3590 trap

3591 Set a trap so the *logout* utility in the directory referred to by the *HOME* environment variable

3592 executes when the shell terminates:

3593 `trap '$HOME/logout' EXIT`

3594 `or:`

3595 `trap '$HOME/logout' 0`

3596 Unset traps on INT, QUIT, TERM, and EXIT:

3597 `trap - INT QUIT TERM EXIT`

3598 **RATIONALE**

3599 Implementations may permit lowercase signal names as an extension. Implementations may
3600 also accept the names with the SIG prefix; no known historical shell does so. The *trap* and *kill*
3601 utilities in this volume of IEEE Std. 1003.1-200x are now consistent in their omission of the SIG
3602 prefix for signal names. Some *kill* implementations do not allow the prefix, and *kill -l* lists the
3603 signals without prefixes.

3604 Trapping SIGKILL or SIGSTOP is syntactically accepted by some historical implementations, but
3605 it has no effect. Portable POSIX applications cannot attempt to trap these signals.

3606 The output format is not historical practice. Since the output of historical *trap* commands is not
3607 portable (because numeric signal values are not portable) and had to change to become so, an
3608 opportunity was taken to format the output in a way that a shell script could use to save and
3609 then later reuse a trap if it wanted.

3610 The KornShell uses an **ERR** trap that is triggered whenever *set -e* would cause an exit. This is
3611 allowable as an extension, but was not mandated, as other shells have not used it.

3612 The text about the environment for the EXIT trap invalidates the behavior of some historical
3613 versions of interactive shells which, for example, close the standard input before executing a
3614 trap on 0. For example, in some historical interactive shell sessions the following trap on 0 would
3615 always print "—":

3616 `trap 'read foo; echo "$foo—" ' 0`

3617 **FUTURE DIRECTIONS**

3618 None.

3619 **SEE ALSO**

3620 Section 2.15 (on page 2276)

3621 **CHANGE HISTORY**

3622 **Issue 6**

3623 XSI-conforming implementations provide the mapping of signal names to numbers given above
3624 (previously this had been marked obsolescent). Other implementations need not provide this
3625 optional mapping.

3626 **NAME**

3627 unset — unset values and attributes of variables and functions

3628 **SYNOPSIS**

3629 unset [-fv] *name* ...

3630 **DESCRIPTION**

3631 Each variable or function specified by *name* shall be unset.

3632 If **-v** is specified, *name* refers to a variable name and the shell shall unset it and remove it from
3633 the environment. Read-only variables cannot be unset.

3634 If **-f** is specified, *name* refers to a function and the shell shall unset the function definition.

3635 If neither **-f** nor **-v** is specified, *name* refers to a variable; if a variable by that name does not
3636 exist, it is unspecified whether a function by that name, if any, shall be unset.

3637 Unsetting a variable or function that was not previously set shall not be considered an error and
3638 does not cause the shell to abort.

3639 The *unset* special built-in shall support the Base Definitions volume of IEEE Std. 1003.1-200x,
3640 Section 12.2, Utility Syntax Guidelines.

3641 Note that:

3642 VARIABLE=

3643 is not equivalent to an *unset* of **VARIABLE**; in the example, **VARIABLE** is set to " ". Also, the
3644 variables that can be *unset* should not be misinterpreted to include the special parameters (see
3645 Section 2.5.2 (on page 2241)).

3646 **OPTIONS**

3647 None.

3648 **OPERANDS**

3649 None.

3650 **STDIN**

3651 None.

3652 **INPUT FILES**

3653 None.

3654 **ENVIRONMENT VARIABLES**

3655 None.

3656 **ASYNCHRONOUS EVENTS**

3657 None.

3658 **STDOUT**

3659 None.

3660 **STDERR**

3661 None.

3662 **OUTPUT FILES**

3663 None.

3664 **EXTENDED DESCRIPTION**

3665 None.

3666 **EXIT STATUS**3667 0 All *name* operands were successfully unset.3668 >0 At least one *name* could not be unset.3669 **CONSEQUENCES OF ERRORS**

3670 None.

3671 **APPLICATION USAGE**

3672 None.

3673 **EXAMPLES**3674 Unset *VISUAL* variable:

3675 unset -v VISUAL

3676 Unset the functions **foo** and **bar**:

3677 unset -f foo bar

3678 **RATIONALE**3679 Consideration was given to omitting the **-f** option in favor of an *unfunction* utility, but the
3680 standard developers decided to retain historical practice.3681 The **-v** option was introduced because System V historically used one name space for both
3682 variables and functions. When *unset* is used without options, System V historically unset either a
3683 function or a variable, and there was no confusion about which one was intended. A portable
3684 POSIX application can use *unset* without an option to unset a variable, but not a function; the **-f**
3685 option must be used.3686 **FUTURE DIRECTIONS**

3687 None.

3688 **SEE ALSO**

3689 Section 2.15 (on page 2276)

3690 **CHANGE HISTORY**

3691 None.

Batch Environment Services

3693

3694 BE This chapter describes the services and utilities that shall be implemented on all systems that
3695 claim conformance to the Batch Environment option. This functionality is dependent on support
3696 of this option (and the rest of this section is not further shaded for this option).

3697 3.1 General Concepts

3698 3.1.1 Batch Client-Server Interaction

3699 Batch jobs are created and managed by batch servers. A batch client interacts with a batch server
3700 to access batch services on behalf of the user. In order to use batch services, a user must have
3701 access to a batch client.

3702 A batch server is a computational entity, such as a daemon process, that provides batch services.
3703 Batch servers route, queue, modify, and execute batch jobs on behalf of batch clients.

3704 The batch utilities described in this volume of IEEE Std. 1003.1-200x (and listed in Table 3-1 (on
3705 page 2314)) are clients of batch services; they allow users to perform actions on the job such as
3706 creating, modifying, and deleting batch jobs from a shell command line. Although these batch
3707 utilities may be said to accomplish certain services, they actually obtain services on behalf of a
3708 user by means of requests to batch servers.

3709

Table 3-1 Batch Utilities

3710

qalter *qmove* *qrls* *qstat*

3711

qdel *qmsg* *qselect* *qsub*

3712

qhold *qrerun* *qsig*

3713

Client-server interaction takes place by means of the batch requests defined in this chapter. Because direct access to batch jobs and queues is limited to batch servers, clients and servers of different implementations can interoperate, since dependencies on private structures for batch jobs and queues are limited to batch servers. Also, batch servers may be clients of other batch servers.

3714

3715

3716

3717

3718 **3.1.2 Batch Queues**

3719

Two types of batch queue are described: *routing queues* and *execution queues*. When a batch job is placed in a routing queue, it is a candidate for routing. A batch job is removed from routing queues under the following conditions:

3720

3721

3722

- The batch job has been routed to another queue.

3723

- The batch job has been deleted from the batch queue.

3724

- The batch job has been aborted.

3725

When a batch job is placed in an execution queue, it is a candidate for execution.

3726

A batch job is removed from an execution queue under the following conditions:

3727

- The batch job has been executed and exited.

3728

- The batch job has been aborted.

3729

- The batch job has been deleted from the batch queue.

3730

- The batch job has been moved to another queue.

3731

Access to a batch queue is limited to the batch server that manages the batch queue. Clients never access a batch queue or a batch job directly, either to read or write information; all client access to batch queues or jobs takes place through batch servers.

3732

3733

3734 **3.1.3 Batch Job Creation**

3735

When a batch server creates a batch job on behalf of a client, it assigns a batch job identifier to the job. A batch job identifier consists of both a sequence number that is unique among the sequence numbers issued by that server and the name of the server. Since the batch server name is unique within a name space, the job identifier is likewise unique within the name space.

3736

3737

3738

3739

The batch server that creates a batch job returns the batch server-assigned job identifier to the client that requested the job creation. If the batch server routes or moves the job to another server, it sends the job identifier with the job. Once assigned, the job identifier of a batch job never changes.

3740

3741

3742

3743 3.1.4 Batch Job Tracking

3744 Since a batch job may be moved after creation, the batch server name component of the job
3745 identifier does not always indicate the location of the job. An implementation may provide a
3746 batch job tracking mechanism, in which case the user generally does not need to know the
3747 location of the job. However, an implementation is not required to provide a batch job tracking
3748 mechanism, in which case the user must find routed jobs by probing the possible destinations.

3749 3.1.5 Batch Job Routing

3750 To route a batch job, a batch server either moves the job to some other queue that is managed by
3751 the batch server, or requests that some other batch server accept the job.

3752 Each routing queue has one or more queues to which it can route batch jobs. The batch server
3753 administrator creates routing queues.

3754 A batch server may route a batch job from a routing queue to another routing queue. Batch
3755 servers shall prevent or otherwise handle cases of circular routing paths. As a deferred service, a
3756 batch server routes jobs from the routing queues that it manages. The algorithm by which a
3757 batch server selects a batch queue to which to route a batch job is implementation-defined.

3758 A batch job need not be eligible for routing to all the batch queues fed by the routing queue from
3759 which it is routed. A batch server that has been asked to accept the job may reject the request if
3760 the job requires resources that are unavailable to that batch server, or if the client is not
3761 authorized to access the batch server.

3762 Batch servers may route high-priority jobs before low-priority jobs, but, on other than
3763 overloaded systems, the effect may be imperceptible to the user. If all the batch servers fed by a
3764 routing queue reject requests to accept the job for reasons that are permanent, the batch server
3765 that manages the job aborts the job. If all or some rejections are temporary, the batch server
3766 should try to route the job again at some later point.

3767 The conformance document for an implementation shall list the reasons for rejecting the routing
3768 of a batch job. The conformance document shall indicate the reasons for which the routing
3769 should be retried later and the reasons for which the job should be aborted.

3770 3.1.6 Batch Job Execution

3771 To execute a batch job is to create a session leader (a process) that runs the shell program
3772 indicated by the *Shell_Path* attribute of the job. The script is passed to the program as its
3773 standard input. An implementation of the batch server may pass the script to the program by
3774 other means. The implementation shall document the alternate means in the conformance
3775 document. At the time a batch job begins execution, it is defined to enter the RUNNING state.
3776 The primary program that is executed by a batch job is typically, though not necessarily, a shell
3777 program.

3778 A batch server executes eligible jobs as a deferred service—no client request is necessary once
3779 the batch job is created and eligible. However, the attributes of a batch job, such as the job hold
3780 type, may render the job ineligible. A batch server scans the execution queues that it manages for
3781 jobs that are eligible for execution. The algorithm by which the batch server selects eligible jobs
3782 for execution is implementation-defined.

3783 As part of creating the process for the batch job, the batch server opens the standard output and
3784 standard error streams of the session.

3785 The attributes of a batch job may indicate that the batch server that executes the job is to send
3786 mail to a list of users at the time it begins execution of the job.

3787 3.1.7 Batch Job Exit

3788 When the session leader of an executing job terminates, the job exits. As part of exiting a batch
3789 job, the batch server that manages the job shall remove the job from the batch queue in which it
3790 resides. The server shall transfer output files of the job to a location described by the attributes of
3791 the job.

3792 The attributes of a batch job may indicate that the batch server that manages the job should send
3793 mail to a list of users at the time the job exits.

3794 3.1.8 Batch Job Abort

3795 A batch server aborts jobs for which a required deferred service cannot be performed. The
3796 attributes of a batch job may indicate that the batch server that aborts the job shall send mail to a
3797 list of users at the time it aborts the job.

3798 3.1.9 Batch Authorization

3799 In order to access batch services, a user must have execute access to a batch client. For example,
3800 to use the command language interface defined in this section, the user must be able to execute
3801 the programs that embody those utilities.

3802 Clients, such as the batch environment utilities (marked BE), access batch services by means of
3803 requests to one or more batch servers. To acquire the services of any given batch server, the user
3804 identifier under which the client runs must be authorized to use that batch server.

3805 The user with an associated user name that creates a batch job owns the job and can perform
3806 actions such as read, modify, delete, and move.

3807 A user identifier of the same value at a different host need not be the same user. For example,
3808 user name *smith* at host **alpha** may or may not represent the same person as user name *smith* at
3809 host **beta**. Likewise, the same person may have access to different user names on different hosts.

3810 An implementation may optionally provide an authorization mechanism that permits one user
3811 name to access jobs under another user name.

3812 A process on a client host may be authorized to run processes under multiple user names at a
3813 batch server host. Where appropriate, the utilities defined in this volume of
3814 IEEE Std. 1003.1-200x provide a means for a user to choose from among such user names when
3815 creating or modifying a batch job.

3816 3.1.10 Batch Administration

3817 The processing of a batch job by a batch server is affected by the attributes of the job. The
3818 processing of a batch job may also be affected by the attributes of the batch queue in which the
3819 job resides and by the status of the batch server that manages the job.

3820 A batch administrator is a user that is authorized to modify all the attributes of queues and jobs
3821 and to change the status of a batch server. A batch operator is a user that is authorized to modify
3822 some, but not all, of the attributes of jobs and queues, and may change the status of the batch
3823 server.

3824 **3.1.11 Batch Notification**

3825 Whereas batch servers are persistent entities, clients are often transient. For example, the *qsub*
 3826 utility creates a batch job and exits. For this reason, batch servers notify users of batch job events
 3827 by sending mail to the user that owns the job, or to other designated users.

3828 **3.2 Batch Services**

3829 The presence of Batch Environment option services is indicated by the configuration variable
 3830 POSIX2_PBS. A conforming batch server provides services as defined in this section.

3831 A batch server provides batch services in two ways:

- 3832 1. The batch server provides a service at the request of a client.
- 3833 2. The batch server provides a deferred service as a result of a change in conditions
 3834 monitored by the batch server.

3835 If a batch server cannot complete a request, it rejects the request. If a batch server cannot
 3836 complete a deferred service for a batch job, the batch server aborts the batch job. Table 3-2 is a
 3837 summary of environment variables that shall be supported by an implementation of the batch
 3838 server and utilities.

3839 **Table 3-2 Environment Variable Summary**

Variable	Description
PBS_DPREFIX	Defines the directive prefix (see <i>qsub</i>)
PBS_ENVIRONMENT	Batch Job is batch or interactive (see Section 3.2.2.1 (on page 2319))
PBS_JOBID	The <i>job_identifier</i> attribute of job (see Section 3.2.3.8 (on page 2331))
PBS_JOBNAME	The <i>job_name</i> attribute of job (see Section 3.2.3.8 (on page 2331))
PBS_O_HOME	Defines the <i>HOME</i> of the batch client (see <i>qsub</i>)
PBS_O_HOST	Defines the host name of the batch client (see <i>qsub</i>)
PBS_O_LANG	Defines the <i>LANG</i> of the batch client (see <i>qsub</i>)
PBS_O_LOGNAME	Defines the <i>LOGNAME</i> of the batch client (see <i>qsub</i>)
PBS_O_MAIL	Defines the <i>MAIL</i> of the batch client (see <i>qsub</i>)
PBS_O_PATH	Defines the <i>PATH</i> of the batch client (see <i>qsub</i>)
PBS_O_QUEUE	Defines the submit queue of the batch client (see <i>qsub</i>)
PBS_O_SHELL	Defines the <i>SHELL</i> of the batch client (see <i>qsub</i>)
PBS_O_TZ	Defines the <i>TZ</i> of the batch client (see <i>qsub</i>)
PBS_O_WORKDIR	Defines the working directory of the batch client (see <i>qsub</i>)
PBS_QUEUE	Defines the initial execution queue (see Section 3.2.2.1 (on page 2319))

3859 **3.2.1 Batch Job States**

3860 A batch job is always in one of several states: QUEUED, RUNNING, HELD, WAITING,
3861 EXITING, or TRANSITING. The state of a batch job determines the types of requests that the
3862 batch server that manages the batch job can accept for the batch job. A batch server changes the
3863 state of a batch job either in response to service requests from clients or as a result of deferred
3864 services, such as job execution or job routing.

3865 A batch job that is in the QUEUED state resides in a queue but is still pending either execution or
3866 routing, depending on the queue type.

3867 A batch server that queues a batch job in a routing queue shall put the batch job in the QUEUED
3868 state. A batch server that puts a batch job in an execution queue, but has not yet executed the
3869 batch job, shall put the batch job in the QUEUED state. A batch job that resides in an execution
3870 queue and is executing is defined to be in the RUNNING state. While a batch job is in the
3871 RUNNING state, a session leader is associated with the batch job.

3872 A batch job that resides in an execution queue, but is ineligible to run because of a hold attribute,
3873 is defined to be in the HELD state.

3874 A batch job that is not held, but must wait until a future date and time before executing, is
3875 defined to be in the WAITING state.

3876 When the session leader associated with a running job exits, the batch job shall be placed in the
3877 EXITING state.

3878 A batch job for which the session leader has terminated is defined to be in the EXITING state,
3879 and the batch server that manages such a batch job cannot accept job modification requests that
3880 affect the batch job. While a batch job is in the EXITING state, the batch server that manages the
3881 batch job is staging output files and notifying clients of job completion. Once a batch job has
3882 exited, it no longer exists as an object managed by a batch server.

3883 A batch job that is being moved from a routing queue to another queue is defined to be in the
3884 TRANSITING state.

3885 When a batch job in a routing queue has been selected to be moved to a new destination, then
3886 the batch job is in either the QUEUED state or the TRANSITING state, depending on the batch
3887 server implementation.

3888 Batch jobs with either a *Execution_Time* attribute value set in the future or a *Hold_Types* attribute
3889 of value not equal to NO_HOLD, or both, may be routed or held in the routing queue. An
3890 implementation shall document the treatment of jobs with the *Execution_Time* or *Hold_Types*
3891 attributes in a routing queue.

3892 When a batch job in a routing queue has not been selected to be moved to a new destination and
3893 the batch job has a *Hold_Types* attribute value of other than NO_HOLD, then the job should be in
3894 the HELD state.

3895 **Note:** The effect of a hold upon a batch job in a routing queue is implementation-defined. |
3896 The implementation should use the state that matches whether the batch job can |
3897 route with a hold or not.

3898 When a batch job in a routing queue has not been selected to be moved to a new destination and
3899 the batch job has:

- 3900 • A *Hold_Types* attribute value of NO_HOLD
- 3901 • An *Execution_Time* attribute in the past

3902 then the batch job shall be in the QUEUED state.

3903 When a batch job in a routing queue has not been selected to be moved to a new destination and
 3904 the batch job has:

- 3905 • A *Hold_Types* attribute value of NO_HOLD
- 3906 • A *Execution_Time* attribute in the future

3907 then the batch job may be in the WAITING state.

3908 **Note:** The effect of a future execution time upon a batch job in a routing queue is
 3909 implementation-defined. The implementation should use the state that matches
 3910 whether the batch job can route with a hold or not.

3911 Table 3-3 describes the next state of a batch job, given the current state of the batch job and the
 3912 type of request. Table 3-4 (on page 2321) describes the response of a batch server to a request,
 3913 given the current state of the batch job and the type of request.

3914 **3.2.2 Deferred Batch Services**

3915 This section describes the deferred services performed by batch servers: job execution, job
 3916 routing, job exit, job abort, and the rerunning of jobs after a restart.

3917 **3.2.2.1 Batch Job Execution**

3918 To execute a batch job is to create a session leader (a process) that runs the shell program
 3919 indicated by the *Shell_Path_List* attribute of the batch job. The script is passed to the program as
 3920 its standard input. An implementation of the batch server may pass the script to the program by
 3921 other means. The implementation shall document the alternate means in the conformance
 3922 document. At the time a batch job begins execution, it is defined to enter the RUNNING state.

3923 **Table 3-3** Next State Table

Request Type	Current State						
	X	Q	R	H	W	E	T
Queue Batch Job Request	Q	e	e	e	e	e	e
Modify Batch Job Request	e	Q	R	H	W	e	T
Delete Batch Job Request	e	X	E	X	X	E	X
Batch Job Message Request	e	Q	R	H	W	E	T
Rerun Batch Job Request	e	e	Q	e	e	e	e
Signal Batch Job Request	e	e	R	H	W	e	e
Batch Job Status Request	e	Q	R	H	W	E	T
Batch Queue Status Request	X	Q	R	H	W	E	T
Server Status Request	X	Q	R	H	W	E	T
Select Batch Jobs Request	X	Q	R	H	W	E	T
Move Batch Job Request	e	Q	R	H	W	e	T
Hold Batch Job Request	e	H	R/H	H	H	e	T
Release Batch Job Request	Q	R	Q/W/H	W	e	T	
Server Shutdown Request	X	Q	Q	H	W	E	T
Locate Batch Job Request	e	Q	R	H	W	E	T

3941 **Legend**

3942 X Nonexistent

3943 Q QUEUED

3944 R RUNNING

3945 H HELD

3946 W WAITING

3947 E EXITING

3948 T TRANSITING

3949 e Error

3950 A batch server that has an execution queue containing jobs is said to own the queue and manage
3951 the batch jobs in that queue. A batch server that has been started shall execute the batch jobs in
3952 the execution queues owned by the batch server. The batch server shall schedule for execution
3953 those jobs in the execution queues that are in the QUEUED state. The algorithm for scheduling
3954 jobs is implementation-defined.

3955 A batch server that executes a batch job shall create, in the environment of the session leader of
3956 the batch job, an environment variable named *PBS_ENVIRONMENT*, the value of which is the
3957 string *PBS_BATCH* encoded in the portable character set.

3958 A batch server that executes a batch job shall create, in the environment of the session leader of
3959 the batch job, an environment variable named *PBS_QUEUE*, the value of which is the name of
3960 the execution queue of the batch job encoded in the portable character set.

3961 To rerun a batch job is to requeue a batch job that is currently executing and then kill the session
3962 leader of the executing job by sending a SIGKILL prior to completion; see Section 3.2.3.11 (on
3963 page 2333). A batch server that reruns a batch job shall append the standard output and
3964 standard error files of the batch job to the corresponding files of the previous execution, if they
3965 exist, with appropriate annotation. If either file does not exist, that file shall be created as in
3966 normal execution.

3967

Table 3-4 Results/Output Table

3968

3969

3970

3971

3972

3973

3974

3975

3976

3977

3978

3979

3980

3981

3982

3983

3984

Request Type	Current State						
	X	Q	R	H	W	E	T
Queue Batch Job Request	O	e	e	e	e	e	e
Modify Batch Job Request	e	O	e	O	O	e	e
Delete Batch Job Request	e	O	O	O	O	e	O
Batch Job Message Request	e	e	O	e	e	e	e
Rerun Batch Job Request	e	e	O	e	e	e	e
Signal Batch Job Request	e	e	O	e	e	e	e
Batch Job Status Request	e	O	O	O	O	O	O
Batch Queue Status Request	O	O	O	O	O	O	O
Server Status Request	O	O	O	O	O	O	O
Select Batch Job Request	e	O	O	O	O	O	O
Move Batch Job Request	e	O	O	O	O	e	e
Hold Batch Job Request	e	O	O	O	O	e	e
Release Batch Job Request	e	O	e	O	O	e	e
Server Shutdown Request	O	O	e	O	O	e	e
Locate Batch Job Request	e	O	O	O	O	O	O

3985

Legend

3986

O OK

3987

e Error message

3988

The execution of a batch job by a batch server is controlled by job, queue, and server attributes, as defined in this section.

3989

3990

Account_Name Attribute

3991

Batch accounting is an optional feature of batch servers. If a batch server implements accounting, the statements in this section apply and the configuration variable `POSIX2_PBS_ACCOUNTING` shall be set to 1.

3992

3993

3994

A batch server that executes a batch job shall charge the account named in the *Account_Name* attribute of the batch job for resources consumed by the batch job.

3995

3996

If the *Account_Name* attribute of the batch job is absent from the batch job attribute list or is altered while the batch job is in execution, the batch server action is implementation-defined.

3997

3998

Checkpoint Attribute

3999

Batch checkpointing is an optional feature of batch servers. If a batch server implements checkpointing, the statements in this section apply and the configuration variable `POSIX2_PBS_CHECKPOINT` shall be set to 1.

4000

4001

4002

There are two attributes associated with the checkpointing feature: *Checkpoint* and *Minimum_Cpu_Interval*. *Checkpoint* is a batch job attribute, while *Minimum_Cpu_Interval* is a queue attribute. An implementation that does not support checkpointing shall support the *Checkpoint* job attribute to the extent that the batch server shall maintain and pass this attribute to other servers.

4003

4004

4005

4006

4007

The behavior of a batch server that executes a batch job for which the value of the *Checkpoint* attribute is `CHECKPOINT_UNSPECIFIED` is implementation-defined. The implementation shall document the behavior of the batch server. A batch server that executes a batch job for which the

4008

4009

- 4010 value of the *Checkpoint* attribute is NO_CHECKPOINT shall not checkpoint the batch job.
- 4011 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is
4012 CHECKPOINT_AT_SHUTDOWN shall checkpoint the batch job only when the batch server
4013 accepts a request to shut down during the time when the batch job is in the RUNNING state.
- 4014 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is
4015 CHECKPOINT_AT_MIN_CPU_INTERVAL shall checkpoint the batch job at the interval
4016 specified by the *Minimum_Cpu_Interval* attribute of the queue for which the batch job has been
4017 selected. The *Minimum_Cpu_Interval* attribute shall be specified in units of CPU minutes.
- 4018 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is an
4019 unsigned integer shall checkpoint the batch job at an interval that is the value of either the
4020 *Checkpoint* attribute, or the *Minimum_Cpu_Interval* attribute of the queue for which the batch job
4021 has been selected, whichever is greater. Both intervals shall be in units of CPU minutes. When
4022 the *Minimum_Cpu_Interval* attribute is greater than the *Checkpoint* attribute, the batch job shall
4023 write a warning message to the standard error stream of the batch job.
- 4024 **Error_Path Attribute**
- 4025 The *Error_Path* attribute of a running job cannot be changed by a *Modify Batch Job Request*. When
4026 the *Join_Path* attribute of the batch job is set to the value FALSE and the *Keep_Files* attribute of
4027 the batch job does not contain the value KEEP_STD_ERROR, a batch server that executes a batch
4028 job shall perform one of the following actions:
- 4029 • Set the standard error stream of the session leader of the batch job to the path described by
4030 the value of the *Error_Path* attribute of the batch job.
 - 4031 • Buffer the standard error of the session leader of the batch job until completion of the batch
4032 job, and when the batch job exits return the contents to the destination described by the value
4033 of the *Error_Path* attribute of the batch job. Where the batch server buffers standard error is
4034 implementation-defined.
- 4035 Applications shall not rely on having access to the standard error of a batch job prior to the
4036 completion of the batch job.
- 4037 When the *Error_Path* attribute does not specify a host name, then the batch server shall retain the
4038 standard error of the batch job on the host of execution.
- 4039 When the *Error_Path* attribute does specify a host name and the *Keep_Files* attribute does not
4040 contain the value KEEP_STD_ERROR, then the final destination of the standard error of the
4041 batch job shall be on the host whose host name is specified.
- 4042 If the path indicated by the value of the *Error_Path* attribute of the batch job is a relative path, the
4043 batch server shall expand the path relative to the home directory of the user on the host to which
4044 the file is being returned.
- 4045 When the batch server buffers the standard error of the batch job and the file cannot be opened
4046 for write upon completion of the batch job, then the server shall place the standard error in an
4047 implementation-defined location and notify the user of the location via mail. It shall be possible
4048 for the user to process this mail using the *mailx* utility.
- 4049 If a batch server that does not buffer the standard error cannot open the standard error path of
4050 the batch job for write access, then the batch server shall abort the batch job.

4051 **Execution_Time Attribute**

4052 A batch server shall not execute a batch job before the time represented by the value of the
 4053 *Execution_Time* attribute of the batch job. The *Execution_Time* attribute is defined in seconds since
 4054 the Epoch.

4055 **Hold_Types Attribute**

4056 A batch server shall support the following hold types:

4057 **s** Can be set or released by a user with at least a privilege level of batch administrator
 4058 (SYSTEM).

4059 **o** Can be set or released by a user with at least a privilege level of batch operator
 4060 (OPERATOR).

4061 **u** Can be set or released by the user with at least a privilege level of user, where the user is
 4062 defined in the *Job_Owner* attribute (USER).

4063 **n** Indicates that none of the *Hold_Types* attributes are set (NO_HOLD).

4064 An implementation may define other hold types. The conformance document for an
 4065 implementation shall describe any additional hold types, how they are specified, their internal
 4066 representation, their behavior, and how they affect the behavior of other utilities.

4067 The value of the *Hold_Types* attribute shall be the union of the valid hold types (**ss**, **oo**, **uu**, and
 4068 any implementation-defined hold types), or **nn**.

4069 A batch server shall not execute a batch job if the *Hold_Types* attribute of the batch job has a
 4070 value other than NO_HOLD. If the *Hold_Types* attribute of the batch job has a value other than
 4071 NO_HOLD, the batch job shall be in the HELD state.

4072 **Job_Owner Attribute**

4073 The *Job_Owner* attribute consists of a pair of user name and host name values of the form:

4074 `username@hostname`

4075 A batch server that accepts a *Queue Batch Job Request* shall set the *Job_Owner* attribute to a string
 4076 that is the *username@hostname* of the user who submitted the job.

4077 **Join_Path Attribute**

4078 A batch server that executes a batch job for which the value of the *Join_Path* attribute is TRUE
 4079 shall ignore the value of the *Error_Path* attribute and merge the standard error of the batch job
 4080 with the standard output of the batch job.

4081 **Keep_Files Attribute**

4082 A batch server that executes a batch job for which the value of the *Keep_Files* attribute includes
 4083 the value KEEP_STD_OUTPUT shall retain the standard output of the batch job on the host
 4084 where execution occurs. The standard output shall be retained in the home directory of the user
 4085 under whose user ID the batch job is executed and the file name shall be the default file name for
 4086 the standard output as defined under the **-o** option of the *qsub* utility. The *Output_Path* attribute
 4087 is not modified.

4088 A batch server that executes a batch job for which the value of the *Keep_Files* attribute includes
 4089 the value KEEP_STD_ERROR shall retain the standard error of the batch job on the host where
 4090 execution occurs. The standard error shall be retained in the home directory of the user under
 4091 whose user ID the batch job is executed and the file name shall be the default file name for

4092 standard error as defined under the `-e` option of the `qsub` utility. The `Error_Path` attribute is not
4093 modified.

4094 A batch server that executes a batch job for which the value of the `Keep_Files` attribute includes
4095 values other than `KEEP_STD_OUTPUT` and `KEEP_STD_ERROR` shall retain these other files on
4096 the host where execution occurs. These files shall be retained in the home directory of the user
4097 under whose user identifier the batch job is executed and the file names shall be the default file
4098 names for the files as defined in the conformance document for the implementation.

4099 **Mail_Points and Mail_Users Attributes**

4100 A batch server that executes a batch job for which one of the values of the `Mail_Points` attribute is
4101 the value `MAIL_AT_BEGINNING` shall send a mail message to each user account listed in the
4102 `Mail_Users` attribute of the batch job.

4103 The mail message shall contain at least the batch job identifier, queue, and server at which the
4104 batch job currently resides, and the `Job_Owner` attribute.

4105 **Output_Path Attribute**

4106 The `Output_Path` attribute of a running job cannot be changed by a *Modify Batch Job Request*.
4107 When the `Keep_Files` attribute of the batch job does not contain the value `KEEP_STD_OUTPUT`, a
4108 batch server that executes a batch job shall either:

4109 • Set the standard output stream of the session leader of the batch job to the destination
4110 described by the value of the `Output_Path` attribute of the batch job.

4111 or:

4112 • Buffer the standard output of the session leader of the batch job until completion of the batch
4113 job, and when the batch job exits return the contents to the destination described by the value
4114 of the `Output_Path` attribute of the batch job.

4115 When the `Output_Path` attribute does not specify a host name, then the batch server shall retain
4116 the standard output of the batch job on the host of execution.

4117 When the `Keep_Files` attribute does not contain the value `KEEP_STD_OUTPUT` and the
4118 `Output_Path` attribute does specify a host name, then the final destination of the standard output
4119 of the batch job shall be on the host specified.

4120 If the path specified in the `Output_Path` attribute of the batch job is a relative path, the batch
4121 server shall expand the path relative to the home directory of the user on the host to which the
4122 file is being returned.

4123 Whether or not the batch server buffers the standard output of the batch job until completion of
4124 the batch job is implementation-defined. Applications shall not rely on having access to the
4125 standard output of a batch job prior to the completion of the batch job.

4126 When the batch server does buffer the standard output of the batch job and the file cannot be
4127 opened for write upon completion of the batch job, then the batch server shall place the standard
4128 output in an implementation-defined location and notify the user of the location via mail. It shall
4129 be possible for the user to process this mail using the `mailx` utility.

4130 If a batch server that does not buffer the standard output cannot open the standard output path
4131 of the batch job for write access, then the batch server shall abort the batch job.

4132 Priority Attribute

4133 A batch server implementation may choose to preferentially execute a batch job based on the
4134 *Priority* attribute. The interpretation of the batch job *Priority* attribute by a batch server is
4135 implementation-defined. If an implementation uses the *Priority* attribute, it shall interpret larger
4136 values of the *Priority* attribute to mean the batch job shall be preferentially selected for execution.

4137 Rerunable Attribute

4138 A batch job that began execution but did not complete, because the batch server either shut
4139 down or terminated abnormally, shall be requeued if the *Rerunable* attribute of the batch job has
4140 the value TRUE.

4141 If a batch job, which was requeued after beginning execution but prior to completion, has a valid
4142 checkpoint file and the batch server supports checkpointing, then the batch job shall be restarted
4143 from the last valid checkpoint.

4144 If the batch job cannot be restarted from a checkpoint, then when a batch job has a *Rerunable*
4145 attribute value of TRUE and was requeued after beginning execution but prior to completion,
4146 the batch server shall place the batch job into execution at the beginning of the job.

4147 When a batch job has a *Rerunable* attribute value other than TRUE and was requeued after
4148 beginning execution but prior to completion, and the batch job cannot be restarted from a
4149 checkpoint, then the batch server shall abort the batch job.

4150 Resource_List Attribute

4151 A batch server that executes a batch job shall establish the resource limits of the session leader of
4152 the batch job according to the values of the *Resource_List* attribute of the batch job. Resource
4153 limits shall be enforced by an implementation-defined method.

4154 Shell_Path_List Attribute

4155 The *Shell_Path_List* job attribute consists of a list of pairs of path name and host name values.
4156 The host name component can be omitted, in which case the path name serves as the default
4157 path name when a batch server cannot find the name of the host on which it is running in the
4158 list.

4159 A batch server that executes a batch job shall select, from the value of the *Shell_Path_List*
4160 attribute of the batch job, a path name where the shell to execute the batch job shall be found.
4161 The batch server shall select the path name, in order of preference, according to the following
4162 methods:

- 4163 • Select the path name that contains the name of the host on which the batch server is running.
- 4164 • Select the path name for which the host name has been omitted.
- 4165 • Select the path name for the login shell of the user under which the batch job is to execute.

4166 If the shell path value selected is an invalid path name, the batch server shall abort the batch job.

4167 If the value of the selected path name from the *Shell_Path_List* attribute of the batch job
4168 represents a partial path, the batch server shall expand the path relative to a path that is
4169 implementation-defined.

4170 The batch server that executes the batch job shall execute the program that was selected from the
4171 *Shell_Path_List* attribute of the batch job. The batch server shall pass the path to the script of the
4172 batch job as the first argument to the shell program.

4173 **User_List Attribute**

4174 The *User_List* job attribute consists of a list of pairs of user name and host name values. The host
4175 name component can be omitted, in which case the user name serves as a default when a batch
4176 server cannot find the name of the host on which it is running in the list.

4177 A batch server that executes a batch job shall select, from the value of the *User_List* attribute of
4178 the batch job, a user name under which to create the session leader. The server shall select the
4179 user name, in order of preference, according to the following methods:

- 4180 • Select the user name of a value that contains the name of the host on which the batch server
4181 executes.
- 4182 • Select the user name of a value for which the host name has been omitted.
- 4183 • Select the user name from the *Job_Owner* attribute of the batch job.

4184 **Variable_List Attribute**

4185 A batch server that executes a batch job shall create, in the environment of the session leader of
4186 the batch job, each environment variable listed in the *Variable_List* attribute of the batch job, and
4187 set the value of each such environment variable to that of the corresponding variable in the
4188 variable list.

4189 **3.2.2.2 Batch Job Routing**

4190 To route a batch job is to select a queue from a list and move the batch job to that queue.

4191 A batch server that has routing queues, which have been started, shall route the jobs in the
4192 routing queues owned by the batch server. A batch server is allowed to delay the routing of a
4193 batch job. The algorithm for selecting a batch job and the queue to which it will be routed is
4194 implementation-defined.

4195 When a routing queue has multiple possible destinations specified, then the precedence of the
4196 destination is implementation-defined.

4197 A batch server that routes a batch job to a queue at another server shall move the batch job into
4198 the target queue with a *Queue Batch Job Request*.

4199 If the target server rejects the *Queue Batch Job Request*, the routing server shall retry routing the
4200 batch job or abort the batch job. A batch server that retries failed routings shall provide a means
4201 for the batch administrator to specify the number of retries and the minimum period of time
4202 between retries. The means by which an administrator specifies the number of retries and the
4203 delay between retries is implementation-defined. When the number of retries specified by the
4204 batch administrator has been exhausted, the batch server shall abort the batch job and perform
4205 the functions of *Batch Job Exit*; see Section 3.2.2.3.

4206 **3.2.2.3 Batch Job Exit**

4207 For each job in the EXITING state, the batch server that exited the batch job shall perform the
4208 following deferred services in the order specified:

- 4209 1. If buffering standard error, move that file into the location specified by the *Error_Path*
4210 attribute of the batch job.
- 4211 2. If buffering standard output, move that file into the location specified by the *Output_Path*
4212 attribute of the batch job.
- 4213 3. If the *Mail_Points* attribute of the batch job includes MAIL_AT_EXIT, send mail to the users
4214 listed in the *Mail_Users* attribute of the batch job. The mail message shall contain at least

4215 the batch job identifier, queue, and server at which the batch job currently resides, and the
4216 *Job_Owner* attribute.

4217 4. Remove the batch job from the queue.

4218 If a batch server that buffers the standard error output cannot return the standard error file to
4219 the standard error path at the time the batch job exits, the batch server shall do one of the
4220 following:

- 4221 • Mail the standard error file to the batch job owner.
- 4222 • Save the standard error file and mail the location and name of the file where the standard
4223 error is stored to the batch job owner.
- 4224 • Save the standard error file and notify the user by other means, in which case the
4225 conformance document for the implementation shall document the method of notification.

4226 If a batch server that buffers the standard output cannot return the standard output file to the
4227 standard output path at the time the batch job exits, the batch server shall do one of the
4228 following:

- 4229 • Mail the standard output file to the batch job owner.
- 4230 • Save the standard output file and mail the location and name of the file where the standard
4231 output is stored to the batch job owner.
- 4232 • Save the standard output file and notify the user by other means, in which case the
4233 conformance document for the implementation shall document the method of notification.

4234 At the conclusion of job exit processing, the batch job is no longer managed by a batch server.

4235 3.2.2.4 *Batch Server Restart*

4236 A batch server that has been either shutdown or terminated abnormally, and has returned to
4237 operation, is said to have *restarted*.

4238 Upon restarting, a batch server shall requeue those jobs managed by the batch server that were
4239 in the RUNNING state at the time the batch server shut down and for which the *Rerunable*
4240 attribute of the batch job has the value TRUE.

4241 Queues are defined to be non-volatile. A batch server shall store the content of queues that it
4242 controls in such a way that server and system shutdowns do not erase the content of the queues.

4243 3.2.2.5 *Batch Job Abort*

4244 A batch server that cannot perform a deferred service for a batch job shall abort the batch job.

4245 A batch server that aborts a batch job shall perform the following services:

- 4246 • Delete the batch job from the queue in which it resides.
- 4247 • If the *Mail_Points* attribute of the batch job includes the value MAIL_AT_ABORT, send mail
4248 to the users listed in the value of the *Mail_Users* attribute of the job. The mail message shall
4249 contain at least the batch job identifier, queue, and server at which the batch job currently
4250 resides, the *Job_Owner* attribute, and the reason for the abort.
- 4251 • If the batch job was in the RUNNING state, terminate the session leader of the executing job
4252 by sending the session leader a SIGKILL, place the batch job in the EXITING state, and
4253 perform the services of *Batch Job Exit*.

4254 **3.2.3 Requested Batch Services**

4255 This section describes the services provided by batch servers in response to requests from
 4256 clients. Table 3-5 summarizes the current set of batch service requests and for each gives its type
 4257 (deferred or not) and whether it is an optional function.

4258 **Table 3-5 Batch Services Summary**

Batch Service	Deferred	Optional
4259 <i>Batch Job Execution</i>	Yes	No
4260 <i>Batch Job Routing</i>	Yes	No
4261 <i>Batch Job Exit</i>	Yes	No
4262 <i>Batch Server Restart</i>	Yes	No
4263 <i>Batch Job Abort</i>	Yes	No
4264 <i>Delete Batch Job Request</i>	No	No
4265 <i>Hold Batch Job Request</i>	No	No
4266 <i>Batch Job Message Request</i>	No	Yes
4267 <i>Batch Job Status Request</i>	No	No
4268 <i>Locate Batch Job Request</i>	No	Yes
4269 <i>Modify Batch Job Request</i>	No	No
4270 <i>Move Batch Job Request</i>	No	No
4271 <i>Queue Batch Job Request</i>	No	No
4272 <i>Batch Queue Status Request</i>	No	No
4273 <i>Release Batch Job Request</i>	No	No
4274 <i>Rerun Batch Job Request</i>	No	No
4275 <i>Select Batch Jobs Request</i>	No	No
4276 <i>Server Shutdown Request</i>	No	No
4277 <i>Server Status Request</i>	No	No
4278 <i>Signal Batch Job Request</i>	No	No
4279 <i>Track Batch Job Request</i>	No	Yes
4280		

4281 If a request is rejected because the batch client is not authorized to perform the action, the batch
 4282 server shall return the same status as when the batch job does not exist.

4283 **3.2.3.1 Delete Batch Job Request**

4284 A batch job is defined to have been deleted when it has been removed from the queue in which it
 4285 resides and not instantiated in another queue. A client requests that the server that manages a
 4286 batch job delete the batch job. Such a request is called a *Delete Batch Job Request*.

4287 A batch server shall reject a *Delete Batch Job Request* if any of the following statements are true:

- 4288 • The user of the batch client is not authorized to delete the designated job.
- 4289 • The designated job is not managed by the batch server.
- 4290 • The designated job is in a state inconsistent with the delete request.

4291 A batch server may reject a *Delete Batch Job Request* for other reasons. The conformance document
 4292 for an implementation shall describe the reasons for which a *Delete Batch Job Request* may be
 4293 rejected. The conformance document for an implementation shall describe the method used to
 4294 determine whether the user of a client is authorized to perform the requested action.

4295 A batch server requested to delete a batch job shall delete the batch job if the batch job exists and
 4296 is not in the EXITING state.

4297 A batch server that deletes a batch job in the RUNNING state shall send a SIGKILL signal to the
4298 session leader of the batch job. A batch server may send additional signals to the session leader
4299 of the job prior to sending the SIGKILL signal. The conformance document for such a batch
4300 server shall document the signals that are sent to the session leader.

4301 A batch server that deletes a batch job in the RUNNING state shall place the batch job in the
4302 EXITING state after it has killed the session leader of the batch job and shall perform the services
4303 of batch job exit.

4304 3.2.3.2 *Hold Batch Job Request*

4305 A batch client can request that the batch server add one or more holds to a batch job. Such a
4306 request is called a *Hold Batch Job Request*.

4307 A batch server shall reject a *Hold Batch Job Request* if any of the following statements are true:

- 4308 • The batch server does not support one or more of the requested holds to be added to the
4309 batch job.
- 4310 • The user of the batch client is not authorized to add one or more of the requested holds to the
4311 batch job.
- 4312 • The batch server does not manage the specified job.
- 4313 • The designated job is in the EXITING state.

4314 A batch server may reject a *Hold Batch Job Request* for other reasons. The conformance document
4315 for an implementation shall document the reasons for which a *Hold Batch Job Request* may be
4316 rejected. The conformance document for an implementation shall describe the method used to
4317 determine whether the user of a client is authorized to perform the requested action.

4318 A batch server that accepts a *Hold Batch Job Request* for a batch job in the RUNNING state shall
4319 place a hold on the batch job. The conformance document shall describe what effect, if any, the
4320 hold will have on a batch job in the RUNNING state.

4321 A batch server that accepts a *Hold Batch Job Request* shall add each type of hold listed in the *Hold*
4322 *Batch Job Request*, that is not already present, to the value of the *Hold_Types* attribute of the batch
4323 job.

4324 3.2.3.3 *Batch Job Message Request*

4325 *Batch Job Message Request* is an optional feature of batch servers. If an implementation supports
4326 *Batch Job Message Request*, the statements in this section apply and the configuration variable
4327 POSIX2_PBS_MESSAGE shall be set to 1.

4328 A batch client can request that a batch server write a message into certain output files of a batch
4329 job. Such a request is called a *Batch Job Message Request*.

4330 A batch server shall reject a *Batch Job Message Request* if any of the following statements are true:

- 4331 • The batch server does not support sending messages to jobs.
- 4332 • The user of the batch client is not authorized to post a message to the designated job.
- 4333 • The designated job does not exist on the batch server.
- 4334 • The designated job is not in the RUNNING state.

4335 A batch server may reject a *Batch Job Message Request* for other reasons. The conformance
4336 document for an implementation shall describe the reasons for which a *Batch Job Message Request*
4337 may be rejected. The conformance document for an implementation shall describe the method

- 4338 used to determine whether the user of a client is authorized to perform the requested action.
- 4339 A batch server that accepts a *Batch Job Message Request* shall write the message sent by the batch
4340 client into the files indicated by the batch client.
- 4341 **3.2.3.4 Batch Job Status Request**
- 4342 A batch client can request that a batch server respond with the status and attributes of a batch
4343 job. Such a request is called a *Batch Job Status Request*.
- 4344 A batch server shall reject a *Batch Job Status Request* if any of the following statements are true:
- 4345 • The user of the batch client is not authorized to query the status of the designated job.
- 4346 • The designated job is not managed by the batch server.
- 4347 A batch server may reject a *Batch Job Status Request* for other reasons. The conformance
4348 document for an implementation shall describe the reasons for which a *Batch Job Status Request*
4349 may be rejected. The conformance document for an implementation shall describe the method
4350 used to determine whether the user of a client is authorized to perform the requested action.
- 4351 A batch server that accepts a *Batch Job Status Request* shall return a *Batch Job Status Message* to the
4352 batch client.
- 4353 A batch server may return other information in response to a *Batch Job Status Request*.
- 4354 **3.2.3.5 Locate Batch Job Request**
- 4355 *Locate Batch Job Request* is an optional feature of batch servers. If an implementation supports
4356 *Locate Batch Job Request*, the statements in this section apply and the configuration variable
4357 POSIX2_PBS_LOCATE shall be set to 1.
- 4358 A batch client can ask a batch server to respond with the location of a batch job that was created
4359 by the batch server. Such a request is called a *Locate Batch Job Request*.
- 4360 A batch server that accepts a *Locate Batch Job Request* shall return a *Batch Job Location Message* to
4361 the batch client.
- 4362 A batch server may reject a *Locate Batch Job Request* for a batch job that was not created by that
4363 server.
- 4364 A batch server may reject a *Locate Batch Job Request* for a batch job that is no longer managed by
4365 that server; that is, for a batch job that is not in a queue owned by that server.
- 4366 A batch server may reject a *Locate Batch Job Request* for other reasons. The conformance
4367 document for an implementation shall document the reasons for which a *Locate Batch Job Request*
4368 may be rejected.
- 4369 **3.2.3.6 Modify Batch Job Request**
- 4370 Batch clients modify (alter) the attributes of a batch job by making a request to the server that
4371 manages the batch job. Such a request is called a *Modify Batch Job Request*.
- 4372 A batch server shall reject a *Modify Batch Job Request* if any of the following statements are true:
- 4373 • The user of the batch client is not authorized to make the requested modification to the batch
4374 job.
- 4375 • The designated job is not managed by the batch server.
- 4376 • The requested modification is inconsistent with the state of the batch job.

- 4377 • An unrecognized resource is requested for a batch job in an execution queue.
- 4378 A batch server may reject a *Modify Batch Job Request* for other reasons. The conformance
4379 document for an implementation shall describe the reasons for which a *Modify Batch Job Request*
4380 may be rejected. The conformance document for an implementation shall describe the method
4381 used to determine whether the user of a client is authorized to perform the requested action.
- 4382 A batch server that accepts a *Modify Batch Job Request* shall modify all the specified attributes of
4383 the batch job. A batch server that rejects a *Modify Batch Job Request* shall modify none of the
4384 attributes of the batch job.
- 4385 If the servicing by a batch server of an otherwise valid request would result in no change, then
4386 the batch server shall indicate successful completion of the request.
- 4387 3.2.3.7 *Move Batch Job Request*
- 4388 A batch client can request that a batch server move a batch job to another destination. Such a
4389 request is called a *Move Batch Job Request*.
- 4390 A batch server shall reject a *Move Batch Job Request* if any of the following statements are true:
- 4391 • The user of the batch client is not authorized to remove the designated job from the queue in
4392 which the batch job resides.
- 4393 • The user of the batch client is not authorized to move the designated job to the destination.
- 4394 • The designated job is not managed by the batch server.
- 4395 • The designated job is in the EXITING state.
- 4396 • The destination is inaccessible.
- 4397 A batch server can reject a *Move Batch Job Request* for other reasons. The conformance document
4398 for an implementation shall describe the reasons for which a *Move Batch Job Request* may be
4399 rejected. The conformance document for an implementation shall describe the method used to
4400 determine whether the user of a client is authorized to perform the requested action.
- 4401 A batch server that accepts a *Move Batch Job Request* shall perform the following services:
- 4402 • Queue the designated job at the destination.
- 4403 • Remove the designated job from the queue in which the batch job resides.
- 4404 If the destination resides on another batch server, the batch server shall queue the batch job at
4405 the destination by sending a *Queue Batch Job Request* to the other server. If the *Queue Batch Job*
4406 *Request* fails, the batch server shall reject the *Move Batch Job Request*. If the *Queue Batch Job Request*
4407 succeeds, the batch server shall remove the batch job from its queue.
- 4408 The batch server shall not modify any attributes of the batch job.
- 4409 3.2.3.8 *Queue Batch Job Request*
- 4410 A batch queue is controlled by one and only one batch server. A batch server is said to own the
4411 queues that it controls. Batch clients make requests of batch servers to have jobs queued. Such a
4412 request is called a *Queue Batch Job Request*.
- 4413 A batch server requested to queue a batch job for which the queue is unspecified shall select a
4414 queue for the batch job. Such a queue is called the *default queue* of the batch server. The
4415 conformance document for the implementation shall document the means by which the batch
4416 server determines the default queue. The implementation shall provide the means for a batch
4417 administrator to specify the default queue. The queue, whether specified or defaulted, is called

4418 the *target queue*.

4419 A batch server shall reject a *Queue Batch Job Request* if any of the following statements are true:

- 4420 • The client is not authorized to create a batch job in the target queue.
- 4421 • The request specifies a queue that does not exist on the batch server.
- 4422 • The target queue is an execution queue and the batch server cannot satisfy a resource
4423 requirement of the batch job.
- 4424 • The target queue is an execution queue and an unrecognized resource is requested.
- 4425 • The target queue is an execution queue, the batch server does not support checkpointing, and
4426 the value of the *Checkpoint* attribute of the batch job is not NO_CHECKPOINT.
- 4427 • The job requires access to a user identifier that the batch client is not authorized to access.

4428 A batch server may reject a *Queue Batch Job Request* for other reasons. The conformance
4429 document for an implementation shall document the reasons for which a *Queue Batch Job Request*
4430 may be rejected.

4431 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the
4432 PBS_O_QUEUE value is missing from the value of the *Variable_List* attribute of the batch job
4433 shall add that variable to the list and set the value to the name of the target queue. Once set, no
4434 server shall change the value of PBS_O_QUEUE, even if the batch job is moved to another
4435 queue.

4436 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the PBS_JOBID
4437 value is missing from the value of the *Variable_List* attribute shall add that variable to the list and
4438 set the value to the batch job identifier assigned by the server in the format:

4439 sequence_number.server

4440 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the
4441 PBS_JOBNAME value is missing from the value of the *Variable_List* attribute of the batch job
4442 shall add that variable to the list and set the value to the *Job_Name* attribute of the batch job.

4443 3.2.3.9 *Batch Queue Status Request*

4444 A batch client can request that a batch server respond with the status and attributes of a queue.
4445 Such a request is called a *Batch Queue Status Request*.

4446 A batch server shall reject a *Batch Queue Status Request* if any of the following statements are true:

- 4447 • The user of the batch client is not authorized to query the status of the designated queue.
- 4448 • The designated queue does not exist on the batch server.

4449 A batch server may reject a *Batch Queue Status Request* for other reasons. The conformance
4450 document for an implementation shall describe the reasons for which a *Batch Queue Status*
4451 *Request* is rejected. The conformance document for an implementation shall describe the method
4452 used to determine whether the user of a client is authorized to perform the requested action.

4453 A batch server that accepts a *Batch Queue Status Request* shall return a *Batch Queue Status Reply* to
4454 the batch client.

4455 3.2.3.10 *Release Batch Job Request*

4456 A batch client can request that server remove one or more holds from a batch job. Such a request
4457 is called a *Release Batch Job Request*.

4458 A batch server shall reject a *Release Batch Job Request* if any of the following statements are true:

- 4459 • The user of the batch client is not authorized to remove one or more of the requested holds
4460 from the batch job.
- 4461 • The batch server does not manage the specified job.

4462 A batch server may reject a *Release Batch Job Request* for other reasons. The conformance
4463 document for an implementation shall document the reasons for which a *Release Batch Job*
4464 *Request* may be rejected. The conformance document for an implementation shall describe the
4465 method used to determine whether the user of a client is authorized to perform the requested
4466 action.

4467 A batch server that accepts a *Release Batch Job Request* shall remove each type of hold listed in the
4468 *Release Batch Job Request*, that is present, from the value of the *Hold_Types* attribute of the batch
4469 job.

4470 3.2.3.11 *Rerun Batch Job Request*

4471 To rerun a batch job is to kill the session leader of the batch job and leave the batch job eligible
4472 for re-execution. A batch client can request that a batch server rerun a batch job. Such a request is
4473 called *Rerun Batch Job Request*.

4474 A batch server shall reject a *Rerun Batch Job Request* if any of the following statements are true:

- 4475 • The user of the batch client is not authorized to rerun the designated job.
- 4476 • The *Rerunable* attribute of the designated job has the value FALSE.
- 4477 • The designated job is not in the RUNNING state.
- 4478 • The batch server does not manage the designated job.

4479 A batch server may reject a *Rerun Batch Job Request* for other reasons. The conformance document
4480 for an implementation shall describe the reasons for which a *Rerun Batch Job Request* may be
4481 rejected. The conformance document for an implementation shall describe the method used to
4482 determine whether the user of a client is authorized to perform the requested action.

4483 A batch server that rejects a *Rerun Batch Job Request* shall in no way modify the execution of the
4484 batch job.

4485 A batch server that accepts a request to rerun a batch job shall perform the following services:

- 4486 • Requeue the batch job in the execution queue in which it was executing.
- 4487 • Send a SIGKILL signal to the process group of the session leader of the batch job.

4488 An implementation may indicate to the batch job owner that the batch job has been rerun. The
4489 conformance document for an implementation shall state whether the batch job owner is
4490 notified that a batch job is rerun, and if so, shall describe the means used.

4491 A batch server that reruns a batch job may send other signals to the session leader of the batch
4492 job prior to sending the SIGKILL signal. The conformance document for an implementation
4493 shall describe any other signals that may be sent.

4494 A batch server may preferentially select a rerun job for execution. The conformance document
4495 for an implementation shall state whether rerun jobs shall be selected for execution before other

4496 jobs.

4497 **3.2.3.12 Select Batch Jobs Request**

4498 A batch client can request from a batch server a list of jobs managed by that server that match a
4499 list of selection criteria. Such a request is called a *Select Batch Jobs Request*. All the batch jobs
4500 managed by the batch server that receives the request are candidates for selection.

4501 A batch server that accepts a *Select Batch Jobs Request* shall return a list of zero or more job
4502 identifiers that correspond to jobs that meet the selection criteria.

4503 If the batch client is not authorized to query the status of a batch job, the batch server shall not
4504 select the batch job.

4505 **3.2.3.13 Server Shutdown Request**

4506 A batch server is defined to have shut down when it does not respond to requests from clients
4507 and does not perform deferred services for jobs. A batch client can request that a batch server
4508 shut down. Such a request is called a *Server Shutdown Request*.

4509 A batch server shall reject a *Server Shutdown Request* from a client that is not authorized to shut
4510 down the batch server. The conformance document for an implementation shall describe the
4511 method used to determine whether the user of a client is authorized to perform the requested
4512 action.

4513 A batch server may reject a *Server Shutdown Request* for other reasons. The conformance
4514 document for an implementation shall document the reasons for which a *Server Shutdown*
4515 *Request* may be rejected.

4516 At server shutdown, a batch server shall do, in order of preference, one of the following:

- 4517 • If checkpointing is implemented and the batch job is checkpointable, then checkpoint the
4518 batch job and requeue it.
- 4519 • If the batch job is rerunnable, then requeue the batch job to be rerun (restarted from the
4520 beginning).
- 4521 • Abort the batch job.

4522 **3.2.3.14 Server Status Request**

4523 A batch client can request that a batch server respond with the status and attributes of the batch
4524 server. Such a request is called a *Server Status Request*.

4525 A batch server shall reject a *Server Status Request* if the following statement is true:

- 4526 • The user of the batch client is not authorized to query the status of the designated server.

4527 A batch server may reject a *Server Status Request* for other reasons. The conformance document
4528 for an implementation shall describe the reasons for which a *Server Status Request* may be
4529 rejected. The conformance document for an implementation shall describe the method used to
4530 determine whether the user of a client is authorized to perform the requested action.

4531 A batch server that accepts a *Server Status Request* shall return a *Server Status Reply* to the batch
4532 client.

4533 3.2.3.15 *Signal Batch Job Request*

4534 A batch client can request that a batch server signal the session leader of a batch job. Such a
4535 request is called a *Signal Batch Job Request*.

4536 A batch server shall reject a *Signal Batch Job Request* if any of the following statements are true:

- 4537 • The user of the batch client is not authorized to signal the batch job.
- 4538 • The job is not in the RUNNING state.
- 4539 • The batch server does not manage the designated job.
- 4540 • The requested signal is not supported by the implementation.

4541 A batch server may reject a *Signal Batch Job Request* for other reasons. The conformance
4542 document for an implementation shall describe the reasons for which a *Signal Batch Job Request*
4543 may be rejected. The conformance document for an implementation shall describe the method
4544 used to determine whether the user of a client is authorized to perform the requested action.

4545 A batch server that accepts a request to signal a batch job shall send the signal requested by the
4546 batch client to the process group of the session leader of the batch job.

4547 3.2.3.16 *Track Batch Job Request*

4548 *Track Batch Job Request* is an optional feature of batch servers. If an implementation supports
4549 *Track Batch Job Request*, the statements in this section apply and the configuration variable
4550 POSIX2_PBS_TRACK shall be set to 1.

4551 *Track Batch Job Request* provides a method for tracking the current location of a batch job. Clients
4552 may use the tracking information to determine the batch server that should receive a batch
4553 server request.

4554 If *Track Batch Job Request* is supported by a batch server, then when the batch server queues a
4555 batch job as a result of a *Queue Batch Job Request*, and the batch server is not the batch server that
4556 created the batch job, the batch server shall send a *Track Batch Job Request* to the batch server that
4557 created the job.

4558 If *Track Batch Job Request* is supported by a batch server, then the *Track Batch Job Request* may also
4559 be sent to other servers as a backup to the primary server. The method by which backup servers
4560 are specified is implementation-defined.

4561 If *Track Batch Job Request* is supported by a batch server that receives a *Track Batch Job Request*,
4562 then the batch server shall record the current location of the batch job as contained in the
4563 request.

4564 3.3 Common Behavior for Batch Environment Utilities

4565 3.3.1 Batch Job Identifier

4566 A utility shall recognize *job_identifiers* of the format:

4567 [sequence_number][.server_name][@server]

4568 where:

4569 *sequence_number* An integer that, when combined with *server_name*, provides a batch job
4570 identifier that is unique within the batch system.

4571 *server_name* The name of the batch server to which the batch job was originally submitted.

4572 *server* The name of the batch server that is currently managing the batch job.

4573 If the application omits the batch *server_name* portion of a batch job identifier, a utility shall use
4574 the name of a default batch server.

4575 If the application omits the batch *server* portion of a batch job identifier, a utility shall use:

- 4576 • The batch server indicated by *server_name*, if present.
- 4577 • The name of the default batch server.
- 4578 • The name of the batch server that is currently managing the batch job.

4579 If only *@server* is specified, then the status of all jobs owned by the user on the requested server
4580 is listed.

4581 The means by which a utility determines the default batch server is implementation-defined.

4582 If the application presents the batch *server* portion of a batch job identifier to a utility, the utility
4583 shall send the request to the specified server.

4584 A strictly conforming application shall use the syntax described for the job identifier. Whenever
4585 a batch job identifier is specified whose syntax is not recognized by an implementation, then a
4586 message for each error that occurs shall be written to standard error and the utility shall exit
4587 with an exit status greater than zero.

4588 When a batch job identifier is supplied as an argument to a batch utility and the *server_name*
4589 portion of the batch job identifier is omitted, then the utility shall use the name of the default
4590 batch server.

4591 When a batch job identifier is supplied as an argument to a batch utility and the batch *server*
4592 portion of the batch job identifier is omitted, then the utility shall use either:

- 4593 • The name of the default batch server

4594 or:

- 4595 • The name of the batch server that is currently managing the batch job

4596 When a batch job identifier is supplied as an argument to a batch utility and the batch *server*
4597 portion of the batch job identifier is specified, then the utility shall send the required *Batch Server*
4598 *Request* to the specified server.

4599 **3.3.2 Destination**4600 The utility shall recognize a *destination* of the format:

4601 [queue][@server]

4602 where:

4603 *queue* The name of a valid execution or routing queue at the batch server denoted by
4604 @*server*, defined as a string of up to 15 alphanumeric characters in the portable
4605 character set (see the Base Definitions volume of IEEE Std. 1003.1-200x,
4606 Section 6.1, Portable Character Set) where the first character is alphabetic.4607 *server* The name of a batch server, defined as a string of alphanumeric characters in
4608 the portable character set.4609 If the application omits the batch *server* portion of a destination, then the utility shall use either:

- 4610 • The name of the default batch server

4611 or:

- 4612 • The name of the batch server that is currently managing the batch job

4613 The means by which a utility determines the default batch server is implementation-defined.

4614 If the application omits the *queue* portion of a destination, then the utility shall use the name of
4615 the default queue at the batch server chosen.

4616 The means by which a batch server determines its default queue is implementation-defined.

4617 If a destination is specified in the *queue*@*server* form, then the utility shall use the specified queue
4618 at the specified server.4619 A strictly conforming application shall use the syntax described for a destination. Whenever a
4620 destination is specified whose syntax is not recognized by an implementation, then a message
4621 shall be written to standard error and the utility shall exit with an exit status greater than zero.4622 **3.3.3 Multiple Keyword-Value Pairs**4623 For each option that can have multiple keyword-value pair arguments, the following rules shall
4624 apply. Examples of options that can have list-oriented option-arguments are `-u value@keyword`
4625 and `-l keyword=value`.

- 4626 1. If a batch utility is presented with a list-oriented option-argument for which a keyword has
-
- 4627 a corresponding value that begins with a single or double quote, then the utility shall stop
-
- 4628 interpreting the input stream for delimiters until a second single or double quote,
-
- 4629 respectively, is encountered. This feature allows some flexibility for a comma (',') or
-
- 4630 equals sign ('=') to be part of the value string for a particular keyword; for example:

4631 `keywd1='val1,val2',keywd2="val3,val4"`4632 **Note:** This may require the user to escape the quotes as in the following command:4633 `foo -xkeywd1=\'val1,val2\',keywd2=\"val3,val4\"`

- 4634 2. If a batch server is presented with a list-oriented attribute that has a keyword that was
-
- 4635 encountered earlier in the list, then the later entry for that keyword shall replace the earlier
-
- 4636 entry.

- 4637 3. If a batch server is presented with a list-oriented attribute that has a keyword without any
-
- 4638 corresponding value of the form
- keyword*
- = or @
- keyword*
- and the same keyword was
-
- 4639 encountered earlier in the list, then the prior entry for that keyword shall be ignored by the

- 4640 batch server.
- 4641 4. If a batch utility is expecting a list-oriented option-argument entry of the form
4642 *keyword=value*, but is presented with an entry of the form *keyword* without any
4643 corresponding *value*, then the entry shall be treated as though a default value of NULL was
4644 assigned (that is, *keyword=NULL*) for entry parsing purposes. The utility shall include only
4645 the keyword, not the NULL value, in the associated job attribute.
- 4646 5. If a batch utility is expecting a list-oriented option-argument entry of the form
4647 *value@keyword*, but is presented with an entry of the form *value* without any corresponding
4648 *keyword*, then the entry shall be treated as though a keyword of NULL was assigned (that
4649 is, *value@NULL*) for entry parsing purposes. The utility shall include only the value, not
4650 the NULL keyword, in the associated job attribute.
- 4651 6. A batch server shall accept a list-oriented attribute that has multiple occurrences of the
4652 same keyword, interpreting the keywords, in order, with the last value encountered taking
4653 precedence over prior instances of the same keyword. This rule allows, but does not
4654 require, a batch utility to preprocess the attribute to remove duplicate keywords.
- 4655 7. If a batch utility is presented with multiple list-oriented option-arguments on the
4656 command line or in script directives, or both, for a single option, then the utility shall
4657 concatenate, in order, any command line keyword and value pairs to the end of any
4658 directive keyword and value pairs separated by a single comma to produce a single string
4659 that is an equivalent, valid option-argument. The resulting string shall be assigned to the
4660 associated attribute of the batch job (after optionally removing duplicate entries as
4661 described in item 6.

Chapter 4 *Utilities*

4662

4663

This chapter contains the definitions of the utilities, as follows:

4664

- Mandatory utilities that are present on every conformant system

4665

4666

4667

- Optional utilities that are present only on systems supporting the associated option; see Section 1.8.1 (on page 2212) for information on the options in this volume of IEEE Std. 1003.1-200x

4668 NAME

4669 admin — create and administer SCCS files (DEVELOPMENT)

4670 SYNOPSIS

```
4671 xSI admin -i[name][-n][-a login][-d flag][-f flag][-m mrlist][-r rel]
4672 [-t[name][-y[comment]] newfile
```

```
4673 admin -n[-a login][-d flag][-f flag][-m mrlist][-t[name]][-y[comment]]
4674 newfile ...
```

```
4675 admin [-a login][-d flag][-m mrlist][-r rel][-t[name]] file ...
```

```
4676 admin -h file ...
```

```
4677 admin -z file ...
```

4678

4679 DESCRIPTION

4680 The *admin* utility shall create new SCCS files or change parameters of existing ones. If a named
 4681 file does not exist, it shall be created, and its parameters shall be initialized according to the
 4682 specified options. Parameters not initialized by an option shall be assigned a default value. If a
 4683 named file does exist, parameters corresponding to specified options shall be changed, and other
 4684 parameters shall be left as is.

4685 All SCCS file names supplied by the application shall be of the form *s.filename*. New SCCS files
 4686 shall be given read-only permission mode. Write permission in the parent directory is required
 4687 to create a file. All writing done by *admin* shall be to a temporary *x-file*, named *x.filename* (see *get*)
 4688 created with read-only mode if *admin* is creating a new SCCS file, or created with the same mode
 4689 as that of the SCCS file if the file already exists. After successful execution of *admin*, the SCCS file
 4690 shall be removed (if it exists), and the *x-file* shall be renamed with the name of the SCCS file. This
 4691 ensures that changes are made to the SCCS file only if no errors occur.

4692 The *admin* utility shall also use a transient lock file (named *z.filename*), which is used to prevent
 4693 simultaneous updates to the SCCS file; see *get* (on page 2685).

4694 OPTIONS

4695 The *admin* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 4696 12.2, Utility Syntax Guidelines, except that the *-i*, *-t*, and *-y* options have optional option-
 4697 arguments. These optional option-arguments shall not be presented as separate arguments. The
 4698 following options are supported:

4699 **-n** Create a new SCCS file. When *-n* is used without *-i*, the SCCS file shall be created
 4700 with control information but without any file data.

4701 **-i[name]** Specify the *name* of a file from which the text for a new SCCS file shall be taken.
 4702 The text constitutes the first delta of the file (see the *-r* option for delta numbering
 4703 scheme). If the *-i* option is used, but the *name* option-argument is omitted, the text
 4704 shall be obtained by reading the standard input. If this option is omitted, the SCCS
 4705 file shall be created with control information but without any file data. The *-i*
 4706 option implies the *-n* option.

4707 **-r rel** Specify the *release* into which the initial delta is inserted. If the *-r* option is not
 4708 used, the initial delta shall be inserted into release 1. The level of the initial delta
 4709 shall always be 1 (by default, initial deltas are named 1.1).

4710 **-t[name]** Specify the *name* of a file from which descriptive text for the SCCS file shall be
 4711 taken. In the case of existing SCCS files (neither *-i* nor *-n* is specified):

- 4712 • A **-t** option without a *name* option-argument shall cause the removal of
4713 descriptive text (if any) currently in the SCCS file.
- 4714 • A **-t** option with a *name* option-argument shall cause the text (if any) in the
4715 named file to replace the descriptive text (if any) currently in the SCCS file.
- 4716 **-f flag** Specify a *flag*, and, possibly, a value for the *flag*, to be placed in the SCCS file.
4717 Several **-f** options may be supplied on a single *admin* command line. The allowable
4718 flags and their values are:
- 4719 **b** Allow use of the **-b** option on a *get* command to create branch deltas.
- 4720 **cceil** Specify the highest release (that is, ceiling), a number less than or equal to
4721 9 999, which may be retrieved by a *get* command for editing. The default
4722 value for an unspecified **c** flag shall be 9 999.
- 4723 **ffloor** Specify the lowest release (that is, floor), a number greater than 0 but less
4724 than 9 999, which may be retrieved by a *get* command for editing. The
4725 default value for an unspecified **f** flag shall be 1.
- 4726 **dSID** Specify the default delta number (SID) to be used by a *get* command.
- 4727 **istr** Treat the “No ID keywords” message issued by *get* or *delta* as a fatal
4728 error. In the absence of this flag, the message is only a warning. The
4729 message is issued if no SCCS identification keywords (see *get* (on page
4730 2685)) are found in the text retrieved or stored in the SCCS file. If a value
4731 is supplied, the application shall ensure that the keywords exactly match
4732 the given string; however, the string shall contain a keyword, and no
4733 embedded <newline>s.
- 4734 **j** Allow concurrent *get* commands for editing on the same SID of an SCCS
4735 file. This allows multiple concurrent updates to the same version of the
4736 SCCS file.
- 4737 **l!list** Specify a *list* of releases to which deltas can no longer be made (that is, *get*
4738 **-e** against one of these locked releases fails). The *list* has the following
4739 syntax:
- 4740 <list> ::= a | <range-list>
4741 <range-list> ::= <range> | <range-list>, <range>
- 4742 The character *a* in the *list* shall be equivalent to specifying all releases for
4743 the named SCCS file.
- 4744 **n** Cause *delta* to create a null delta in each of those releases (if any) being
4745 skipped when a delta is made in a new release (for example, in making
4746 delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas
4747 serve as anchor points so that branch deltas may later be created from
4748 them. The absence of this flag shall cause skipped releases to be
4749 nonexistent in the SCCS file, preventing branch deltas from being created
4750 from them in the future.
- 4751 **qtext** Substitute user-definable *text* for all occurrences of the %Q% keyword in
4752 the SCCS file text retrieved by *get*.
- 4753 **mmod** Specify the module name of the SCCS file substituted for all occurrences
4754 of the %M% keyword in the SCCS file text retrieved by *get*. If the **m** flag
4755 is not specified, the value assigned shall be the name of the SCCS file with
4756 the leading ' . ' removed.

- 4757 **type** Specify the *type* of module in the SCCS file substituted for all occurrences
4758 of the %Y% keyword in the SCCS file text retrieved by *get*.
- 4759 **vpgm** Cause *delta* to prompt for modification request (MR) numbers as the
4760 reason for creating a delta. The optional value specifies the name of an
4761 MR number validation program. (If this flag is set when creating an SCCS
4762 file, the application shall ensure that the **m** option is also used even if its
4763 value is null.)
- 4764 **-d flag** Remove (delete) the specified *flag* from an SCCS file. Several **-d** options may be
4765 supplied on a single *admin* command. See the **-f** option for allowable *flag* names.
4766 (The **l**list flag gives a *list* of releases to be unlocked. See the **-f** option for further
4767 description of the **l** flag and the syntax of a *list*.)
- 4768 **-a login** Specify a *login* name, or numerical group ID, to be added to the list of users who
4769 may make deltas (changes) to the SCCS file. A group ID is equivalent to specifying
4770 all *login* names common to that group ID. Several **-a** options may be used on a
4771 single *admin* command line. As many *logins*, or numerical group IDs, as desired
4772 may be on the list simultaneously. If the list of users is empty, then anyone may
4773 add deltas. If *login* or group ID is preceded by a '!', the users so specified are
4774 denied permission to make deltas.
- 4775 **-e login** Specify a *login* name, or numerical group ID, to be erased from the list of users
4776 allowed to make deltas (changes) to the SCCS file. Specifying a group ID is
4777 equivalent to specifying all *login* names common to that group ID. Several **-e**
4778 options may be used on a single *admin* command line.
- 4779 **-y[comment]** Insert the *comment* text into the SCCS file as a comment for the initial delta in a
4780 manner identical to that of *delta*. In the POSIX locale, omission of the **-y** option
4781 results in a default comment line being inserted in the form:
- 4782 "date and time created %s %s by %s", <date>, <time>, <login>
4783 where <date> is expressed in the *date* utility's %y/%m/%d format, <time> in the
4784 *date* utility's %T format, and <login> is the login name of the user creating the file.
- 4785 **-m mrlist** Insert the list of modification request (MR) numbers into the SCCS file as the
4786 reason for creating the initial delta in a manner identical to *delta*. The application
4787 shall ensure that the **v** flag is set and the MR numbers are validated if the **v** flag has
4788 a value (the name of an MR number validation program). Diagnostics occur if the
4789 **v** flag is not set or MR validation fails.
- 4790 **-h** Check the structure of the SCCS file and compare the newly computed checksum
4791 (the sum of all the characters in the SCCS file except those in the first line) with the
4792 checksum that is stored in the first line of the SCCS file. Appropriate error
4793 diagnostics are produced.
- 4794 **-z** Recompute the SCCS file checksum and store it in the first line of the SCCS file (see
4795 the **-h** option above). Note that use of this option on a truly corrupted file may
4796 prevent future detection of the corruption.

4797 **OPERANDS**

4798 The following operands shall be supported:

- 4799 **file** A path name of an existing SCCS file or a directory. If *file* is a directory, the *admin*
4800 utility shall behave as though each file in the directory were specified as a named
4801 file, except that non-SCCS files (last component of the path name does not begin
4802 with **s**.) and unreadable files shall be silently ignored.

- 4803 *newfile* A path name of an SCCS file to be created.
- 4804 If a single instance of *file* or *newfile* is specified as *'-'*, the standard input shall be read; each line
4805 of the standard input shall be taken to be the name of an SCCS file to be processed. Non-SCCS
4806 files and unreadable files shall be silently ignored.
- 4807 **STDIN**
4808 The standard input shall be a text file used only if the *-i* is specified without an option-argument
4809 or if a *file* or *newfile* operand is specified as *'-'*. If the first character of any standard input line is
4810 SOH (binary 001), the results are unspecified.
- 4811 **INPUT FILES**
4812 The existing SCCS files are text files of an unspecified format. The file named by the *-i* option's
4813 *name* option-argument is a text file; if the first character of any line in this file is SOH (binary
4814 001), the results are unspecified.
- 4815 **ENVIRONMENT VARIABLES**
4816 The following environment variables shall affect the execution of *admin*:
- 4817 *LANG* Provide a default value for the internationalization variables that are unset or null.
4818 If *LANG* is unset or null, the corresponding value from the implementation-
4819 defined default locale shall be used. If any of the internationalization variables
4820 contains an invalid setting, the utility shall behave as if none of the variables had
4821 been defined.
- 4822 *LC_ALL* If set to a non-empty string value, override the values of all the other
4823 internationalization variables.
- 4824 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
4825 characters (for example, single-byte as opposed to multi-byte characters in
4826 arguments and input files).
- 4827 *LC_MESSAGES*
4828 Determine the locale that should be used to affect the format and contents of
4829 diagnostic messages written to standard error and the contents of the default *-y*
4830 comment.
- 4831 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 4832 **ASYNCHRONOUS EVENTS**
4833 Default.
- 4834 **STDOUT**
4835 Not used.
- 4836 **STDERR**
4837 Used only for diagnostic messages.
- 4838 **OUTPUT FILES**
4839 Any SCCS files created shall be text files of an unspecified format. During processing of a *file*, a
4840 locking *z-file*, as described in *get* (on page 2685), may be created and deleted.
- 4841 **EXTENDED DESCRIPTION**
4842 None.
- 4843 **EXIT STATUS**
4844 The following exit values shall be returned:
4845 0 Successful completion.

4846 >0 An error occurred.

4847 **CONSEQUENCES OF ERRORS**

4848 Default.

4849 **APPLICATION USAGE**

4850 It is recommended that directories containing SCCS files be writable by the owner only, and that
4851 SCCS files themselves be read-only. The mode of the directories should allow only the owner to
4852 modify SCCS files contained in the directories. The mode of the SCCS files prevents any
4853 modification at all except by SCCS commands.

4854 **EXAMPLES**

4855 None.

4856 **RATIONALE**

4857 None.

4858 **FUTURE DIRECTIONS**

4859 None.

4860 **SEE ALSO**

4861 *delta, get, prs, what*

4862 **CHANGE HISTORY**

4863 First released in Issue 2.

4864 **Issue 4**

4865 Format reorganized.

4866 Conformance to Utility Syntax Guidelines mandated, with exceptions as noted.

4867 Internationalized environment variable support mandated.

4868 **Issue 6**

4869 The normative text is reworded to avoid use of the term “must” for application requirements. |

4870 The normative text is reworded to emphasise the term “shall” for implementation requirements. |

4871 The grammar is updated. |

4872 **NAME**

4873 alias — define or display aliases

4874 **SYNOPSIS**4875 UP alias [*alias-name*[=*string*] ...]
48764877 **DESCRIPTION**4878 The *alias* utility shall create or redefine alias definitions or write the values of existing alias
4879 definitions to standard output. An alias definition provides a string value that shall replace a
4880 command name when it is encountered; see Section 2.3.1 (on page 2239).4881 An alias definition shall affect the current shell execution environment and the execution
4882 environments of the subshells of the current shell. When used as specified by this volume of
4883 IEEE Std. 1003.1-200x, the alias definition shall not affect the parent process of the current shell
4884 nor any utility environment invoked by the shell; see Section 2.13 (on page 2273).4885 **OPTIONS**

4886 None.

4887 **OPERANDS**

4888 The following operands shall be supported:

4889 *alias-name* Write the alias definition to standard output.4890 *alias-name=string*4891 Assign the value of *string* to the alias *alias-name*.

4892 If no operands are given, all alias definitions shall be written to standard output.

4893 **STDIN**

4894 Not used.

4895 **INPUT FILES**

4896 None.

4897 **ENVIRONMENT VARIABLES**4898 The following environment variables shall affect the execution of *alias*:4899 *LANG* Provide a default value for the internationalization variables that are unset or null.
4900 If *LANG* is unset or null, the corresponding value from the implementation-
4901 defined default locale shall be used. If any of the internationalization variables
4902 contains an invalid setting, the utility shall behave as if none of the variables had
4903 been defined.4904 *LC_ALL* If set to a non-empty string value, override the values of all the other
4905 internationalization variables.4906 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
4907 characters (for example, single-byte as opposed to multi-byte characters in
4908 arguments).4909 *LC_MESSAGES*4910 Determine the locale that should be used to affect the format and contents of
4911 diagnostic messages written to standard error.4912 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

4913 **ASYNCHRONOUS EVENTS**

4914 Default.

4915 **STDOUT**4916 The format for displaying aliases (when no operands or only *name* operands are specified) shall
4917 be:4918 "%s=%s\n", *name*, *value*4919 The *value* string shall be written with appropriate quoting so that it is suitable for reinput to the
4920 shell. See the description of shell quoting in Section 2.2 (on page 2236).4921 **STDERR**

4922 Used only for diagnostic messages.

4923 **OUTPUT FILES**

4924 None.

4925 **EXTENDED DESCRIPTION**

4926 None.

4927 **EXIT STATUS**

4928 The following exit values shall be returned:

4929 0 Successful completion.

4930 >0 One of the *name* operands specified did not have an alias definition, or an error occurred.4931 **CONSEQUENCES OF ERRORS**

4932 Default.

4933 **APPLICATION USAGE**

4934 None.

4935 **EXAMPLES**4936 1. Change *ls* to give a columnated, more annotated output:4937 `alias ls="ls -CF"`

4938 2. Create a simple “redo” command to repeat previous entries in the command history file:

4939 `alias r='fc -s'`4940 3. Use 1K units for *du*:4941 `alias du=du\ -k`4942 4. Set up *nohup* so that it can deal with an argument that is itself an alias name:4943 `alias nohup="nohup "`4944 **RATIONALE**4945 The *alias* description is based on historical KornShell implementations. Known differences exist
4946 between that and the C shell. The KornShell version was adopted to be consistent with all the
4947 other KornShell features in this volume of IEEE Std. 1003.1-200x, such as command line editing.4948 Since *alias* affects the current shell execution environment, it is generally provided as a shell
4949 regular built-in.4950 Historical versions of the KornShell have allowed aliases to be exported to scripts that are
4951 invoked by the same shell. This is triggered by the *alias -x* flag; it is allowed by this volume of
4952 IEEE Std. 1003.1-200x only when an explicit extension such as *-x* is used. The standard
4953 developers considered that aliases were of use primarily to interactive users and that they

4954 should normally not affect shell scripts called by those users; functions are available to such
4955 scripts.

4956 Historical versions of the KornShell had not written aliases in a quoted manner suitable for
4957 reentry to the shell, but this volume of IEEE Std. 1003.1-200x has made this a requirement for all
4958 similar output. Therefore, consistency with this volume of IEEE Std. 1003.1-200x was chosen
4959 over this detail of historical practice.

4960 **FUTURE DIRECTIONS**

4961 None.

4962 **SEE ALSO**

4963 Section 2.9.5 (on page 2263)

4964 **CHANGE HISTORY**

4965 First released in Issue 4.

4966 **Issue 6**

4967 This utility is now marked as part of the User Portability Utilities option.

4968 The APPLICATION USAGE section is added.

4969 **NAME**

4970 ar — create and maintain library archives

4971 **SYNOPSIS**

4972 SD ar -d[-v] archive file ...

4973

4974 XSI ar -m[-abiv][posname] archive file ...

4975

4976 XSI ar -p[-v][-s]archive [file ...]

4977 XSI ar -q[-cv] archive file ...

4978

4979 XSI ar -r[-cuv][-abi][posname]archive file ...

4980 XSI ar -t[-v][-s]archive [file ...]

4981 XSI ar -x[-v][-sCT]archive [file ...]

4982 **DESCRIPTION**

4983 The *ar* utility can be used to create and maintain groups of files combined into an archive. Once
 4984 an archive has been created, new files can be added, and existing files can be extracted, deleted,
 4985 or replaced. When an archive consists entirely of valid object files, the implementation shall
 4986 format the archive so that it is usable as a library for link editing (see *c99*, *cc*, and *fort77*). When
 4987 some of the archived files are not valid object files, the suitability of the archive for library use is
 4988 XSI undefined. If an archive file consists entirely of printable files, the entire archive file is printable.

4989 When *ar* creates an archive file, it creates administrative information indicating whether a
 4990 symbol table is present in the archive. When there is at least one object file that *ar* recognizes as
 4991 such in the archive, an archive symbol table is created in the archive file and maintained by *ar*; it
 4992 is used by the link editor to search the archive file. Whenever the *ar* utility is used to create or
 4993 update the contents of such an archive, the symbol table is rebuilt. The *-s* option forces the
 4994 symbol table to be rebuilt.

4995 All *file* operands can be path names. However, files within archives shall be named by a file
 4996 name, which is the last component of the path name used when the file was entered into the
 4997 archive. The comparison of *file* operands to the names of files in archives shall be performed by
 4998 comparing the last component of the operand to the name of the archive file.

4999 It is unspecified whether multiple files in the archive may be identically named. In the case of
 5000 XSI such files, however, each *file* and *posname* operand shall match only the first archive file having a
 5001 name that is the same as the last component of the operand.

5002 **OPTIONS**

5003 The *ar* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,
 5004 Utility Syntax Guidelines.

5005 The following options shall be supported:

5006 XSI **-a** Position new files in the archive after the file named by the *posname* operand.

5007 XSI **-b** Position new files in the archive before the file named by the *posname* operand.

5008 **-c** Suppress the diagnostic message that is written to standard error by default when
 5009 the archive file *archive* is created.

5010 XSI **-C** Prevent extracted files from replacing like-named files in the file system. This
 5011 option is useful when **-T** is also used, to prevent truncated file names from
 5012 replacing files with the same prefix.

5013		-d	Delete one or more <i>files</i> from <i>archive</i> .
5014	XSI	-i	Position new files in the archive before the file named by the <i>posname</i> operand (equivalent to -b).
5015			
5016	XSI	-m	Move the named files. The -a , -b , or -i options with the <i>posname</i> operand indicate the position; otherwise, move the files to the end of the archive.
5017			
5018		-p	Write the contents of the <i>files</i> from <i>archive</i> to the standard output. If no <i>files</i> are specified, the contents of all files in the archive shall be written in the order of the archive.
5019			
5020			
5021	XSI	-q	Quickly append the named files to the end of the archive file. In this case <i>ar</i> does not check whether the added members are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece by piece.
5022			
5023			
5024		-r	Replace or add <i>files</i> to <i>archive</i> . If the archive named by <i>archive</i> does not exist, a new archive file shall be created and a diagnostic message shall be written to standard error (unless the -c option is specified). If no <i>files</i> are specified and the <i>archive</i> exists, the results are undefined. Files that replace existing files shall not change the order of the archive. Files that do not replace existing files shall be appended to the archive unless a -a , -b , or -i option specifies another position.
5025			
5026			
5027			
5028			
5029	XSI		
5030	XSI	-s	Force the regeneration of the archive symbol table even if <i>ar</i> is not invoked with an option that modifies the archive file contents. This option is useful to restore the archive symbol table after it has been stripped; see <i>strip</i> .
5031			
5032			
5033		-t	Write a table of contents of <i>archive</i> to the standard output. The files specified by the <i>file</i> operands shall be included in the written list. If no <i>file</i> operands are specified, all files in <i>archive</i> shall be included in the order of the archive.
5034			
5035			
5036	XSI	-T	Allow file name truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long is an error; a diagnostic message is written and the file is not extracted.
5037			
5038			
5039		-u	Update older files. When used with the -r option, files within the archive are replaced only if the corresponding <i>file</i> has a modification time that is at least as new as the modification time of the file within the archive.
5040			
5041			
5042		-v	Give verbose output. When used with the option characters -d , -r , or -x , write a detailed file-by-file description of the archive creation and maintenance activity, as described in the STDOUT section.
5043			
5044			
5045			When used with -p , write the name of the file to the standard output before writing the file itself to the standard output, as described in the STDOUT section.
5046			
5047			When used with -t , include a long listing of information about the files within the archive, as described in the STDOUT section.
5048			
5049		-x	Extract the files named by the <i>file</i> operands from <i>archive</i> . The contents of the archive file shall not be changed. If no <i>file</i> operands are given, all files in the archive shall be extracted. The modification time of each file extracted shall be set to the time the file is extracted from the archive.
5050			
5051			
5052			

5053 OPERANDS

5054 The following operands shall be supported:

5055 *archive* A path name of the archive file.

5056 *file* A path name. Only the last component shall be used when comparing against the
5057 names of files in the archive. If two or more *file* operands have the same last path
5058 name component (basename), the results are unspecified. The implementation's
5059 archive format shall not truncate valid file names of files added to or replaced in
5060 the archive.

5061 XSI *posname* The name of a file in the archive file, used for relative positioning; see options **-m**
5062 and **-r**.

5063 **STDIN**
5064 Not used.

5065 **INPUT FILES**
5066 The input file named by *archive* shall be a file in the format created by *ar -r*.

5067 **ENVIRONMENT VARIABLES**
5068 The following environment variables shall affect the execution of *ar*:

5069 *LANG* Provide a default value for the internationalization variables that are unset or null.
5070 If *LANG* is unset or null, the corresponding value from the implementation-
5071 defined default locale shall be used. If any of the internationalization variables
5072 contains an invalid setting, the utility shall behave as if none of the variables had
5073 been defined.

5074 *LC_ALL* If set to a non-empty string value, override the values of all the other
5075 internationalization variables.

5076 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
5077 characters (for example, single-byte as opposed to multi-byte characters in
5078 arguments and input files).

5079 *LC_MESSAGES*
5080 Determine the locale that should be used to affect the format and contents of
5081 diagnostic messages written to standard error.

5082 *LC_TIME* Determine the format and content for date and time strings written by *ar -tv*.

5083 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

5084 *TMPDIR* Determine the path name that overrides the default directory for temporary files, if
5085 any.

5086 **ASYNCHRONOUS EVENTS**
5087 Default.

5088 **STDOUT**
5089 If the **-d** option is used with the **-v** option, the standard output format shall be:
5090 "d - %s\n", <*file*>
5091 where *file* is the operand specified on the command line.
5092 If the **-p** option is used with the **-v** option, *ar* shall precede the contents of each file with:
5093 "\n<%s>\n\n", <*file*>
5094 where *file* is the operand specified on the command line, if *file* operands were specified, and the
5095 name of the file in the archive if they were not.
5096 If the **-r** option is used with the **-v** option:

- 5097 • If *file* is already in the archive, the standard output format shall be:
- 5098 "r - %s\n", <*file*>
- 5099 where <*file*> is the operand specified on the command line.
- 5100 • If *file* is not already in the archive, the standard output format shall be:
- 5101 "a - %s\n", <*file*>
- 5102 where <*file*> is the operand specified on the command line.

5103 **Notes to Reviewers**

5104 *This section with side shading will not appear in the final copy. - Ed.*

5105 D3, XCU, ERN 48 suggests changing the above to "where <*file*> is the member name found to be
5106 in conflict". If the command line contains a path name which is not a simple file name (that is,
5107 contains a slash), does it print the member name (which seems what's intended) or the actual
5108 text from the command line (which is what's said)? This will eventually need to be an
5109 interpretation against .2b.

5110 If the **-t** option is used, *ar* shall write the names of the files to the standard output in the format:

5111 "%s\n", <*file*>

5112 where *file* is the operand specified on the command line, if *file* operands were specified, or the
5113 name of the file in the archive if they were not.

5114 If the **-t** option is used with the **-v** option, the standard output format shall be:

5115 "%s %u/%u %u %s %d %d:%d %d %s\n", <*member mode*>, <*user ID*>,
5116 <*group ID*>, <*number of bytes in member*>,
5117 <*abbreviated month*>, <*day-of-month*>, <*hour*>,
5118 <*minute*>, <*year*>, <*file*>

5119 where:

5120 <*file*> Shall be the operand specified on the command line, if *file* operands were specified,
5121 or the name of the file in the archive if they were not.

5122 <*member*

5123 *Shall be formatted the same as the <file mode> string defined in the STDOUT section of*
5124 *ls*, except that the first character, the <*entry type*>, is not used; the string represents
5125 the file mode of the archive member at the time it was added to or replaced in the
5126 archive.

5127 The following represent the last-modification time of a file when it was most recently added to
5128 or replaced in the archive:

5129 <*abbreviated month*>

5130 Equivalent to the %b format in *date*.

5131 <*day-of-month*>

5132 Equivalent to the %e format in *date*.

5133 <*hour*> Equivalent to the %H format in *date*.

5134 <*minute*> Equivalent to the %M format in *date*.

5135 <*year*> Equivalent to the %Y format in *date*.

5136 When *LC_TIME* does not specify the POSIX locale, a different format and order of presentation
5137 of these fields relative to each other may be used in a format appropriate in the specified locale.

5138 If the *-x* option is used with the *-v* option, the standard output format shall be:

5139 "x - %s\n", <*file*>

5140 where *file* is the operand specified on the command line, if *file* operands were specified, or the
5141 name of the file in the archive if they were not.

5142 **STDERR**

5143 Used only for diagnostic messages. The diagnostic message about creating a new archive when
5144 *-c* is not specified shall not modify the exit status.

5145 **OUTPUT FILES**

5146 Archives are files with unspecified formats.

5147 **EXTENDED DESCRIPTION**

5148 None.

5149 **EXIT STATUS**

5150 The following exit values shall be returned:

5151 0 Successful completion.

5152 >0 An error occurred.

5153 **CONSEQUENCES OF ERRORS**

5154 Default.

5155 **APPLICATION USAGE**

5156 None.

5157 **EXAMPLES**

5158 None.

5159 **RATIONALE**

5160 The archive format is not described. It is recognized that there are several known *ar* formats,
5161 which are not compatible. The *ar* utility is included, however, to allow creation of archives that
5162 are intended for use only on one machine. The archive file is specified as a file, and it can be
5163 moved as a file. This does allow an archive to be moved from one machine to another machine
5164 that uses the same implementation of *ar*.

5165 Utilities such as *pax* (and its forebears *tar* and *cpio*) also provide portable “archives”. This is a not
5166 a duplication; the *ar* utility is included to provide an interface primarily for *make* and the
5167 compilers, based on a historical model.

5168 In historical implementations, the *-q* option (available on XSI-conforming systems) is known to
5169 execute quickly because *ar* does not check on whether the added members are already in the
5170 archive. This is useful to bypass the searching otherwise done when creating a large archive
5171 piece-by-piece. These remarks may but need not remain true for a brand new implementation of
5172 this utility; hence, these remarks have been moved into the RATIONALE.

5173 BSD implementations historically required applications to provide the *-s* option whenever the
5174 archive was supposed to contain a symbol table. As in this volume of IEEE Std. 1003.1-200x,
5175 System V historically creates or updates an archive symbol table whenever an object file is
5176 removed from, added to, or updated in the archive.

5177 The OPERANDS section requires what might seem to be true without specifying it: the archive
5178 cannot truncate the file names below {NAME_MAX}. Some historical implementations do so,
5179 however, causing unexpected results for the application. Therefore, this volume of

5180 IEEE Std. 1003.1-200x makes the requirement explicit to avoid misunderstandings.

5181 According to the System V documentation, the options `-dmpqrtx` are not required to begin with
5182 a hyphen ('-'). This volume of IEEE Std. 1003.1-200x requires that a conforming application
5183 use the leading hyphen.

5184 The archive format used by the 4.4 BSD implementation is documented in this RATIONALE as
5185 an example:

5186 A file created by *ar* begins with the "magic" string "`!<arch>\n`". The rest of the archive is
5187 made up of objects, each of which is composed of a header for a file, a possible file name, and
5188 the file contents. The header is portable between machine architectures, and, if the file
5189 contents are printable, the archive is itself printable.

5190 The header is made up of six ASCII fields, followed by a two-character trailer. The fields are
5191 the object name (16 characters), the file last modification time (12 characters), the user and
5192 group IDs (each 6 characters), the file mode (8 characters), and the file size (10 characters). All
5193 numeric fields are in decimal, except for the file mode, which is in octal.

5194 The modification time is the file *st_mtime* field. The user and group IDs are the file *st_uid* and
5195 *st_gid* fields. The file mode is the file *st_mode* field. The file size is the file *st_size* field. The
5196 two-byte trailer is the string "`<newline>`".

5197 Only the name field has any provision for overflow. If any file name is more than 16
5198 characters in length or contains an embedded space, the string "`#1/`" followed by the ASCII
5199 length of the name is written in the name field. The file size (stored in the archive header) is
5200 incremented by the length of the name. The name is then written immediately following the
5201 archive header.

5202 Any unused characters in any of these fields are written as `<space>` characters. If any fields
5203 are their particular maximum number of characters in length, there is no separation between
5204 the fields.

5205 Objects in the archive are always an even number of bytes long; files that are an odd number
5206 of bytes long are padded with a `<newline>` character, although the size in the header does
5207 not reflect this.

5208 The *ar* utility description requires that (when all its members are valid object files) *ar* produce an
5209 object code library, which the linkage editor can use to extract object modules. If the linkage
5210 editor needs a symbol table to permit random access to the archive, *ar* must provide it; however,
5211 *ar* does not require a symbol table.

5212 The BSD `-o` option was omitted. It is a rare portable application that uses *ar* to extract object
5213 code from a library with concern for its modification time, since this can only be of importance
5214 to *make*. Hence, since this functionality is not deemed important for applications portability, the
5215 modification time of the extracted files is set to the current time.

5216 There is at least one known implementation (for a small computer) that can accommodate only
5217 object files for that system, disallowing mixed object and other files. The ability to handle any
5218 type of file is not only historical practice for most implementations, but is also a reasonable
5219 expectation.

5220 Consideration was given to changing the output format of *ar -tv* to the same format as the
5221 output of *ls -l*. This would have made parsing the output of *ar* the same as that of *ls*. This was
5222 rejected in part because the current *ar* format is commonly used and changes would break
5223 historical usage. Second, *ar* gives the user ID and group ID in numeric format separated by a
5224 slash. Changing this to be the user name and group name would not be correct if the archive
5225 were moved to a machine that contained a different user database. Since *ar* cannot know

- 5226 whether the archive file was generated on the same machine, it cannot tell what to report.
- 5227 The text on the `-ur` option combination is historical practice—since one file name can easily
5228 represent two different files (for example, `/a/foo` and `/b/foo`), it is reasonable to replace the
5229 member in the archive even when the modification time in the archive is identical to that in the
5230 file system.
- 5231 **FUTURE DIRECTIONS**
- 5232 None.
- 5233 **SEE ALSO**
- 5234 *c99*, *pax*, *strip* the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 13, Headers, |
5235 `<unistd.h>` description of `{POSIX_NO_TRUNC}`
- 5236 **CHANGE HISTORY**
- 5237 First released in Issue 2.
- 5238 **Issue 4**
- 5239 Aligned with the ISO/IEC 9945-2:1993 standard.
- 5240 The `-C` and `-T` options are added.
- 5241 **Issue 5**
- 5242 FUTURE DIRECTIONS section added.
- 5243 **Issue 6**
- 5244 This utility is now marked as part of the Software Development Utilities option.
- 5245 The `STDOUT` description is changed for the `-v` option to align with the IEEE P1003.2b draft
5246 standard.
- 5247 The normative text is reworded to avoid use of the term “must” for application requirements.

5248 **NAME**

5249 asa — interpret carriage-control characters

5250 **SYNOPSIS**5251 FR asa [*file* ...]

5252

5253 **DESCRIPTION**5254 The *asa* utility shall write its input files to standard output, mapping carriage-control characters
5255 from the text files to line-printer control sequences in an implementation-defined manner.5256 The first character of every line shall be removed from the input, and the following actions are
5257 performed:

5258 If the character removed is:

5259 <space> The rest of the line is output without change.

5260 0 A <newline> character is output, then the rest of the input line.

5261 1 One or more implementation-defined characters that causes an advance to the next
5262 page shall be output, followed by the rest of the input line.5263 + The <newline> character of the previous line shall be replaced with one or more
5264 implementation-defined characters that causes printing to return to column position 1,
5265 followed by the rest of the input line. If the ' + ' is the first character in the input, it shall
5266 have the same effect as the <space> character.5267 The action of the *asa* utility is unspecified upon encountering any character other than those
5268 listed above as the first character in a line.5269 **OPTIONS**

5270 None.

5271 **OPERANDS**5272 *file* A path name of a text file used for input. If no *file* operands are specified, the
5273 standard input shall be used.5274 **STDIN**5275 The standard input is used only if no *file* operands are specified; see the INPUT FILES section.5276 **INPUT FILES**

5277 The input files shall be text files.

5278 **ENVIRONMENT VARIABLES**5279 The following environment variables shall affect the execution of *asa*:5280 *LANG* Provide a default value for the internationalization variables that are unset or null.
5281 If *LANG* is unset or null, the corresponding value from the implementation-
5282 defined default locale shall be used. If any of the internationalization variables
5283 contains an invalid setting, the utility shall behave as if none of the variables had
5284 been defined.5285 *LC_ALL* If set to a non-empty string value, override the values of all the other
5286 internationalization variables.5287 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
5288 characters (for example, single-byte as opposed to multi-byte characters in
5289 arguments and input files).

5290 *LC_MESSAGES*

5291 Determine the locale that should be used to affect the format and contents of
5292 diagnostic messages written to standard error.

5293 XSI *NLS_PATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

5294 **ASYNCHRONOUS EVENTS**

5295 Default.

5296 **STDOUT**

5297 The standard output shall be the text from the input file modified as described in the
5298 DESCRIPTION section.

5299 **STDERR**

5300 None.

5301 **OUTPUT FILES**

5302 None.

5303 **EXTENDED DESCRIPTION**

5304 None.

5305 **EXIT STATUS**

5306 The following exit values shall be returned:

5307 0 All input files were output successfully.

5308 >0 An error occurred.

5309 **CONSEQUENCES OF ERRORS**

5310 Default.

5311 **APPLICATION USAGE**

5312 None.

5313 **EXAMPLES**

5314 1. The following command:

5315 *asa file*

5316 permits the viewing of *file* (created by a program using FORTRAN-style carriage control
5317 characters) on a terminal.

5318 2. The following command:

5319 *a.out | asa | lp*

5320 formats the FORTRAN output of **a.out** and directs it to the printer.

5321 **RATIONALE**

5322 The *asa* utility is needed to map “standard” FORTRAN 77 output into a form acceptable to
5323 contemporary printers. Usually, *asa* is used to pipe data to the *lp* utility; see *lp*.

5324 This utility is generally used only by FORTRAN programs. The standard developers decided to
5325 retain *asa* to avoid breaking the historical large base of FORTRAN applications that put
5326 carriage-control characters in their output files. There is no requirement that a system have a
5327 FORTRAN compiler in order to run applications that need *asa*.

5328 Historical implementations have used an ASCII <form-feed> character in response to a 1 and an
5329 ASCII <carriage-return> in response to a '+'. It is suggested that implementations treat
5330 characters other than 0, 1, and '+' as <space> in the absence of any compelling reason to do
5331 otherwise. However, the action is listed here as “unspecified”, permitting an implementation to

5332 provide extensions to access fast multiple-line slewing and channel seeking in a non-portable
5333 manner.

5334 **FUTURE DIRECTIONS**

5335 None.

5336 **SEE ALSO**

5337 *fort77, lp*

5338 **CHANGE HISTORY**

5339 First released in Issue 4.

5340 **Issue 6**

5341 This utility is now marked as part of the FORTRAN Runtime Utilities option.

5342 The normative text is reworded to avoid use of the term “must” for application requirements.

5343 NAME

5344 at — execute commands at a later time

5345 SYNOPSIS

5346 UP at [-m][-f *file*][-q *queuename*] -t *time_arg*5347 at [-m][-f *file*][-q *queuename*] *timespec* ...5348 at -r *at_job_id* ...5349 at -l -q *queuename*5350 at -l [*at_job_id* ...]

5351

5352 DESCRIPTION

5353 The *at* utility shall read commands from standard input and group them together as an *at-job*, to
5354 be executed at a later time.5355 The *at-job* shall be executed in a separate invocation of the shell, running in a separate process
5356 group with no controlling terminal, except that the environment variables, current working
5357 directory, file creation mask, and other implementation-defined execution-time attributes in
5358 effect when the *at* utility is executed shall be retained and used when the *at-job* is executed.5359 When the *at-job* is submitted, the *at_job_id* and scheduled time shall be written to standard error.
5360 The *at_job_id* is an identifier that shall be a string consisting solely of alphanumeric characters
5361 and the period character. The *at_job_id* shall be assigned by the system when the job is scheduled
5362 such that it uniquely identifies a particular job.5363 User notification and the processing of the job's standard output and standard error are
5364 described under the **-m** option.5365 XSI Users are permitted to use *at* if their name appears in the file **/usr/lib/cron/at.allow**. If that file
5366 does not exist, the file **/usr/lib/cron/at.deny** is checked to determine whether the user should be
5367 denied access to *at*. If neither file exists, only a process with the appropriate privileges is
5368 allowed to submit a job. If only **at.deny** exists and is empty, global usage is permitted. The
5369 **at.allow** and **at.deny** files consist of one user name per line.

5370 OPTIONS

5371 The *at* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,
5372 Utility Syntax Guidelines.

5373 The following options shall be supported:

5374 **-f file** Specify the path name of a file to be used as the source of the *at-job*, instead of
5375 standard input.5376 **-l** (The letter ell.) Report all jobs scheduled for the invoking user if no *at_job_id*
5377 operands are specified. If *at_job_ids* are specified, report only information for these
5378 jobs. The output shall be written to standard output.5379 **-m** Send mail to the invoking user after the *at-job* has run, announcing its completion.
5380 Standard output and standard error produced by the *at-job* shall be mailed to the
5381 user as well, unless redirected elsewhere. Mail shall be sent even if the job
5382 produces no output.5383 If **-m** is not used, the job's standard output and standard error shall be provided to
5384 the user by means of mail, unless they are redirected elsewhere; if there is no such
5385 output to provide, the implementation need not notify the user of the job's
5386 completion.

- 5387 **-q** *queuename*
 5388 Specify in which queue to schedule a job for submission. When used with the **-l**
 5389 option, limit the search to that particular queue. By default, at-jobs shall be
 5390 scheduled in queue *a*. In contrast, queue *b* shall be reserved for batch jobs; see
 5391 *batch*. The meanings of all other *queuenames* are implementation-defined. If **-q** is
 5392 specified along with either of the **-t** *time_arg* or *timespec* arguments, the results are
 5393 unspecified.
- 5394 **-r** Remove the jobs with the specified *at_job_id* operands that were previously
 5395 scheduled by the *at* utility.
- 5396 **-t** *time_arg* Submit the job to be run at the time specified by the *time* option-argument, which
 5397 the application shall ensure has the format as specified by the *touch -t time* utility.

5398 **OPERANDS**

5399 The following operands shall be supported:

5400 *at_job_id* The name reported by a previous invocation of the *at* utility at the time the job was
 5401 scheduled.

5402 *timespec* Submit the job to be run at the date and time specified. All of the *timespec* operands
 5403 are interpreted as if they were separated by <space> characters and concatenated,
 5404 and shall be parsed as described in the grammar at the end of this section. The date
 5405 and time shall be interpreted as being in the timezone of the user (as determined
 5406 by the *TZ* variable), unless a timezone name appears as part of *time*, below.

5407 In the POSIX locale, the following describes the three parts of the time
 5408 specification string. All of the values from the *LC_TIME* categories in the POSIX
 5409 locale shall be recognized in a case-insensitive manner.

5410 *time* The time can be specified as one, two, or four digits. One-digit and
 5411 two-digit numbers shall be taken to be hours; four-digit numbers to
 5412 be hours and minutes. The time can alternatively be specified as two
 5413 numbers separated by a colon, meaning *hour:minute*. An AM/PM
 5414 indication (one of the values from the **am_pm** keywords in the
 5415 *LC_TIME* locale category) can follow the time; otherwise, a 24-hour
 5416 clock time shall be understood. A timezone name can also follow to
 5417 further qualify the time. The acceptable timezone names are
 5418 implementation-defined, except that they shall be case-insensitive
 5419 and the string **utc** is supported to indicate the time is in Coordinated
 5420 Universal Time. In the POSIX locale, the *time* field can also be one of
 5421 the following tokens:

5422 **midnight** Indicates the time 12:00 am (00:00).

5423 **noon** Indicates the time 12:00 pm.

5424 **now** Indicates the current day and time. Invoking *at* <**now**>
 5425 shall submit an at-job for potentially immediate
 5426 execution (that is, subject only to unspecified
 5427 scheduling delays).

5428 *date* An optional *date* can be specified as either a month name (one of the
 5429 values from the **mon** or **abmon** keywords in the *LC_TIME* locale
 5430 category) followed by a day number (and possibly year number
 5431 preceded by a comma), or a day of the week (one of the values from
 5432 the **day** or **abday** keywords in the *LC_TIME* locale category). In the
 5433 POSIX locale, two special days shall be recognized:

5434 **today** Indicates the current day.

5435 **tomorrow** Indicates the day following the current day.

5436 If no *date* is given, **today** shall be assumed if the given time is greater
5437 than the current time, and **tomorrow** shall be assumed if it is less. If
5438 the given month is less than the current month (and no year is given),
5439 next year shall be assumed.

5440 *increment* The optional *increment* shall be a number preceded by a plus sign
5441 ('+') and suffixed by one of the following: **minutes, hours, days,**
5442 **weeks, months, or years.** (The singular forms shall be also
5443 accepted.) The keyword **next** shall be equivalent to an increment
5444 number of +1. For example, the following are equivalent commands:

5445 at 2pm + 1 week
5446 at 2pm next week

5447 The following grammar describes the precise format of *timespec* in the POSIX locale. The general
5448 conventions for this style of grammar are described in Section 1.10 (on page 2223). This formal
5449 syntax shall take precedence over the preceding text syntax description. The longest possible
5450 token or delimiter shall be recognized at a given point. When used in a *timespec*, white space
5451 shall also delimit tokens.

```
5452 %token hr24clock_hr_min
5453 %token hr24clock_hour
5454 /*
5455     A hr24clock_hr_min is a one, two, or four-digit number. A one-digit
5456     or two-digit number constitutes a hr24clock_hour. A hr24clock_hour
5457     may be any of the single digits 0-9, or may be double digits, ranging
5458     from 00-23. If a hr24clock_hr_min is a four digit number, the
5459     first two digits shall be a valid hr24clock_hour, while the last two
5460     represent the number of minutes, from 00-59.
5461 */
```

```
5462 %token wallclock_hr_min
5463 %token wallclock_hour
5464 /*
5465     A wallclock_hr_min is a one, two-digit, or four-digit number.
5466     A one-digit or two-digit number constitutes a wallclock_hour.
5467     A wallclock_hour may be any of the single digits 1-9, or may
5468     be double digits, ranging from 01-12. If a wallclock_hr_min
5469     is a four-digit number, the first two digits shall be a valid
5470     wallclock_hour, while the last two represent the number of
5471     minutes, from 00-59.
5472 */
```

```
5473 %token minute
5474 /*
5475     A minute is a one or two-digit number whose values can be 0-9
5476     or 00-59.
5477 */
```

```
5478 %token day_number
5479 /*
5480     A day_number is a number in the range appropriate for the particular
5481     month and year specified by month_name and year_number, respectively.
```

```

5482         If no year_number is given, the current year is assumed if the given
5483         date and time are later this year. If no year_number is given and
5484         the date and time have already occurred this year and the month is
5485         not the current month, next year is the assumed year.
5486     */

5487     %token year_number
5488     /*
5489         A year_number is a four-digit number representing the year A.D., in
5490         which the at_job is to be run.
5491     */

5492     %token inc_number
5493     /*
5494         The inc_number is the number of times the succeeding increment
5495         period is to be added to the specified date and time.
5496     */

5497     %token timezone_name
5498     /*
5499         The name of an optional timezone suffix to the time field, in an
5500         implementation-defined format.
5501     */

5502     %token month_name
5503     /*
5504         One of the values from the mon or abmon keywords in the LC_TIME
5505         locale category.
5506     */

5507     %token day_of_week
5508     /*
5509         One of the values from the day or abday keywords in the LC_TIME
5510         locale category.
5511     */

5512     %token am_pm
5513     /*
5514         One of the values from the am_pm keyword in the LC_TIME locale
5515         category.
5516     */

5517     %start timespec
5518     %%
5519     timespec      : time
5520                   | time date
5521                   | time increment
5522                   | time date increment
5523                   | nowspec
5524                   ;

5525     nowspec      : "now"
5526                   | "now" increment
5527                   ;

5528     time         : hr24clock_hr_min
5529                   | hr24clock_hr_min timezone_name

```

```

5530         | hr24clock_hour ":" minute
5531         | hr24clock_hour ":" minute timezone_name
5532         | wallclock_hr_min am_pm
5533         | wallclock_hr_min am_pm timezone_name
5534         | wallclock_hour ":" minute am_pm
5535         | wallclock_hour ":" minute am_pm timezone_name
5536         | "noon"
5537         | "midnight"
5538         ;

5539     date      : month_name day_number
5540         | month_name day_number "," year_number
5541         | day_of_week
5542         | "today"
5543         | "tomorrow"
5544         ;

5545     increment : "+" inc_number inc_period
5546         | "next" inc_period
5547         ;

5548     inc_period : "minute" | "minutes"
5549         | "hour" | "hours"
5550         | "day" | "days"
5551         | "week" | "weeks"
5552         | "month" | "months"
5553         | "year" | "years"
5554         ;

```

5555 STDIN

5556 The standard input shall be a text file consisting of commands acceptable to the shell command
5557 language described in Chapter 2 (on page 2235). The standard input shall only be used if no **-f**
5558 *file* option is specified.

5559 INPUT FILES

5560 See the STDIN section.

5561 XSI The text files **/usr/lib/cron/at.allow** and **/usr/lib/cron/at.deny** contain user names, one per line, of
5562 users who are, respectively, authorized or denied access to the *at* and *batch* utilities.

5563 ENVIRONMENT VARIABLES

5564 The following environment variables shall affect the execution of *at*:

5565 **LANG** Provide a default value for the internationalization variables that are unset or null.
5566 If *LANG* is unset or null, the corresponding value from the implementation-
5567 defined default locale shall be used. If any of the internationalization variables
5568 contains an invalid setting, the utility shall behave as if none of the variables had
5569 been defined.

5570 **LC_ALL** If set to a non-empty string value, override the values of all the other
5571 internationalization variables.

5572 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
5573 characters (for example, single-byte as opposed to multi-byte characters in
5574 arguments and input files).

5575 **LC_MESSAGES**

5576 Determine the locale that should be used to affect the format and contents of

- 5577 diagnostic messages written to standard error and informative messages written to
5578 standard output.
- 5579 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 5580 **LC_TIME** Determine the format and contents for date and time strings written and accepted
5581 by *at*.
- 5582 **SHELL** Determine a name of a command interpreter to be used to invoke the at-job. If the
5583 variable is unset or null, *sh* shall be used. If it is set to a value other than a name for
5584 *sh*, the implementation shall do one of the following: use that shell; use *sh*; use the
5585 login shell from the user database; or any of the preceding accompanied by a
5586 warning diagnostic about which was chosen.
- 5587 **TZ** Determine the timezone. The job shall be submitted for execution at the time
5588 specified by *timespec* or *-t time* relative to the timezone specified by the *TZ*
5589 variable. If *timespec* specifies a timezone, it shall override *TZ*. If *timespec* does not
5590 specify a timezone and *TZ* is unset or null, an unspecified default timezone shall
5591 be used.
- 5592 **ASYNCHRONOUS EVENTS**
- 5593 Default.
- 5594 **STDOUT**
- 5595 When standard input is a terminal, prompts of unspecified format for each line of the user input
5596 described in the STDIN section may be written to standard output.
- 5597 In the POSIX locale, the following shall be written to the standard output for each job when jobs
5598 are listed in response to the *-l* option:
- 5599 `"%s\t%s\n", at_job_id, <date>`
- 5600 where *date* shall be equivalent in format to the output of:
- 5601 `date +"%a %b %e %T %Y"`
- 5602 The date and time written shall be adjusted so that they appear in the timezone of the user (as
5603 determined by the *TZ* variable).
- 5604 **STDERR**
- 5605 In the POSIX locale, the following shall be written to standard error when a job has been
5606 successfully submitted:
- 5607 `"job %s at %s\n", at_job_id, <date>`
- 5608 where *date* has the same format as is described in the STDOUT section. Neither this, nor warning
5609 messages concerning the selection of the command interpreter, shall be considered a diagnostic
5610 that changes the exit status.
- 5611 Diagnostic messages, if any, shall be written to standard error.
- 5612 **OUTPUT FILES**
- 5613 None.
- 5614 **EXTENDED DESCRIPTION**
- 5615 None.
- 5616 **EXIT STATUS**
- 5617 The following exit values shall be returned:
- 5618 **0** The *at* utility successfully submitted, removed, or listed a job or jobs.

5619 >0 An error occurred.

5620 CONSEQUENCES OF ERRORS

5621 The job shall not be scheduled, removed, or listed.

5622 APPLICATION USAGE

5623 The format of the *at* command line shown here is guaranteed only for the POSIX locale. Other
5624 cultures may be supported with substantially different interfaces, although implementations are
5625 encouraged to provide comparable levels of functionality.

5626 Since the commands run in a separate shell invocation, running in a separate process group with
5627 no controlling terminal, open file descriptors, traps, and priority inherited from the invoking
5628 environment are lost.

5629 Some implementations do not allow substitution of different shells using *SHELL*. System V
5630 systems, for example, have used the login shell value for the user in */etc/passwd*. To select
5631 reliably another command interpreter, the user must include it as part of the script, such as:

```
5632 $ at 1800
5633 myshell myscript
5634 job ... at ...
5635 $
```

5636 EXAMPLES

5637 1. This sequence can be used at a terminal:

```
5638 at -m 0730 tomorrow
5639 sort < file >outfile
5640 EOT
```

5641 2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a
5642 command procedure (the sequence of output redirection specifications is significant):

```
5643 at now + 1 hour <<!
5644 diff file1 file2 2>&1 >outfile | mailx mygroup
5645 !
```

5646 3. To have a job reschedule itself, *at* can be invoked from within the *at*-job. For example, this
5647 daily processing script named **my.daily** runs every day (although *crontab* is a more
5648 appropriate vehicle for such work):

```
5649 # my.daily runs every day
5650 daily processing
5651 at now tomorrow < my.daily
```

5652 4. The spacing of the three portions of the POSIX locale *timespec* is quite flexible as long as
5653 there are no ambiguities. Examples of various times and operand presentation include:

```
5654 at 0815am Jan 24
5655 at 8 :15amjan24
5656 at now "+ 1day"
5657 at 5 pm FRIday
5658 at '17
5659 utc+
5660 30minutes'
```

5661 RATIONALE

5662 The *at* utility reads from standard input the commands to be executed at a later time. It may be
5663 useful to redirect standard output and standard error within the specified commands.

5664 The **-t** *time* option was added as a new capability to support an internationalized way of
5665 specifying a time for execution of the submitted job.

5666 Early proposals added a “jobname” concept as a way of giving submitted jobs names that are
5667 meaningful to the user submitting them. The historical, system-specified *at_job_id* gives no
5668 indication of what the job is. Upon further reflection, it was decided that the benefit of this was
5669 not worth the change in historical interface. It is anticipated that considerably more
5670 sophisticated ways of controlling background or batch work will be the subject of a future
5671 version of this volume of IEEE Std. 1003.1-200x.

5672 The **-q** option historically has been an undocumented option, used mainly by the *batch* utility.

5673 The System V **-m** option was added to provide a method for informing users that an at-job had
5674 completed. Otherwise, users are only informed when output to standard error or standard
5675 output are not redirected.

5676 The behavior of *at* <now> was changed in an early proposal from being unspecified to
5677 submitting a job for potentially immediate execution. Historical BSD *at* implementations
5678 support this. Historical System V implementations give an error in that case, but a change to the
5679 System V versions should have no backwards compatibility ramifications.

5680 On BSD-based systems, a **-u** *user* option has allowed those with appropriate privileges to access
5681 the work of other users. Since this is primarily a system administration feature and is not
5682 universally implemented, it has been omitted. Similarly, a specification for the output format for
5683 user with appropriate privileges viewing the queues of other users has been omitted.

5684 The **-f** *file* option from System V is used instead of the BSD method of using the last operand as
5685 the path name. The BSD method is ambiguous—does:

5686 `at 1200 friday`

5687 mean the same thing if there is a file named **friday** in the current directory?

5688 The *at_job_id* is composed of a limited character set in historical practice, and it is mandated here
5689 to invalidate systems that might try using characters that require shell quoting or that could not
5690 be easily parsed by shell scripts.

5691 The *at* utility varies between System V and BSD systems in the way timezones are used. On
5692 System V systems, the *TZ* variable affects the at-job submission times and the times displayed
5693 for the user. On BSD systems, *TZ* is not taken into account. The BSD behavior is easily achieved
5694 with the current specification. If the user wishes to have the timezone default to that of the
5695 system, they merely need to issue the *at* command immediately following an unsetting or null
5696 assignment to *TZ*. For example:

5697 `TZ= at noon ...`

5698 gives the desired BSD result.

5699 While the *yacc*-like grammar specified in the OPERANDS section is lexically unambiguous with
5700 respect to the digit strings, a lexical analyzer would probably be written to look for and return
5701 digit strings in those cases. The parser could then check whether the digit string returned is a
5702 valid *day_number*, *year_number*, and so on, based on the context.

5703 **FUTURE DIRECTIONS**

5704 None.

5705 **SEE ALSO**5706 *batch, crontab*5707 **CHANGE HISTORY**

5708 First released in Issue 2.

5709 **Issue 4**

5710 Aligned with the ISO/IEC 9945-2:1993 standard.

5711 **Issue 6**

5712 This utility is now marked as part of the User Portability Utilities option.

5713 The following new requirements on POSIX implementations derive from alignment with the
5714 Single UNIX Specification:5715 • If **-m** is not used, the job's standard output and standard error are provided to the user by
5716 mail.5717 The effects of using the **-q** and **-t** options as defined in the IEEE P1003.2b draft standard are
5718 specified.

5719 The normative text is reworded to avoid use of the term “must” for application requirements.

5720 **NAME**

5721 awk — pattern scanning and processing language

5722 **SYNOPSIS**5723 awk [-F *ERE*][-v *assignment*] ... *program* [*argument* ...]5724 awk [-F *ERE*] -f *progfile* ... [-v *assignment*] ... [*argument* ...]5725 **DESCRIPTION**

5726 The *awk* utility shall execute programs written in the *awk* programming language, which is
 5727 specialized for textual data manipulation. An *awk* program is a sequence of patterns and
 5728 corresponding actions. When input is read that matches a pattern, the action associated with
 5729 that pattern is carried out.

5730 Input shall be interpreted as a sequence of records. By default, a record is a line, but this can be
 5731 changed by using the **RS** built-in variable. Each record of input shall be matched in turn against
 5732 each pattern in the program. For each pattern matched, the associated action shall be executed.

5733 The *awk* utility shall interpret each input record as a sequence of fields where, by default, a field
 5734 is a string of non-<blank> characters. This default white-space field delimiter can be changed by
 5735 using the **FS** built-in variable or the **-F *ERE***. The *awk* utility shall denote the first field in a
 5736 record \$1, the second \$2, and so on. The symbol \$0 shall refer to the entire record; setting any
 5737 other field causes the re-evaluation of \$0. Assigning to \$0 shall reset the values of all other fields
 5738 and the **NF** built-in variable.

5739 **OPTIONS**

5740 The *awk* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 5741 12.2, Utility Syntax Guidelines.

5742 The following options shall be supported:

5743 **-F *ERE*** Define the input field separator to be the extended regular expression *ERE*, before
 5744 any input is read; see **Regular Expressions** (on page 2375).

5745 **-f *progfile*** Specify the path name of the file *progfile* containing an *awk* program. If multiple
 5746 instances of this option are specified, the concatenation of the files specified as
 5747 *progfile* in the order specified shall be the *awk* program. The *awk* program can
 5748 alternatively be specified in the command line as a single argument.

5749 **-v *assignment***

5750 The application shall ensure that the *assignment* argument is in the same form as an
 5751 *assignment* operand. The specified variable assignment shall occur prior to
 5752 executing the *awk* program, including the actions associated with **BEGIN** patterns
 5753 (if any). Multiple occurrences of this option can be specified.

5754 **OPERANDS**

5755 The following operands shall be supported:

5756 *program* If no **-f** option is specified, the first operand to *awk* shall be the text of the *awk*
 5757 program. The application shall supply the *program* operand as a single argument to
 5758 *awk*. If the text does not end in a <newline> character, *awk* shall interpret the text
 5759 as if it did.

5760 *argument* Either of the following two types of *argument* can be intermixed:

5761 *file* A path name of a file that contains the input to be read, which is
 5762 matched against the set of patterns in the program. If no *file* operands
 5763 are specified, or if a *file* operand is '-', the standard input shall be
 5764 used.

5765 *assignment* An operand that begins with an underscore or alphabetic character
 5766 from the portable character set (see the table in the Base Definitions
 5767 volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set),
 5768 followed by a sequence of underscores, digits, and alphabetic characters
 5769 from the portable character set, followed by the '=' character, shall
 5770 specify a variable assignment rather than a path name. The
 5771 characters before the '=' represent the name of an *awk* variable; if
 5772 that name is an *awk* reserved word (see **Grammar** (on page 2384)) the
 5773 behavior is undefined. The characters following the equal sign shall
 5774 be interpreted as if they appeared in the *awk* program preceded and
 5775 followed by a double-quote (' ') character, as a **STRING** token (see
 5776 **Grammar** (on page 2384)), except that if the last character is an
 5777 unescaped backslash, it shall be interpreted as a literal backslash
 5778 rather than as the first character of the sequence "\ ". The variable
 5779 shall be assigned the value of that **STRING** token and, if
 5780 appropriate, shall be considered a *numeric string* (see **Expressions in**
 5781 **awk** (on page 2370)), the variable shall also be assigned its numeric
 5782 value. Each such variable assignment shall occur just prior to the
 5783 processing of the following *file*, if any. Thus, an assignment before
 5784 the first *file* argument shall be executed after the **BEGIN** actions (if
 5785 any), while an assignment after the last *file* argument shall occur
 5786 before the **END** actions (if any). If there are no *file* arguments,
 5787 assignments shall be executed before processing the standard input.

5788 **STDIN**

5789 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-';
 5790 see the INPUT FILES section. If the *awk* program contains no actions and no patterns, but is
 5791 otherwise a valid *awk* program, standard input and any *file* operands shall not be read and *awk*
 5792 shall exit with a return status of zero.

5793 **INPUT FILES**

5794 Input files to the *awk* program from any of the following sources shall be text files:

- 5795 • Any *file* operands or their equivalents, achieved by modifying the *awk* variables **ARGV** and
- 5796 **ARGC**
- 5797 • Standard input in the absence of any *file* operands
- 5798 • Arguments to the **getline** function

5799 Whether the variable **RS** is set to a value other than a <newline> character or not, for these files,
 5800 implementations shall support records terminated with the specified separator up to
 5801 {**LINE_MAX**} bytes and may support longer records.

5802 If **-f progfile** is specified, the application shall ensure that the files named by each of the *progfile*
 5803 option-arguments are text files containing an *awk* program.

5804 **ENVIRONMENT VARIABLES**

5805 The following environment variables shall affect the execution of *awk*:

5806 **LANG** Provide a default value for the internationalization variables that are unset or null.
 5807 If **LANG** is unset or null, the corresponding value from the implementation-
 5808 defined default locale shall be used. If any of the internationalization variables
 5809 contains an invalid setting, the utility shall behave as if none of the variables had
 5810 been defined.

- 5811 *LC_ALL* If set to a non-empty string value, override the values of all the other
5812 internationalization variables.
- 5813 *LC_COLLATE*
5814 Determine the locale for the behavior of ranges, equivalence classes, and multi-
5815 character collating elements within regular expressions and in comparisons of
5816 string values.
- 5817 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
5818 characters (for example, single-byte as opposed to multi-byte characters in
5819 arguments and input files), the behavior of character classes within regular
5820 expressions, the identification of characters as letters, and the mapping of
5821 uppercase and lowercase characters for the **toupper** and **tolower** functions.
- 5822 *LC_MESSAGES*
5823 Determine the locale that should be used to affect the format and contents of
5824 diagnostic messages written to standard error.
- 5825 *LC_NUMERIC*
5826 Determine the radix character used when interpreting numeric input, performing
5827 conversions between numeric and string values, and formatting numeric output.
5828 Regardless of locale, the period character (the decimal-point character of the
5829 POSIX locale) is the decimal-point character recognized in processing *awk*
5830 programs (including assignments in command line arguments).
- 5831 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 5832 *PATH* Determine the search path when looking for commands executed by *system(expr)*,
5833 or input and output pipes; see the Base Definitions volume of
5834 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.
- 5835 In addition, all environment variables shall be visible via the *awk* variable **ENVIRON**.
- 5836 **ASYNCHRONOUS EVENTS**
5837 Default.
- 5838 **STDOUT**
5839 The nature of the output files depends on the *awk* program.
- 5840 **STDERR**
5841 Used only for diagnostic messages.
- 5842 **OUTPUT FILES**
5843 The nature of the output files depends on the *awk* program.
- 5844 **EXTENDED DESCRIPTION**
- 5845 **Overall Program Structure**
5846 An *awk* program is composed of pairs of the form:
5847 *pattern* { *action* }
5848 Either the pattern or the action (including the enclosing brace characters) can be omitted.
5849 A missing pattern shall match any record of input, and a missing action shall be equivalent to:
5850 { *print* }
5851 Execution of the *awk* program shall start by first executing the actions associated with all **BEGIN**
5852 patterns in the order they occur in the program. Then each *file* operand (or standard input if no

5853 files were specified) shall be processed in turn by reading data from the file until a record
 5854 separator is seen (<newline> by default). Before the first reference to a field in the record is
 5855 evaluated, the record shall be split into fields, according to the rules in **Regular Expressions** (on
 5856 page 2375), using the value of **FS** that was current at the time the record was read. Each pattern
 5857 in the program then shall be evaluated in the order of occurrence, and the action associated with
 5858 each pattern that matches the current record executed. The action for a matching pattern shall be
 5859 executed before evaluating subsequent patterns. Finally, the actions associated with all **END**
 5860 patterns shall be executed in the order they occur in the program.

5861 Expressions in awk

5862 Expressions describe computations used in *patterns* and *actions*. In the following table, valid
 5863 expression operations are given in groups from highest precedence first to lowest precedence
 5864 last, with equal-precedence operators grouped between horizontal lines. In expression
 5865 evaluation, where the grammar is formally ambiguous, higher precedence operators shall be
 5866 evaluated before lower precedence operators. In this table *expr*, *expr1*, *expr2*, and *expr3* represent
 5867 any expression, while *lvalue* represents any entity that can be assigned to (that is, on the left side
 5868 of an assignment operator). The precise syntax of expressions is given in **Grammar** (on page
 5869 2384).

5870 **Table 4-1** Expressions in Decreasing Precedence in *awk*

Syntax	Name	Type of Result	Associativity
(<i>expr</i>)	Grouping	Type of <i>expr</i>	N/A
<i>\$expr</i>	Field reference	String	N/A
<i>++ lvalue</i>	Pre-increment	Numeric	N/A
<i>-- lvalue</i>	Pre-decrement	Numeric	N/A
<i>lvalue ++</i>	Post-increment	Numeric	N/A
<i>lvalue --</i>	Post-decrement	Numeric	N/A
<i>expr ^ expr</i>	Exponentiation	Numeric	Right
<i>! expr</i>	Logical not	Numeric	N/A
<i>+ expr</i>	Unary plus	Numeric	N/A
<i>- expr</i>	Unary minus	Numeric	N/A
<i>expr * expr</i>	Multiplication	Numeric	Left
<i>expr / expr</i>	Division	Numeric	Left
<i>expr % expr</i>	Modulus	Numeric	Left
<i>expr + expr</i>	Addition	Numeric	Left
<i>expr - expr</i>	Subtraction	Numeric	Left
<i>expr expr</i>	String concatenation	String	Left
<i>expr < expr</i>	Less than	Numeric	None
<i>expr <= expr</i>	Less than or equal to	Numeric	None
<i>expr != expr</i>	Not equal to	Numeric	None
<i>expr == expr</i>	Equal to	Numeric	None
<i>expr > expr</i>	Greater than	Numeric	None
<i>expr >= expr</i>	Greater than or equal to	Numeric	None
<i>expr ~ expr</i>	ERE match	Numeric	None
<i>expr !~ expr</i>	ERE non-match	Numeric	None

5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912

Syntax	Name	Type of Result	Associativity
<code>expr in array</code>	Array membership	Numeric	Left
<code>(index) in array</code>	Multi-dimension array membership	Numeric	Left
<code>expr && expr</code>	Logical AND	Numeric	Left
<code>expr expr</code>	Logical OR	Numeric	Left
<code>expr1 ? expr2 : expr3</code>	Conditional expression	Type of selected <i>expr2</i> or <i>expr3</i>	Right
<code>lvalue ^= expr</code>	Exponentiation assignment	Numeric	Right
<code>lvalue %= expr</code>	Modulus assignment	Numeric	Right
<code>lvalue *= expr</code>	Multiplication assignment	Numeric	Right
<code>lvalue /= expr</code>	Division assignment	Numeric	Right
<code>lvalue += expr</code>	Addition assignment	Numeric	Right
<code>lvalue -= expr</code>	Subtraction assignment	Numeric	Right
<code>lvalue = expr</code>	Assignment	Type of <i>expr</i>	Right

5913
5914
5915
5916

Each expression shall have either a string value, a numeric value, or both. Except as stated for specific contexts, the value of an expression shall be implicitly converted to the type needed for the context in which it is used. A string value shall be converted to a numeric value by the equivalent of the following calls to functions defined by the ISO C standard:

5917
5918

```
setlocale(LC_NUMERIC, "");
numeric_value = atof(string_value);
```

5919
5920
5921
5922
5923
5924
5925
5926
5927
5928

A numeric value that is exactly equal to the value of an integer shall be converted to a string by the equivalent of a call to the **sprintf** function (see **String Functions** (on page 2381)) with the string "%d" as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. Any other numeric value shall be converted to a string by the equivalent of a call to the **sprintf** function with the value of the variable **CONVFMT** as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. The result of the conversion is unspecified if the value of **CONVFMT** is not a floating-point format specification. This volume of IEEE Std. 1003.1-200x specifies no explicit conversions between numbers and strings. An application can force an expression to be treated as a number by adding zero to it, or can force it to be treated as a string by concatenating the null string (" ") to it.

5929

A string value shall be considered a *numeric string* if it comes from one of the following:

5930
5931
5932
5933
5934
5935
5936
5937

1. Field variables
2. Input from the *getline()* function
3. **FILENAME**
4. **ARGV** array elements
5. **ENVIRON** array elements
6. Array elements created by the *split()* function
7. A command line variable assignment
8. Variable assignment from another numeric string variable

5938
5939
5940

and after all the following conversions have been applied, the resulting string would lexically be recognized as a **NUMBER** token as described by the lexical conventions in **Grammar** (on page 2384):

- 5941 • All leading and trailing <blank>s are discarded
- 5942 • If the first non-<blank> character is '+' or '-', it is discarded
- 5943 • Changing each occurrence of the decimal point character from the current locale to a period
- 5944 If a '-' character is ignored in the preceding description, the numeric value of the *numeric string*
- 5945 shall be the negation of the numeric value of the recognized **NUMBER** token. Otherwise, the
- 5946 numeric value of the *numeric string* shall be the numeric value of the recognized **NUMBER**
- 5947 token. Whether or not a string is a *numeric string* shall be relevant only in contexts where that
- 5948 term is used in this section.
- 5949 When an expression is used in a Boolean context, if it has a numeric value, a value of zero shall
- 5950 be treated as false and any other value shall be treated as true. Otherwise, a string value of the
- 5951 null string shall be treated as false and any other value shall be treated as true. A Boolean
- 5952 context shall be one of the following:
- 5953 • The first subexpression of a conditional expression
- 5954 • An expression operated on by logical NOT, logical AND, or logical OR
- 5955 • The second expression of a **for** statement
- 5956 • The expression of an **if** statement
- 5957 • The expression of the **while** clause in either a **while** or **do...while** statement
- 5958 • An expression used as a pattern (as in Overall Program Structure)
- 5959 All arithmetic shall follow the semantics of floating-point arithmetic as specified by the ISO C
- 5960 standard.
- 5961 The value of the expression:
- 5962 `expr1 ^ expr2`
- 5963 shall be equivalent to the value returned by the ISO C standard function call:
- 5964 `pow(expr1, expr2)`
- 5965 The expression:
- 5966 `lvalue ^= expr`
- 5967 shall be equivalent to the ISO C standard expression:
- 5968 `lvalue = pow(lvalue, expr)`
- 5969 except that *lvalue* shall be evaluated only once. The value of the expression:
- 5970 `expr1 % expr2`
- 5971 shall be equivalent to the value returned by the ISO C standard function call:
- 5972 `fmod(expr1, expr2)`
- 5973 The expression:
- 5974 `lvalue %= expr`
- 5975 shall be equivalent to the ISO C standard expression:
- 5976 `lvalue = fmod(lvalue, expr)`
- 5977 except that *lvalue* shall be evaluated only once.

5978 Variables and fields shall be set by the assignment statement:

5979 *lvalue* = *expression*

5980 and the type of *expression* shall determine the resulting variable type. The assignment includes
 5981 the arithmetic assignments (" += ", " -= ", " *= ", " /= ", " %=", " ^= ", " ++", " —") all of which
 5982 shall produce a numeric result. The left-hand side of an assignment and the target of increment
 5983 and decrement operators can be one of a variable, an array with index, or a field selector.

5984 The *awk* language supplies arrays that are used for storing numbers or strings. Arrays need not
 5985 be declared. They shall initially be empty, and their sizes shall change dynamically. The
 5986 subscripts, or element identifiers, are strings, providing a type of associative array capability. An
 5987 array name followed by a subscript within square brackets can be used as an *lvalue* and thus as
 5988 an expression, as described in the grammar; see **Grammar** (on page 2384). Unsubscripted array
 5989 names can be used in only the following contexts:

- 5990 • A parameter in a function definition or function call
- 5991 • The **NAME** token following any use of the keyword **in** as specified in the grammar (see
 5992 **Grammar** (on page 2384)); if the name used in this context is not an array name, the behavior
 5993 is undefined

5994 A valid array *index* shall consist of one or more comma-separated expressions, similar to the way
 5995 in which multi-dimensional arrays are indexed in some programming languages. Because *awk*
 5996 arrays are really one-dimensional, such a comma-separated list shall be converted to a single
 5997 string by concatenating the string values of the separate expressions, each separated from the
 5998 other by the value of the **SUBSEP** variable. Thus, the following two index operations shall be
 5999 equivalent:

6000 *var*[*expr1*, *expr2*, ... *exprn*]

6001 *var*[*expr1* **SUBSEP** *expr2* **SUBSEP** ... **SUBSEP** *exprn*]

6002 The application shall ensure that a multi-dimensioned *index* used with the **in** operator is
 6003 parenthesized. The **in** operator, which tests for the existence of a particular array element, shall
 6004 not cause that element to exist. Any other reference to a nonexistent array element shall
 6005 automatically create it.

6006 Comparisons (with the '<', '<=', '!=', '==', '>', and '>=' operators) shall be made
 6007 numerically if both operands are numeric, if one is numeric and the other has a string value that
 6008 is a numeric string, or if one is numeric and the other has the uninitialized value. Otherwise,
 6009 operands shall be converted to strings as required and a string comparison shall be made using
 6010 the locale-specific collation sequence. The value of the comparison expression shall be 1 if the
 6011 relation is true, or 0 if the relation is false.

6012 **Variables and Special Variables**

6013 Variables can be used in an *awk* program by referencing them. With the exception of function
 6014 parameters (see **User-Defined Functions** (on page 2383)), they are not explicitly declared.
 6015 Function parameter names shall be local to the function; all other variable names shall be global.
 6016 The same name shall not be used as both a function parameter name and as the name of a
 6017 function or a special *awk* variable. The same name shall not be used both as a variable name with
 6018 global scope and as the name of a function. The same name shall not be used within the same
 6019 scope both as a scalar variable and as an array. Uninitialized variables, including scalar
 6020 variables, array elements, and field variables, shall have an uninitialized value. An uninitialized
 6021 value shall have both a numeric value of zero and a string value of the empty string. Evaluation
 6022 of variables with an uninitialized value, to either string or numeric, shall be determined by the
 6023 context in which they are used.

6024 Field variables shall be designated by a '\$' followed by a number or numerical expression. The
6025 effect of the field number *expression* evaluating to anything other than a non-negative integer is
6026 unspecified; uninitialized variables or string values need not be converted to numeric values in
6027 this context. New field variables can be created by assigning a value to them. References to
6028 nonexistent fields (that is, fields after \$NF), shall evaluate to the uninitialized value. Such
6029 references shall not create new fields. However, assigning to a nonexistent field (for example,
6030 \$(NF+2)=5) shall increase the value of NF; create any intervening fields with the uninitialized
6031 value; and cause the value of \$0 to be recomputed, with the fields being separated by the value
6032 of OFS. Each field variable shall have a string value or an uninitialized value when created.
6033 Field variables shall have the uninitialized value when created from \$0 using FS and the variable
6034 does not contain any characters. If appropriate, the field variable shall be considered a numeric
6035 string (see **Expressions in awk** (on page 2370)).

6036 Implementations shall support the following other special variables that are set by *awk*:

6037 **ARGC** The number of elements in the **ARGV** array.

6038 **ARGV** An array of command line arguments, excluding options and the *program*
6039 argument, numbered from zero to **ARGC**-1.

6040 The arguments in **ARGV** can be modified or added to; **ARGC** can be altered. As
6041 each input file ends, *awk* shall treat the next non-null element of **ARGV**, up to the
6042 current value of **ARGC**-1, inclusive, as the name of the next input file. Thus,
6043 setting an element of **ARGV** to null means that it shall not be treated as an input
6044 file. The name '-' indicates the standard input. If an argument matches the
6045 format of an *assignment* operand, this argument shall be treated as an assignment
6046 rather than a *file* argument.

6047 **CONVFMT** The **printf** format for converting numbers to strings (except for output statements,
6048 where **OFMT** is used); "% . 6g" by default.

6049 **ENVIRON** The variable **ENVIRON** is an array representing the value of the environment, as
6050 described in the *exec* functions defined in the System Interfaces volume of
6051 IEEE Std. 1003.1-200x. The indices of the array shall be strings consisting of the
6052 names of the environment variables, and the value of each array element is a string
6053 consisting of the value of that variable. If appropriate, the environment variable
6054 shall be considered a *numeric string* (see **Expressions in awk** (on page 2370)), the
6055 array element shall also have its numeric value.

6056 In all cases where the behavior of *awk* is affected by environment variables
6057 (including the environment of any commands that *awk* executes via the **system**
6058 function or via pipeline redirections with the **print** statement, the **printf** statement,
6059 or the **getline** function), the environment used shall be the environment at the time
6060 *awk* began executing; it is implementation-defined whether any modification of
6061 **ENVIRON** affects this environment.

6062 **FILENAME** A path name of the current input file. Inside a **BEGIN** action the value is
6063 undefined. Inside an **END** action the value is the name of the last input file
6064 processed.

6065 **FNR** The ordinal number of the current record in the current file. Inside a **BEGIN** action
6066 the value is zero. Inside an **END** action the value is the number of the last record
6067 processed in the last file processed.

6068 **FS** Input field separator regular expression; a <space> character by default.

6069 **NF** The number of fields in the current record. Inside a **BEGIN** action, the use of **NF** is
6070 undefined unless a **getline** function without a *var* argument is executed

6071 previously. Inside an **END** action, **NF** retains the value it had for the last record
 6072 read, unless a subsequent redirected, **getline** function without a *var* argument is
 6073 performed prior to entering the **END** action.

6074 **NR** The ordinal number of the current record from the start of input. Inside a **BEGIN**
 6075 action the value is zero. Inside an **END** action the value is the number of the last
 6076 record processed.

6077 **OFMT** The **printf** format for converting numbers to strings in output statements (see
 6078 **Output Statements** (on page 2379)); "%.*g*" by default. The result of the
 6079 conversion is unspecified if the value of **OFMT** is not a floating-point format
 6080 specification.

6081 **OFS** The **print** statement output field separation; <space> by default.

6082 **ORS** The **print** statement output record separator; a <newline> character by default.

6083 **RLENGTH** The length of the string matched by the **match** function.

6084 **RS** The first character of the string value of **RS** is the input record separator; a
 6085 <newline> character by default. If **RS** contains more than one character, the results
 6086 are unspecified. If **RS** is null, then records are separated by sequences of one or
 6087 more blank lines, leading or trailing blank lines do not result in empty records at
 6088 the beginning or end of the input, and a <newline> character is always a field
 6089 separator, no matter what the value of **FS** is.

6090 **RSTART** The starting position of the string matched by the **match** function, numbering from
 6091 1. This is always equivalent to the return value of the **match** function.

6092 **SUBSEP** The subscript separator string for multi-dimensional arrays; the default value is
 6093 implementation-defined.

6094 **Regular Expressions**

6095 The *awk* utility shall make use of the extended regular expression notation (see the Base
 6096 Definitions volume of IEEE Std. 1003.1-200x, Section 9.4, Extended Regular Expressions) except
 6097 that it shall allow the use of C-language conventions for escaping special characters within the
 6098 EREs, as specified in the table in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5,
 6099 File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v') and the following
 6100 table; these escape sequences shall be recognized both inside and outside bracket expressions.
 6101 Note that records need not be separated by <newline> characters and string constants can
 6102 contain <newline> characters, so even the "\n" sequence is valid in *awk* EREs. Using a slash
 6103 character within an ERE requires the escaping shown in the following table.

6104

Table 4-2 Escape Sequences in *awk*

6105

6106

6107

6108

6109

6110

6111

6112

6113

6114

6115

6116

6117

6118

6119

6120

6121

6122

6123

6124

6125

Escape Sequence	Description	Meaning
\ "	Backslash quotation-mark	Quotation-mark character
\ /	Backslash slash	Slash character
\ ddd	A backslash character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.	The character whose encoding is represented by the one, two, or three-digit octal integer. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation-defined. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading '\ ' for each byte.
\ c	A backslash character followed by any character not described in this table or in the table in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v')	Undefined

6126

6127

6128

6129

6130

6131

6132

6133

6134

6135

6136

6137

6138

6139

6140

A regular expression can be matched against a specific field or string by using one of the two regular expression matching operators, '~' and '!~'. These operators shall interpret their right-hand operand as a regular expression and their left-hand operand as a string. If the regular expression matches the string, the '~' expression shall evaluate to a value of 1, and the '!~' expression shall evaluate to a value of 0. (The regular expression matching operation is as defined by the term *matched* in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.1, Regular Expression Definitions, where a match occurs on any part of the string unless the regular expression is limited with the circumflex or dollar sign special characters.) If the regular expression does not match the string, the '~' expression shall evaluate to a value of 0, and the '!~' expression shall evaluate to a value of 1. If the right-hand operand is any expression other than the lexical token **ERE**, the string value of the expression shall be interpreted as an extended regular expression, including the escape conventions described above. Note that these same escape conventions shall also be applied in determining the value of a string literal (the lexical token **STRING**), and thus shall be applied a second time when a string literal is used in this context.

6141

6142

6143

When an **ERE** token appears as an expression in any context other than as the right-hand of the '~' or '!~' operator or as one of the built-in function arguments described below, the value of the resulting expression shall be the equivalent of:

6144

```
$0 ~ /ere/
```

6145

6146

6147

6148

The *ere* argument to the **gsub**, **match**, **sub** functions, and the *fs* argument to the **split** function (see **String Functions** (on page 2381)) shall be interpreted as extended regular expressions. These can be either **ERE** tokens or arbitrary expressions, and shall be interpreted in the same manner as the right-hand side of the '~' or '!~' operator.

6149

6150

6151

An extended regular expression can be used to separate fields by using the **-F ERE** option or by assigning a string containing the expression to the built-in variable **FS**. The default value of the **FS** variable shall be a single <space> character. The following describes **FS** behavior:

- 6152 1. If **FS** is a null string, the behavior is unspecified.
- 6153 2. If **FS** is a single character:
- 6154 a. If **FS** is the <space> character, skip leading and trailing <blank> characters; fields
- 6155 shall be delimited by sets of one or more <blank> characters.
- 6156 b. Otherwise, if **FS** is any other character *c*, fields shall be delimited by each single
- 6157 occurrence of *c*.
- 6158 3. Otherwise, the string value of **FS** shall be considered to be an extended regular expression.
- 6159 Each occurrence of a sequence matching the extended regular expression shall delimit
- 6160 fields.

6161 Except for the '*~*' and '!~' operators, and in the **gsub**, **match**, **split**, and **sub** built-in functions,

6162 ERE matching shall be based on input records; that is, record separator characters (the first

6163 character of the value of the variable **RS**, <newline> by default) cannot be embedded in the

6164 expression, and no expression shall match the record separator character. If the record separator

6165 is not <newline>, <newline> characters embedded in the expression can be matched. For the

6166 '*~*' and '!~' operators, and in those four built-in functions, ERE matching shall be based on

6167 text strings; that is, any character (including <newline> and the record separator) can be

6168 embedded in the pattern, and an appropriate pattern shall match any character. However, in all

6169 *awk* ERE matching, the use of one or more NUL characters in the pattern, input record, or text

6170 string produces undefined results.

6171 **Patterns**

6172 A *pattern* is any valid *expression*, a range specified by two expressions separated by comma, or

6173 one of the two special patterns **BEGIN** or **END**.

6174 **Special Patterns**

6175 The *awk* utility shall recognize two special patterns, **BEGIN** and **END**. Each **BEGIN** pattern

6176 shall be matched once and its associated action executed before the first record of input is read

6177 (except possibly by use of the **getline** function—see **Input/Output and General Functions** (on

6178 page 2382)—in a prior **BEGIN** action) and before command line assignment is done. Each **END**

6179 pattern shall be matched once and its associated action executed after the last record of input has

6180 been read. These two patterns shall have associated actions.

6181 **BEGIN** and **END** shall not combine with other patterns. Multiple **BEGIN** and **END** patterns

6182 shall be allowed. The actions associated with the **BEGIN** patterns shall be executed in the order

6183 specified in the program, as are the **END** actions. An **END** pattern can precede a **BEGIN** pattern

6184 in a program.

6185 If an *awk* program consists of only actions with the pattern **BEGIN**, and the **BEGIN** action

6186 contains no **getline** function, *awk* shall exit without reading its input when the last statement in

6187 the last **BEGIN** action is executed. If an *awk* program consists of only actions with the pattern

6188 **END** or only actions with the patterns **BEGIN** and **END**, the input shall be read before the

6189 statements in the **END** actions are executed.

6190 **Expression Patterns**

6191 An expression pattern shall be evaluated as if it were an expression in a Boolean context. If the
6192 result is true, the pattern shall be considered to match, and the associated action (if any) shall be
6193 executed. If the result is false, the action shall not be executed.

6194 **Pattern Ranges**

6195 A pattern range consists of two expressions separated by a comma; in this case, the action shall
6196 be performed for all records between a match of the first expression and the following match of
6197 the second expression, inclusive. At this point, the pattern range can be repeated starting at
6198 input records subsequent to the end of the matched range.

6199 **Actions**

6200 An action is a sequence of statements as shown in the grammar in **Grammar** (on page 2384).
6201 Any single statement can be replaced by a statement list enclosed in braces. The application shall
6202 ensure that statements in a statement list are separated by <newline> characters or semicolons,
6203 and are executed sequentially in the order that they appear.

6204 The *expression* acting as the conditional in an **if** statement shall be evaluated and if it is non-zero
6205 or non-null, the following *statement* shall be executed; otherwise, if **else** is present, the statement
6206 following the **else** shall be executed.

6207 The **if**, **while**, **do...while**, **for**, **break**, and **continue** statements are based on the ISO C standard,
6208 except that the Boolean expressions shall be treated as described in **Expressions in awk** (on page
6209 2370), and except in the case of:

6210 `for (variable in array)`

6211 which shall iterate, assigning each *index of array* to *variable* in an unspecified order. The results of
6212 adding new elements to *array* within such a **for** loop are undefined. If a **break** or **continue**
6213 statement occurs outside of a loop, the behavior is undefined.

6214 The **delete** statement shall remove an individual array element. Thus, the following code deletes
6215 an entire array:

```
6216 for (index in array)
6217     delete array[index]
```

6218 The **next** statement shall cause all further processing of the current input record to be
6219 abandoned. The behavior is undefined if a **next** statement appears or is invoked in a **BEGIN** or
6220 **END** action.

6221 The **exit** statement shall invoke all **END** actions in the order in which they occur in the program
6222 source and then terminate the program without reading further input. An **exit** statement inside
6223 an **END** action shall terminate the program without further execution of **END** actions. If an
6224 expression is specified in an **exit** statement, its numeric value shall be the exit status of *awk*,
6225 unless subsequent errors are encountered or a subsequent **exit** statement with an expression is
6226 executed.

6227 **Output Statements**

6228 Both **print** and **printf** statements shall write to standard output by default. The output shall be
 6229 written to the location specified by *output_redirection* if one is supplied, as follows:

```
6230 > expression
6231 >> expression
6232 | expression
```

6233 In all cases, the *expression* shall be evaluated to produce a string that is used as a path name into
 6234 which to write (for '>' or ">>") or as a command to be executed (for '|'). Using the first two
 6235 forms, if the file of that name is not currently open, it shall be opened, creating it if necessary and
 6236 using the first form, truncating the file. The output then shall be appended to the file. As long as
 6237 the file remains open, subsequent calls in which *expression* evaluates to the same string value
 6238 shall simply append output to the file. The file remains open until the **close** function (see
 6239 **Input/Output and General Functions** (on page 2382)) is called with an expression that evaluates
 6240 to the same string value.

6241 The third form shall write output onto a stream piped to the input of a command. The stream
 6242 shall be created if no stream is currently open with the value of *expression* as its command name.
 6243 The stream created shall be equivalent to one created by a call to the *popen()* function defined in
 6244 the System Interfaces volume of IEEE Std. 1003.1-200x with the value of *expression* as the
 6245 *command* argument and a value of *w* as the *mode* argument. As long as the stream remains open,
 6246 subsequent calls in which *expression* evaluates to the same string value shall write output to the
 6247 existing stream. The stream shall remain open until the **close** function (see **Input/Output and**
 6248 **General Functions** (on page 2382)) is called with an expression that evaluates to the same string
 6249 value. At that time, the stream shall be closed as if by a call to the *pclose()* function defined in
 6250 the System Interfaces volume of IEEE Std. 1003.1-200x.

6251 As described in detail by the grammar in **Grammar** (on page 2384), these output statements shall
 6252 take a comma-separated list of *expressions* referred to in the grammar by the non-terminal
 6253 symbols **expr_list**, **print_expr_list**, or **print_expr_list_opt**. This list is referred to here as the
 6254 *expression list*, and each member is referred to as an *expression argument*.

6255 The **print** statement shall write the value of each expression argument onto the indicated output
 6256 stream separated by the current output field separator (see variable **OFS** above), and terminated
 6257 by the output record separator (see variable **ORS** above). All expression arguments shall be
 6258 taken as strings, being converted if necessary; this conversion shall be as described in
 6259 **Expressions in awk** (on page 2370), with the exception that the **printf** format in **OFMT** shall be
 6260 used instead of the value in **CONVFMT**. An empty expression list shall stand for the whole
 6261 input record (\$0).

6262 The **printf** statement shall produce output based on a notation similar to the File Format
 6263 Notation used to describe file formats in this volume of IEEE Std. 1003.1-200x (see the Base
 6264 Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation). Output shall be
 6265 produced as specified with the first expression argument as the string *format* and subsequent
 6266 expression arguments as the strings *arg1* to *argn*, inclusive, with the following exceptions:

- 6267 1. The *format* shall be an actual character string rather than a graphical representation.
 6268 Therefore, it cannot contain empty character positions. The <space> character in the *format*
 6269 string, in any context other than a *flag* of a conversion specification, shall be treated as an
 6270 ordinary character that is copied to the output.
- 6271 2. If the character set contains a 'Δ' character and that character appears in the *format* string,
 6272 it shall be treated as an ordinary character that is copied to the output.

- 6273 3. The *escape sequences* beginning with a backslash character shall be treated as sequences of
 6274 ordinary characters that are copied to the output. Note that these same sequences shall be
 6275 interpreted lexically by *awk* when they appear in literal strings, but they shall not be
 6276 treated specially by the **printf** statement.
- 6277 4. A *field width* or *precision* can be specified as the '*' character instead of a digit string. In
 6278 this case the next argument from the expression list shall be fetched and its numeric value
 6279 taken as the field width or precision.
- 6280 5. The implementation shall not precede or follow output from the *d* or *u* conversion
 6281 specifications with <blank> characters not specified by the *format* string.
- 6282 6. The implementation shall not precede output from the *o* conversion specification with
 6283 leading zeros not specified by the *format* string.
- 6284 7. For the *c* conversion specification: if the argument has a numeric value, the character
 6285 whose encoding is that value shall be output. If the value is zero or is not the encoding of
 6286 any character in the character set, the behavior is undefined. If the argument does not have
 6287 a numeric value, the first character of the string value shall be output; if the string does not
 6288 contain any characters, the behavior is undefined.
- 6289 8. For each conversion specification that consumes an argument, the next expression
 6290 argument shall be evaluated. With the exception of the *c* conversion, the value shall be
 6291 converted (according to the rules specified in **Expressions in awk** (on page 2370)) to the
 6292 appropriate type for the conversion specification.
- 6293 9. If there are insufficient expression arguments to satisfy all the conversion specifications in
 6294 the *format* string, the behavior is undefined.
- 6295 10. If any character sequence in the *format* string begins with a '%' character, but does not
 6296 form a valid conversion specification, the behavior is unspecified.

6297 Both **print** and **printf** can output at least {LINE_MAX} bytes.

6298 **Functions**

6299 The *awk* language has a variety of built-in functions: arithmetic, string, input/output, and
 6300 general.

6301 **Arithmetic Functions**

6302 The arithmetic functions, except for **int**, shall be based on the ISO C standard. The behavior is
 6303 undefined in cases where the ISO C standard specifies that an error be returned or that the
 6304 behavior is undefined. Although the grammar (see **Grammar** (on page 2384)) permits built-in
 6305 functions to appear with no arguments or parentheses, unless the argument or parentheses are
 6306 indicated as optional in the following list (by displaying them within the "[]" brackets), such
 6307 use is undefined.

- 6308 **atan2**(*y,x*) Return arctangent of y/x in radians in the range $-\{\pi\}$ to $\{\pi\}$.
- 6309 **cos**(*x*) Return cosine of *x*, where *x* is in radians.
- 6310 **sin**(*x*) Return sine of *x*, where *x* is in radians.
- 6311 **exp**(*x*) Return the exponential function of *x*.
- 6312 **log**(*x*) Return the natural logarithm of *x*.
- 6313 **sqrt**(*x*) Return the square root of *x*.

- 6314 **int**(*x*) Truncate its argument to an integer. It shall be truncated toward 0 when $x > 0$.
- 6315 **rand**() Return a random number n , such that $0 \leq n < 1$.
- 6316 **srand**([*expr*]) Set the seed value for *rand* to *expr* or use the time of day if *expr* is omitted. The
6317 previous seed value shall be returned.

6318 **String Functions**

6319 The string functions in the following list shall be supported. Although the grammar (see
6320 **Grammar** (on page 2384)) permits built-in functions to appear with no arguments or
6321 parentheses, unless the argument or parentheses are indicated as optional in the following list
6322 (by displaying them within the "[]" brackets), such use is undefined.

6323 **gsub**(*ere, repl*[, *in*])

6324 Behave like **sub** (see below), except that it shall replace all occurrences of the
6325 regular expression (like the *ed* utility global substitute) in \$0 or in the *in* argument,
6326 when specified.

6327 **index**(*s, t*) Return the position, in characters, numbering from 1, in string *s* where string *t* first
6328 occurs, or zero if it does not occur at all.

6329 **length**([*(s)*]) Return the length, in characters, of its argument taken as a string, or of the whole
6330 record, \$0, if there is no argument.

6331 **match**(*s, ere*) Return the position, in characters, numbering from 1, in string *s* where the
6332 extended regular expression *ere* occurs, or zero if it does not occur at all. RSTART
6333 shall be set to the starting position (which is the same as the returned value), zero
6334 if no match is found; RLENGTH shall be set to the length of the matched string, -1
6335 if no match is found.

6336 **split**(*s, a*[, *fs*])

6337 Split the string *s* into array elements $a[1]$, $a[2]$, ..., $a[n]$, and return n . All elements
6338 of the array shall be deleted before the split is performed. The separation shall be
6339 done with the ERE *fs* or with the field separator **FS** if *fs* is not given. Each array
6340 element shall have a string value when created and, if appropriate, the array
6341 element shall be considered a numeric string (see **Expressions in awk** (on page
6342 2370)). The effect of a null string as the value of *fs* is unspecified.

6343 **sprintf**(*fmt, expr, expr, ...*)

6344 Format the expressions according to the **printf** format given by *fmt* and return the
6345 resulting string.

6346 **sub**(*ere, repl*[, *in*])

6347 Substitute the string *repl* in place of the first instance of the extended regular
6348 expression *ERE* in string *in* and return the number of substitutions. An ampersand
6349 ('&') appearing in the string *repl* shall be replaced by the string from *in* that
6350 matches the ERE. An ampersand preceded with a backslash ('\&') shall be
6351 interpreted as the literal ampersand character. Any other occurrence of a backslash
6352 (for example, preceding any other character) shall be treated as a literal backslash
6353 character. Note that if *repl* is a string literal (the lexical token **STRING**; see
6354 **Grammar** (on page 2384)), the handling of the ampersand character occurs after
6355 any lexical processing, including any lexical backslash escape sequence processing.
6356 If *in* is specified and it is not an *lvalue* (see **Expressions in awk** (on page 2370)), the
6357 behavior is undefined. If *in* is omitted, *awk* shall use the current record (\$0) in its
6358 place.

- 6359 **substr**(*s*, *m* [, *n*])
 6360 Return the at most *n*-character substring of *s* that begins at position *m*, numbering
 6361 from 1. If *n* is missing, or if *n* specifies more characters than are left in the string,
 6362 the length of the substring shall be limited by the length of the string *s*.
- 6363 **tolower**(*s*) Return a string based on the string *s*. Each character in *s* that is an uppercase letter
 6364 specified to have a **tolower** mapping by the *LC_CTYPE* category of the current
 6365 locale shall be replaced in the returned string by the lowercase letter specified by
 6366 the mapping. Other characters in *s* shall be unchanged in the returned string.
- 6367 **toupper**(*s*) Return a string based on the string *s*. Each character in *s* that is a lowercase letter
 6368 specified to have a **toupper** mapping by the *LC_CTYPE* category of the current
 6369 locale is replaced in the returned string by the uppercase letter specified by the
 6370 mapping. Other characters in *s* are unchanged in the returned string.

6371 All of the preceding functions that take *ERE* as a parameter expect a pattern or a string valued
 6372 expression that is a regular expression as defined in **Regular Expressions** (on page 2375).

6373 **Input/Output and General Functions**

6374 The input/output and general functions are:

- 6375 **close**(*expression*)
 6376 Close the file or pipe opened by a **print** or **printf** statement or a call to **getline** with
 6377 the same string-valued *expression*. The limit on the number of open *expression*
 6378 arguments is implementation-defined. If the close was successful, the function
 6379 shall return zero; otherwise, it shall return non-zero.

- 6380 *expression* / **getline** [*var*]
 6381 Read a record of input from a stream piped from the output of a command. The
 6382 stream shall be created if no stream is currently open with the value of *expression* as
 6383 its command name. The stream created shall be equivalent to one created by a call
 6384 to the *popen*() function with the value of *expression* as the *command* argument and a
 6385 value of *r* as the *mode* argument. As long as the stream remains open, subsequent
 6386 calls in which *expression* evaluates to the same string value shall read subsequent
 6387 records from the stream. The stream shall remain open until the **close** function is
 6388 called with an *expression* that evaluates to the same string value. At that time, the
 6389 stream shall be closed as if by a call to the *pclose*() function. If *var* is missing, \$0 and
 6390 **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered
 6391 a numeric string (see **Expressions in awk** (on page 2370)).

6392 The **getline** operator can form ambiguous constructs when there are
 6393 unparenthesized operators (including concatenate) to the left of the ' | ' (to the
 6394 beginning of the expression containing **getline**). In the context of the '\$'
 6395 operator, ' | ' shall behave as if it had a lower precedence than '\$'. The result of
 6396 evaluating other operators is unspecified, and portable applications shall
 6397 parenthesize properly all such usages.

- 6398 **getline** Set \$0 to the next input record from the current input file. This form of **getline** shall
 6399 set the **NF**, **NR**, and **FNR** variables.

- 6400 **getline var** Set variable *var* to the next input record from the current input file and, if
 6401 appropriate, *var* shall be considered a numeric string (see **Expressions in awk** (on
 6402 page 2370)). This form of **getline** shall set the **FNR** and **NR** variables.

- 6403 **getline** [*var*] < *expression*
 6404 Read the next record of input from a named file. The *expression* shall be evaluated

6405 to produce a string that is used as a path name. If the file of that name is not
 6406 currently open, it shall be opened. As long as the stream remains open, subsequent
 6407 calls in which *expression* evaluates to the same string value shall read subsequent
 6408 records from the file. The file shall remain open until the **close** function is called
 6409 with an expression that evaluates to the same string value. If *var* is missing, \$0 and
 6410 **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered
 6411 a numeric string (see **Expressions in awk** (on page 2370)).

6412 The **getline** operator can form ambiguous constructs when there are
 6413 unparenthesized binary operators (including concatenate) to the right of the '<'
 6414 (up to the end of the expression containing the **getline**). The result of evaluating
 6415 such a construct is unspecified, and portable applications shall parenthesize
 6416 properly all such usages.

6417 **system**(*expression*)

6418 Execute the command given by *expression* in a manner equivalent to the *system*()
 6419 function defined in the System Interfaces volume of IEEE Std. 1003.1-200x and
 6420 return the exit status of the command.

6421 All forms of **getline** shall return 1 for successful input, zero for end-of-file, and -1 for an error.

6422 Where strings are used as the name of a file or pipeline, the application shall ensure that the
 6423 strings are textually identical. The terminology “same string value” implies that “equivalent
 6424 strings”, even those that differ only by <space> characters, represent different files.

6425 User-Defined Functions

6426 The *awk* language also provides user-defined functions. Such functions can be defined as:

```
6427 function name([parameter, ...]) { statements }
```

6428 A function can be referred to anywhere in an *awk* program; in particular, its use can precede its
 6429 definition. The scope of a function is global.

6430 Function parameters, if present, can be either scalars or arrays; the behavior is undefined if an
 6431 array name is passed as a parameter that the function uses as a scalar, or if a scalar expression is
 6432 passed as a parameter that the function uses as an array. Function parameters shall be passed by
 6433 value if scalar and by reference if array name.

6434 The number of parameters in the function definition need not match the number of parameters
 6435 in the function call. Excess formal parameters can be used as local variables. If fewer arguments
 6436 are supplied in a function call than are in the function definition, the extra parameters that are
 6437 used in the function body as scalars shall evaluate to the uninitialized value until they are
 6438 otherwise initialized, and the extra parameters that are used in the function body as arrays shall
 6439 be treated as uninitialized arrays where each element evaluates to the uninitialized value until
 6440 otherwise initialized.

6441 When invoking a function, no white space can be placed between the function name and the
 6442 opening parenthesis. Function calls can be nested and recursive calls can be made upon
 6443 functions. Upon return from any nested or recursive function call, the values of all of the calling
 6444 function's parameters shall be unchanged, except for array parameters passed by reference. The
 6445 **return** statement can be used to return a value. If a **return** statement appears outside of a
 6446 function definition, the behavior is undefined.

6447 In the function definition, <newline> characters shall be optional before the opening brace and
 6448 after the closing brace. Function definitions can appear anywhere in the program where a
 6449 *pattern-action* pair is allowed.

```

6450      Grammar
6451      The grammar in this section and the lexical conventions in the following section shall together
6452      describe the syntax for awk programs. The general conventions for this style of grammar are
6453      described in Section 1.10 (on page 2223). A valid program can be represented as the non-
6454      terminal symbol program in the grammar. This formal syntax shall take precedence over the
6455      preceding text syntax description.
6456      %token NAME NUMBER STRING ERE
6457      %token FUNC_NAME /* Name followed by '(' without white space. */
6458      /* Keywords */
6459      %token Begin End
6460      /* 'BEGIN' 'END' */
6461      %token Break Continue Delete Do Else
6462      /* 'break' 'continue' 'delete' 'do' 'else' */
6463      %token Exit For Function If In
6464      /* 'exit' 'for' 'function' 'if' 'in' */
6465      %token Next Print Printf Return While
6466      /* 'next' 'print' 'printf' 'return' 'while' */
6467      /* Reserved function names */
6468      %token BUILTIN_FUNC_NAME
6469      /* One token for the following:
6470      * atan2 cos sin exp log sqrt int rand srand
6471      * gsub index length match split sprintf sub
6472      * substr tolower toupper close system
6473      */
6474      %token GETLINE
6475      /* Syntactically different from other built-ins. */
6476      /* Two-character tokens. */
6477      %token ADD_ASSIGN SUB_ASSIGN MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN POW_ASSIGN
6478      /* '+=' '-=' '*=' '/=' '%=' '^=' */
6479      %token OR AND NO_MATCH EQ LE GE NE INCR DECR APPEND
6480      /* '|'| '&&' '!~' '==' '<=' '>=' '!=' '++' '—' '>>' */
6481      /* One-character tokens. */
6482      %token '{' '}' '(' ')' '[' ']' ',' ';' NEWLINE
6483      %token '+' '-' '*' '%' '^' '!' '>' '<' '|' '?' ':' '~' '$' '='
6484      %start program
6485      %%
6486      program : item_list
6487              | actionless_item_list
6488              ;
6489      item_list : newline_opt
6490                | actionless_item_list item terminator
6491                | item_list item terminator
6492                | item_list action terminator
6493                ;

```

```

6494     actionless_item_list : item_list           pattern terminator
6495         | actionless_item_list pattern terminator
6496         ;

6497     item                   : pattern action
6498         | Function NAME     '(' param_list_opt ')'
6499         | newline_opt action
6500         | Function FUNC_NAME '(' param_list_opt ')'
6501         | newline_opt action
6502         ;

6503     param_list_opt        : /* empty */
6504         | param_list
6505         ;

6506     param_list            : NAME
6507         | param_list ',' NAME
6508         ;

6509     pattern               : Begin
6510         | End
6511         | expr
6512         | expr ',' newline_opt expr
6513         ;

6514     action                : '{' newline_opt
6515         | '{' newline_opt terminated_statement_list
6516         | '{' newline_opt unterminated_statement_list
6517         ;

6518     terminator            : terminator ';'
6519         | terminator NEWLINE
6520         |
6521         | NEWLINE
6522         ;

6523     terminated_statement_list : terminated_statement
6524         | terminated_statement_list terminated_statement
6525         ;

6526     unterminated_statement_list : unterminated_statement
6527         | terminated_statement_list unterminated_statement
6528         ;

6529     terminated_statement  : action newline_opt
6530         | If '(' expr ')' newline_opt terminated_statement
6531         | If '(' expr ')' newline_opt terminated_statement
6532         | Else newline_opt terminated_statement
6533         | While '(' expr ')' newline_opt terminated_statement
6534         | For '(' simple_statement_opt ';'
6535         |   expr_opt ';' simple_statement_opt ')' newline_opt
6536         |   terminated_statement
6537         | For '(' NAME In NAME ')' newline_opt
6538         |   terminated_statement
6539         | ';' newline_opt
6540         | terminatable_statement NEWLINE newline_opt
6541         | terminatable_statement ';'      newline_opt

```

```

6542         ;
6543     unterminated_statement : terminatable_statement
6544         | If '(' expr ')' newline_opt unterminated_statement
6545         | If '(' expr ')' newline_opt terminated_statement
6546         | Else newline_opt unterminated_statement
6547         | While '(' expr ')' newline_opt unterminated_statement
6548         | For '(' simple_statement_opt ';'
6549         |   expr_opt ';' simple_statement_opt ')' newline_opt
6550         |   unterminated_statement
6551         | For '(' NAME In NAME ')' newline_opt
6552         |   unterminated_statement
6553         ;
6554     terminatable_statement : simple_statement
6555         | Break
6556         | Continue
6557         | Next
6558         | Exit expr_opt
6559         | Return expr_opt
6560         | Do newline_opt terminated_statement While '(' expr ')'
6561         ;
6562     simple_statement_opt : /* empty */
6563         | simple_statement
6564         ;
6565     simple_statement : Delete NAME '[' expr_list ']'
6566         | expr
6567         | print_statement
6568         ;
6569     print_statement : simple_print_statement
6570         | simple_print_statement output_redirection
6571         ;
6572     simple_print_statement : Print print_expr_list_opt
6573         | Print '(' multiple_expr_list ')'
6574         | Printf print_expr_list
6575         | Printf '(' multiple_expr_list ')'
6576         ;
6577     output_redirection : '>' expr
6578         | APPEND expr
6579         | '|' expr
6580         ;
6581     expr_list_opt : /* empty */
6582         | expr_list
6583         ;
6584     expr_list : expr
6585         | multiple_expr_list
6586         ;
6587     multiple_expr_list : expr ',' newline_opt expr
6588         | multiple_expr_list ',' newline_opt expr

```



```

6589          ;
6590  expr_opt   : /* empty */
6591             | expr
6592             ;
6593  expr       : unary_expr
6594             | non_unary_expr
6595             ;
6596  unary_expr : '+' expr
6597             | '-' expr
6598             | unary_expr '^'      expr
6599             | unary_expr '*'      expr
6600             | unary_expr '/'      expr
6601             | unary_expr '%'      expr
6602             | unary_expr '+'      expr
6603             | unary_expr '-'      expr
6604             | unary_expr          non_unary_expr
6605             | unary_expr '<'      expr
6606             | unary_expr LE      expr
6607             | unary_expr NE      expr
6608             | unary_expr EQ      expr
6609             | unary_expr '>'      expr
6610             | unary_expr GE      expr
6611             | unary_expr '~'      expr
6612             | unary_expr NO_MATCH expr
6613             | unary_expr In NAME
6614             | unary_expr AND newline_opt expr
6615             | unary_expr OR  newline_opt expr
6616             | unary_expr '?' expr ':' expr
6617             | unary_input_function
6618             ;
6619  non_unary_expr : '(' expr ')'
6620                | '!' expr
6621                | non_unary_expr '^'      expr
6622                | non_unary_expr '*'      expr
6623                | non_unary_expr '/'      expr
6624                | non_unary_expr '%'      expr
6625                | non_unary_expr '+'      expr
6626                | non_unary_expr '-'      expr
6627                | non_unary_expr          non_unary_expr
6628                | non_unary_expr '<'      expr
6629                | non_unary_expr LE      expr
6630                | non_unary_expr NE      expr
6631                | non_unary_expr EQ      expr
6632                | non_unary_expr '>'      expr
6633                | non_unary_expr GE      expr
6634                | non_unary_expr '~'      expr
6635                | non_unary_expr NO_MATCH expr
6636                | non_unary_expr In NAME
6637                | '(' multiple_expr_list ')' In NAME
6638                | non_unary_expr AND newline_opt expr

```

```

6639         | non_unary_expr OR newline_opt expr
6640         | non_unary_expr '?' expr ':' expr
6641         | NUMBER
6642         | STRING
6643         | lvalue
6644         | ERE
6645         | lvalue INCR
6646         | lvalue DECR
6647         | INCR lvalue
6648         | DECR lvalue
6649         | lvalue POW_ASSIGN expr
6650         | lvalue MOD_ASSIGN expr
6651         | lvalue MUL_ASSIGN expr
6652         | lvalue DIV_ASSIGN expr
6653         | lvalue ADD_ASSIGN expr
6654         | lvalue SUB_ASSIGN expr
6655         | lvalue '=' expr
6656         | FUNC_NAME '(' expr_list_opt ')'
6657         | /* no white space allowed before '(' */
6658         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
6659         | BUILTIN_FUNC_NAME
6660         | non_unary_input_function
6661         ;

6662     print_expr_list_opt : /* empty */
6663         | print_expr_list
6664         ;

6665     print_expr_list    : print_expr
6666         | print_expr_list ',' newline_opt print_expr
6667         ;

6668     print_expr        : unary_print_expr
6669         | non_unary_print_expr
6670         ;

6671     unary_print_expr  : '+' print_expr
6672         | '-' print_expr
6673         | unary_print_expr '^'      print_expr
6674         | unary_print_expr '*'      print_expr
6675         | unary_print_expr '/'      print_expr
6676         | unary_print_expr '%'      print_expr
6677         | unary_print_expr '+'      print_expr
6678         | unary_print_expr '-'      print_expr
6679         | unary_print_expr          non_unary_print_expr
6680         | unary_print_expr '~'      print_expr
6681         | unary_print_expr NO_MATCH print_expr
6682         | unary_print_expr In NAME
6683         | unary_print_expr AND newline_opt print_expr
6684         | unary_print_expr OR  newline_opt print_expr
6685         | unary_print_expr '?' print_expr ':' print_expr
6686         ;

6687     non_unary_print_expr : '(' expr ')'
6688         | '!' print_expr

```

```

6689         | non_unary_print_expr '^'      print_expr
6690         | non_unary_print_expr '*'      print_expr
6691         | non_unary_print_expr '/'    print_expr
6692         | non_unary_print_expr '%'    print_expr
6693         | non_unary_print_expr '+'    print_expr
6694         | non_unary_print_expr '-'    print_expr
6695         | non_unary_print_expr      non_unary_print_expr
6696         | non_unary_print_expr '~'    print_expr
6697         | non_unary_print_expr NO_MATCH print_expr
6698         | non_unary_print_expr In NAME
6699         | '(' multiple_expr_list ')' In NAME
6700         | non_unary_print_expr AND newline_opt print_expr
6701         | non_unary_print_expr OR  newline_opt print_expr
6702         | non_unary_print_expr '?' print_expr ':' print_expr
6703         | NUMBER
6704         | STRING
6705         | lvalue
6706         | ERE
6707         | lvalue INCR
6708         | lvalue DECR
6709         | INCR lvalue
6710         | DECR lvalue
6711         | lvalue POW_ASSIGN print_expr
6712         | lvalue MOD_ASSIGN print_expr
6713         | lvalue MUL_ASSIGN print_expr
6714         | lvalue DIV_ASSIGN print_expr
6715         | lvalue ADD_ASSIGN print_expr
6716         | lvalue SUB_ASSIGN print_expr
6717         | lvalue '=' print_expr
6718         | FUNC_NAME '(' expr_list_opt ')'
6719         | /* no white space allowed before '(' */
6720         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
6721         | BUILTIN_FUNC_NAME
6722         ;

6723     lvalue      : NAME
6724                 | NAME '[' expr_list ']'
6725                 | '$' expr
6726                 ;

6727     non_unary_input_function : simple_get
6728                             | simple_get '<' expr
6729                             | non_unary_expr '|' simple_get
6730                             ;

6731     unary_input_function : unary_expr '|' simple_get
6732                         ;

6733     simple_get      : GETLINE
6734                     | GETLINE lvalue
6735                     ;

6736     newline_opt    : /* empty */
6737                     | newline_opt NEWLINE
6738                     ;

```

6739 This grammar has several ambiguities that shall be resolved as follows:

- 6740 • Operator precedence and associativity shall be as described in Table 4-1 (on page 2370).
- 6741 • In case of ambiguity, an **else** shall be associated with the most immediately preceding **if** that
- 6742 would satisfy the grammar.
- 6743 • In some contexts, a slash ('/') that is used to surround an ERE could also be the division
- 6744 operator. This shall be resolved in such a way that wherever the division operator could
- 6745 appear, a slash is assumed to be the division operator. (There is no unary division operator.)

6746 One convention that might not be obvious from the formal grammar is where <newline>

6747 characters are acceptable. There are several obvious placements such as terminating a statement,

6748 and a backslash can be used to escape <newline> characters between any lexical tokens. In

6749 addition, <newline> characters without backslashes can follow a comma, an open brace, logical

6750 AND operator ("&&"), logical OR operator (" || "), the **do** keyword, the **else** keyword, and the

6751 closing parenthesis of an **if**, **for**, or **while** statement. For example:

```
6752 { print $1,
6753         $2 }
```

6754 Lexical Conventions

6755 The lexical conventions for *awk* programs, with respect to the preceding grammar, shall be as

6756 follows:

- 6757 1. Except as noted, *awk* shall recognize the longest possible token or delimiter beginning at a
- 6758 given point.
- 6759 2. A comment shall consist of any characters beginning with the number sign character and
- 6760 terminated by, but excluding the next occurrence of, a <newline> character. Comments
- 6761 shall have no effect, except to delimit lexical tokens.
- 6762 3. The <newline> character shall be recognized as the token **NEWLINE**.
- 6763 4. A backslash character immediately followed by a <newline> character shall have no effect.
- 6764 5. The token **STRING** shall represent a string constant. A string constant shall begin with the
- 6765 character ' " '. Within a string constant, a backslash character shall be considered to begin
- 6766 an escape sequence as specified in the table in the Base Definitions volume of
- 6767 IEEE Std. 1003.1-200x, Chapter 5, File Format Notation ('\ ' , '\a' , '\b' , '\f' , '\n' ,
- 6768 '\r' , '\t' , '\v'). In addition, the escape sequences in Table 4-2 (on page 2376) shall be
- 6769 recognized. A <newline> character shall not occur within a string constant. A string
- 6770 constant shall be terminated by the first unescaped occurrence of the character ' " ' after
- 6771 the one that begins the string constant. The value of the string shall be the sequence of all
- 6772 unescaped characters and values of escape sequences between, but not including, the two
- 6773 delimiting ' " ' characters.
- 6774 6. The token **ERE** represents an extended regular expression constant. An ERE constant shall
- 6775 begin with the slash character. Within an ERE constant, a backslash character shall be
- 6776 considered to begin an escape sequence as specified in the table in the Base Definitions
- 6777 volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation. In addition, the escape
- 6778 sequences in Table 4-2 (on page 2376) shall be recognized. The application shall ensure that
- 6779 a <newline> character does not occur within an ERE constant. An ERE constant shall be
- 6780 terminated by the first unescaped occurrence of the slash character after the one that
- 6781 begins the ERE constant. The extended regular expression represented by the ERE constant
- 6782 shall be the sequence of all unescaped characters and values of escape sequences between,
- 6783 but not including, the two delimiting slash characters.

6784 7. A <blank> character shall have no effect, except to delimit lexical tokens or within
6785 **STRING** or **ERE** tokens.

6786 8. The token **NUMBER** shall represent a numeric constant. Its form and numeric value shall
6787 be equivalent to either of the tokens **floating-constant** or **integer-constant** as specified by
6788 the ISO C standard, with the following exceptions:

6789 a. An integer constant cannot begin with 0x or include the hexadecimal digits 'a', 'b',
6790 'c', 'd', 'e', 'f', 'A', 'B', 'C', 'D', 'E', or 'F'.

6791 b. The value of an integer constant beginning with 0 shall be taken in decimal rather
6792 than octal.

6793 c. An integer constant cannot include a suffix ('u', 'U', 'l', or 'L').

6794 d. A floating constant cannot include a suffix ('f', 'F', 'l', or 'L').

6795 If the value is too large or too small to be representable, the behavior is undefined.

6796 9. A sequence of underscores, digits, and alphabetic characters from the portable character set (see the
6797 Base Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set),
6798 beginning with an underscore or alphabetic, shall be considered a word.

6799 10. The following words are keywords that shall be recognized as individual tokens; the name
6800 of the token is the same as the keyword:

6801	BEGIN	delete	END	function	in	printf
6802	break	do	exit	getline	next	return
6803	continue	else	for	if	print	while

6804 11. The following words are names of built-in functions and shall be recognized as the token
6805 **BUILTIN_FUNC_NAME**:

6806	atan2	gsub	log	split	sub	toupper
6807	close	index	match	sprintf	substr	
6808	cos	int	rand	sqrt	system	
6809	exp	length	sin	srand	tolower	

6810 The above-listed keywords and names of built-in functions are considered reserved words.

6811 12. The token **NAME** shall consist of a word that is not a keyword or a name of a built-in
6812 function and is not followed immediately (without any delimiters) by the '(' character.

6813 13. The token **FUNC_NAME** shall consist of a word that is not a keyword or a name of a
6814 built-in function, followed immediately (without any delimiters) by the '(' character. The
6815 '(' character shall not be included as part of the token.

6816 14. The following two-character sequences shall be recognized as the named tokens:

Token Name	Sequence	Token Name	Sequence
6817 ADD_ASSIGN	+=	6818 NO_MATCH	!~
6819 SUB_ASSIGN	-=	EQ	==
6820 MUL_ASSIGN	*=	LE	<=
6821 DIV_ASSIGN	/=	GE	>=
6822 MOD_ASSIGN	%=	NE	!=
6823 POW_ASSIGN	^=	INCR	++
6824 OR		DECR	--
6825 AND	&&	APPEND	>>

6826 15. The following single characters shall be recognized as tokens whose names are the
6827 character:

6828 <newline> { } () [] , ; + - * % ^ ! > < | ? : ~ \$ =

6829 There is a lexical ambiguity between the token **ERE** and the tokens **'/'** and **DIV_ASSIGN**.
6830 When an input sequence begins with a slash character in any syntactic context where the token
6831 **'/'** or **DIV_ASSIGN** could appear as the next token in a valid program, the longer of those two
6832 tokens that can be recognized shall be recognized. In any other syntactic context where the token
6833 **ERE** could appear as the next token in a valid program, the token **ERE** shall be recognized.

6834 EXIT STATUS

6835 The following exit values shall be returned:

6836 0 All input files were processed successfully.

6837 >0 An error occurred.

6838 The exit status can be altered within the program by using an **exit** expression.

6839 CONSEQUENCES OF ERRORS

6840 If any *file* operand is specified and the named file cannot be accessed, *awk* shall write a
6841 diagnostic message to standard error and terminate without any further action.

6842 If the program specified by either the *program* operand or a *progfile* operand is not a valid *awk*
6843 program (as specified in the EXTENDED DESCRIPTION section), the behavior is undefined.

6844 APPLICATION USAGE

6845 The **index**, **length**, **match**, and **substr** functions should not be confused with similar functions in
6846 the ISO C standard; the *awk* versions deal with characters, while the ISO C standard deals with
6847 bytes.

6848 Because the concatenation operation is represented by adjacent expressions rather than an
6849 explicit operator, it is often necessary to use parentheses to enforce the proper evaluation
6850 precedence.

6851 EXAMPLES

6852 The *awk* program specified in the command line is most easily specified within single-quotes (for
6853 example, *'program'*) for applications using *sh*, because *awk* programs commonly contain
6854 characters that are special to the shell, including double-quotes. In the cases where an *awk*
6855 program contains single-quote characters, it is usually easiest to specify most of the program as
6856 strings within single-quotes concatenated by the shell with quoted single-quote characters. For
6857 example:

```
6858 awk '/\''/ { print "quote:", $0 }'
```

6859 prints all lines from the standard input containing a single-quote character, prefixed with *quote:*.

6860 The following are examples of simple *awk* programs:

6861 1. Write to the standard output all input lines for which field 3 is greater than 5:

```
6862 $3 > 5
```

6863 2. Write every tenth line:

```
6864 (NR % 10) == 0
```

6865 3. Write any line with a substring matching the regular expression:

```
6866 /(G|D)(2[0-9][[:alpha:]]*)/
```

- 6867 4. Print any line with a substring containing a 'G' or 'D', followed by a sequence of digits
6868 and characters. This example uses character classes **digit** and **alpha** to match language-
6869 independent digit and alphabetic characters respectively:
- ```
6870 /(G|D)([[:digit:][:alpha:]]*)/
```
- 6871 5. Write any line in which the second field matches the regular expression and the fourth  
6872 field does not:
- ```
6873 $2 ~ /xyz/ && $4 !~ /xyz/
```
- 6874 6. Write any line in which the second field contains a backslash:
- ```
6875 $2 ~ /\//
```
- 6876 7. Write any line in which the second field contains a backslash. Note that backslash escapes  
6877 are interpreted twice, once in lexical processing of the string and once in processing the  
6878 regular expression:
- ```
6879 $2 ~ "\\\""
```
- 6880 8. Write the second to the last and the last field in each line. Separate the fields by a colon:
- ```
6881 {OFS=":";print $(NF-1), $NF}
```
- 6882 9. Write the line number and number of fields in each line. The three strings representing the  
6883 line number, the colon, and the number of fields are concatenated and that string is written  
6884 to standard output:
- ```
6885 {print NR ":" NF}
```
- 6886 10. Write lines longer than 72 characters:
- ```
6887 length($0) > 72
```
- 6888 11. Write first two fields in opposite order separated by the **OFS**:
- ```
6889 { print $2, $1 }
```
- 6890 12. Same, with input fields separated by comma or <space> and <tab> characters, or both:
- ```
6891 BEGIN { FS = ",[\t]*|[\t]+" }
6892 { print $2, $1 }
```
- 6893 13. Add up first column, print sum, and average:
- ```
6894 {s += $1 }
6895 END {print "sum is ", s, " average is", s/NR}
```
- 6896 14. Write fields in reverse order, one per line (many lines out for each line in):
- ```
6897 { for (i = NF; i > 0; --i) print $i }
```
- 6898 15. Write all lines between occurrences of the strings **start** and **stop**:
- ```
6899 /start/, /stop/
```
- 6900 16. Write all lines whose first field is different from the previous one:
- ```
6901 $1 != prev { print; prev = $1 }
```
- 6902 17. Simulate *echo*:
- ```
6903 BEGIN {
6904     for (i = 1; i < ARGV; ++i)
6905         printf("%s%s", ARGV[i], i==ARGV-1?"\n":" ")
```

```

6906     }
6907 18. Write the path prefixes contained in the PATH environment variable, one per line:
6908     BEGIN {
6909         n = split (ENVIRON["PATH"], path, ":")
6910         for (i = 1; i <= n; ++i)
6911             print path[i]
6912     }
6913 19. If there is a file named input containing page headers of the form:
6914     Page #
6915     and a file named program that contains:
6916     /Page/    { $2 = n++; }
6917             { print }
6918     then the command line:
6919     awk -f program n=5 input
6920     prints the file input, filling in page numbers starting at 5.

```

6921 RATIONALE

6922 The ISO POSIX-2 standard description is based on the new *awk*, “*nawk*”, (see the referenced *The* |
6923 *AWK Programming Language*), which introduced a number of new features to the historical *awk*:

- 6924 1. New keywords: **delete**, **do**, **functin**, **return**
- 6925 2. New built-in functions: **atan2**, **close**, **cos**, **gsub**, **match**, **rand**, **sin**, **srand**, **sub**, **system**
- 6926 3. New predefined variables: **FNR**, **ARGC**, **ARGV**, **RSTART**, **RLENGTH**, **SUBSEP**
- 6927 4. New expression operators: **?**, **:**, **..**, **^**
- 6928 5. The **FS** variable and the third argument to **split**, now treated as extended regular
6929 expressions.
- 6930 6. The operator precedence, changed to more closely match the C language. Two examples
6931 of code that operate differently are:

```

6932     while ( n /= 10 > 1) ...
6933     if (!"wk" ~ /bwk/) ...

```

6934 Several features have been added based on newer implementations of *awk*:

- 6935 • Multiple instances of **-f progfile** are permitted
- 6936 • The new option **-v assignment**
- 6937 • The new predefined variable **ENVIRON**
- 6938 • New built-in functions **toupper**, and **tolower**
- 6939 • More formatting capabilities are added to **printf** to match the ISO C standard

6940 The overall *awk* syntax has always been based on the C language, with a few features from the
6941 shell command language and other sources. Because of this, it is not completely compatible with
6942 any other language, which has caused confusion for some users. It is not the intent of the
6943 standard developers to address such issues. IEEE Std. 1003.1-200x has made a few relatively
6944 minor changes toward making the language more compatible with the C language as specified
6945 by the ISO C standard; most of these changes are based on similar changes in recent

6946 implementations, as described above. There remain several C-language conventions that are not
 6947 in *awk*. One of the notable ones is the comma operator, which is commonly used to specify
 6948 multiple expressions in the C language **for** statement. Also, there are various places where *awk*
 6949 is more restrictive than the C language regarding the type of expression that can be used in a given
 6950 context. These limitations are due to the different features that the *awk* language does provide.

6951 Regular expressions in *awk* have been extended somewhat from historical implementations to
 6952 make them a pure superset of extended regular expressions, as defined by IEEE Std. 1003.1-200x
 6953 (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.4, Extended Regular
 6954 Expressions). The main extensions are internationalization features and interval expressions.
 6955 Historical implementations of *awk* have long supported backslash escape sequences as an
 6956 extension to extended regular expressions, and this extension has been retained despite
 6957 inconsistency with other utilities. The number of escape sequences recognized in both extended
 6958 regular expressions and strings has varied (generally increasing with time) among
 6959 implementations. The set specified by IEEE Std. 1003.1-200x includes most sequences known to
 6960 be supported by popular implementations and by the ISO C standard. One sequence that is not
 6961 supported is hexadecimal value escapes beginning with `'\x'`. This would allow values
 6962 expressed in more than 9 bits to be used within *awk* as in the ISO C standard. However, because
 6963 this syntax has a non-deterministic length, it does not permit the subsequent character to be a
 6964 hexadecimal digit. This limitation can be dealt with in the C language by the use of lexical string
 6965 concatenation. In the *awk* language, concatenation could also be a solution for strings, but not for
 6966 extended regular expressions (either lexical ERE tokens or strings used dynamically as regular
 6967 expressions). Because of this limitation, the feature has not been added to IEEE Std. 1003.1-200x.

6968 When a string variable is used in a context where an extended regular expression normally
 6969 appears (where the lexical token ERE is used in the grammar) the string does not contain the
 6970 literal slashes.

6971 Some versions of *awk* allow the form:

```
6972 func name(args, ... ) { statements }
```

6973 This has been deprecated by the authors of the language, who asked that it not be included in
 6974 IEEE Std. 1003.1-200x.

6975 Historical implementations of *awk* produce an error if a **next** statement is executed in a **BEGIN**
 6976 action, and cause *awk* to terminate if a **next** statement is executed in an **END** action. This
 6977 behavior has not been documented, and it was not believed that it was necessary to standardize
 6978 it.

6979 The specification of conversions between string and numeric values is much more detailed than
 6980 in the documentation of historical implementations or in the referenced *The AWK Programming*
 6981 *Language*. Although most of the behavior is designed to be intuitive, the details are necessary to
 6982 ensure compatible behavior from different implementations. This is especially important in
 6983 relational expressions since the types of the operands determine whether a string or numeric
 6984 comparison is performed. From the perspective of an application writer, it is usually sufficient to
 6985 expect intuitive behavior and to force conversions (by adding zero or concatenating a null
 6986 string) when the type of an expression does not obviously match what is needed. The intent has
 6987 been to specify historical practice in almost all cases. The one exception is that, in historical
 6988 implementations, variables and constants maintain both string and numeric values after their
 6989 original value is converted by any use. This means that referencing a variable or constant can
 6990 have unexpected side effects. For example, with historical implementations the following
 6991 program:

```
6992 {  
6993     a = "+2"
```

```

6994         b = 2
6995         if (NR % 2)
6996             c = a + b
6997         if (a == b)
6998             print "numeric comparison"
6999         else
7000             print "string comparison"
7001     }

```

7002 would perform a numeric comparison (and output numeric comparison) for each odd-
7003 numbered line, but perform a string comparison (and output string comparison) for each even-
7004 numbered line. IEEE Std. 1003.1-200x ensures that comparisons will be numeric if necessary.
7005 With historical implementations, the following program:

```

7006 BEGIN {
7007     OFMT = "%e"
7008     print 3.14
7009     OFMT = "%f"
7010     print 3.14
7011 }

```

7012 would output "3.140000e+00" twice, because in the second **print** statement the constant
7013 "3.14" would have a string value from the previous conversion. IEEE Std. 1003.1-200x requires
7014 that the output of the second **print** statement be "3.140000". The behavior of historical
7015 implementations was seen as too unintuitive and unpredictable.

7016 It was pointed out that with the rules contained in early drafts, the following script would print
7017 nothing:

```

7018 BEGIN {
7019     y[1.5] = 1
7020     OFMT = "%e"
7021     print y[1.5]
7022 }

```

7023 Therefore, a new variable, **CONVFMT**, was introduced. The **OFMT** variable is now restricted to
7024 affecting output conversions of numbers to strings and **CONVFMT** is used for internal
7025 conversions, such as comparisons or array indexing. The default value is the same as that for
7026 **OFMT**, so unless a program changes **CONVFMT** (which no historical program would do), it
7027 will receive the historical behavior associated with internal string conversions.

7028 The POSIX *awk* lexical and syntactic conventions are specified more formally than in other
7029 sources. Again the intent has been to specify historical practice. One convention that may not be
7030 obvious from the formal grammar as in other verbal descriptions is where <newline> characters
7031 are acceptable. There are several obvious placements such as terminating a statement, and a
7032 backslash can be used to escape <newline> characters between any lexical tokens. In addition,
7033 <newline> characters without backslashes can follow a comma, an open brace, a logical AND
7034 operator ("&&"), a logical OR operator ("||"), the **do** keyword, the **else** keyword, and the
7035 closing parenthesis of an **if**, **for**, or **while** statement. For example:

```

7036 { print $1,
7037     $2 }

```

7038 The requirement that *awk* add a trailing <newline> character to the program argument text is to
7039 simplify the grammar, making it match a text file in form. There is no way for an application or
7040 test suite to determine whether a literal <newline> is added or whether *awk* simply acts as if it
7041 did.

7042 IEEE Std. 1003.1-200x requires several changes from historical implementations in order to
 7043 support internationalization. Probably the most subtle of these is the use of the decimal-point
 7044 character, defined by the *LC_NUMERIC* category of the locale, in representations of floating-
 7045 point numbers. This locale-specific character is used in recognizing numeric input, in converting
 7046 between strings and numeric values, and in formatting output. However, regardless of locale,
 7047 the period character (the decimal-point character of the POSIX locale) is the decimal-point
 7048 character recognized in processing *awk* programs (including assignments in command line
 7049 arguments). This is essentially the same convention as the one used in the ISO C standard. The
 7050 difference is that the C language includes the *setlocale()* function, which permits an application
 7051 to modify its locale. Because of this capability, a C application begins executing with its locale
 7052 set to the C locale, and only executes in the environment-specified locale after an explicit call to
 7053 *setlocale()*. However, adding such an elaborate new feature to the *awk* language was seen as
 7054 inappropriate for IEEE Std. 1003.1-200x. It is possible to execute an *awk* program explicitly in any
 7055 desired locale by setting the environment in the shell.

7056 The undefined behavior resulting from NULs in extended regular expressions allows future
 7057 extensions for the GNU *gawk* program to process binary data.

7058 The behavior in the case of invalid *awk* programs (including lexical, syntactic, and semantic
 7059 errors) is undefined because it was considered overly limiting on implementations to specify. In
 7060 most cases such errors can be expected to produce a diagnostic and a non-zero exit status.
 7061 However, some implementations may choose to extend the language in ways that make use of
 7062 certain invalid constructs. Other invalid constructs might be deemed worthy of a warning, but
 7063 otherwise cause some reasonable behavior. Still other constructs may be very difficult to detect
 7064 in some implementations. Also, different implementations might detect a given error during an
 7065 initial parsing of the program (before reading any input files) while others might detect it when
 7066 executing the program after reading some input. Implementors should be aware that diagnosing
 7067 errors as early as possible and producing useful diagnostics can ease debugging of applications,
 7068 and thus make an implementation more usable.

7069 The unspecified behavior from using multi-character **RS** values is to allow possible future
 7070 extensions based on extended regular expressions used for record separators. Historical
 7071 implementations take the first character of the string and ignore the others.

7072 Unspecified behavior when *split(string,array,<null>)* is used is to allow a proposed future
 7073 extension that would split up a string into an array of individual characters.

7074 In the context of the **getline** function, equally good arguments for different precedences of the |
 7075 and < operators can be made. Historical practice has been that:

```
7076 getline < "a" "b"
```

7077 is parsed as:

```
7078 ( getline < "a" ) "b"
```

7079 although many would argue that the intent was that the file **ab** should be read. However:

```
7080 getline < "x" + 1
```

7081 parses as:

```
7082 getline < ( "x" + 1 )
```

7083 Similar problems occur with the | version of **getline**, particularly in combination with \$. For
 7084 example:

```
7085 $"echo hi" | getline
```

7086 (This situation is particularly problematic when used in a **print** statement, where the `|getline`
7087 part might be a redirection of the **print**.)

7088 Since in most cases such constructs are not (or at least should not) be used (because they have a
7089 natural ambiguity for which there is no conventional parsing), the meaning of these constructs
7090 has been made explicitly unspecified. (The effect is that a portable application that runs into the
7091 problem must parenthesize to resolve the ambiguity.) There appeared to be few if any actual
7092 uses of such constructs.

7093 Grammars can be written that would cause an error under these circumstances. Where
7094 backwards compatibility is not a large consideration, implementors may wish to use such
7095 grammars.

7096 Some historical implementations have allowed some built-in functions to be called without an
7097 argument list, the result being a default argument list chosen in some “reasonable” way. Use of
7098 **length** as a synonym for **length(\$0)** is the only one of these forms that is thought to be widely
7099 known or widely used; this particular form is documented in various places (for example, most
7100 historical *awk* reference pages, although not in the referenced *The AWK Programming Language*)
7101 as legitimate practice. With this exception, default argument lists have always been
7102 undocumented and vaguely defined, and it is not at all clear how (or if) they should be
7103 generalized to user-defined functions. They add no useful functionality and preclude possible
7104 future extensions that might need to name functions without calling them. Not standardizing
7105 them seems the simplest course. The standard developers considered that **length** merited special
7106 treatment, however, since it has been documented in the past and sees possibly substantial use
7107 in historical programs. Accordingly, this usage has been made legitimate, but Issue 5 removed
7108 the obsolescent marking for XSI-conforming implementations and many otherwise conforming
7109 applications depend on this feature.

7110 In **sub** and **gsub**, if *repl* is a string literal (the lexical token **STRING**), then two consecutive
7111 backslash characters should be used in the string to ensure a single backslash will precede the
7112 ampersand when the resultant string is passed to the function. (For example, to specify one
7113 literal ampersand in the replacement string, use **gsub(ERE, "\\&")**.)

7114 Historically the only special character in the *repl* argument of **sub** and **gsub** string functions was
7115 the ampersand ('&') character and preceding it with the backslash character was used to turn
7116 off its special meaning.

7117 The description in the ISO POSIX-2:1993 standard introduced behavior such that the backslash
7118 character was another special character and it was unspecified whether there were any other
7119 special characters. This description introduced several portability problems, some of which are
7120 described below, and so it has been replaced with the more historical description. Some of the
7121 problems include:

- 7122 • Historically, to create the replacement string, a script could use **gsub(ERE, "\\&")**, but with
7123 the ISO POSIX-2:1993 standard wording, it was necessary to use **gsub(ERE, "\\&")**.
7124 Backslash characters are doubled here because all string literals are subject to lexical analysis,
7125 which would reduce each pair of backslash characters to a single backslash before being
7126 passed to **gsub**.
- 7127 • Since it was unspecified what the special characters were, for portable scripts to guarantee
7128 that characters are printed literally, each character had to be preceded with a backslash. (For
7129 example, a portable script had to use **gsub(ERE, "\\h\\i")** to produce a replacement string
7130 of "hi".)

7131 The description for comparisons in the ISO POSIX-2:1993 standard did not properly describe
7132 historical practice because of the way numeric strings are compared as numbers. The current
7133 rules cause the following code:

```

7134     if (0 == "000")
7135         print "strange, but true"
7136     else
7137         print "not true"

```

7138 to do a numeric comparison, causing the **if** to succeed. It should be intuitively obvious that this
 7139 is incorrect behavior, and indeed, no historical implementation of *awk* actually behaves this way.

7140 To fix this problem, the definition of *numeric string* was enhanced to include only those values
 7141 obtained from specific circumstances (mostly external sources) where it is not possible to
 7142 determine unambiguously whether the value is intended to be a string or a numeric.

7143 Variables that are assigned to a numeric string shall also be treated as a numeric string. (For
 7144 example, the notion of a numeric string can be propagated across assignments.) In comparisons,
 7145 all variables having the uninitialized value are to be treated as a numeric operand evaluating to
 7146 the numeric value zero.

7147 Uninitialized variables include all types of variables including scalars, array elements, and fields.
 7148 The definition of an uninitialized value in **Variables and Special Variables** (on page 2373) is
 7149 necessary to describe the value placed on uninitialized variables and on fields that are valid (for
 7150 example, < \$NF) but have no characters in them and to describe how these variables are to be
 7151 used in comparisons. A valid field, such as \$1, that has no characters in it can be obtained by
 7152 from an input line of "\t\t" when FS=' \t '. Historically, the comparison (\$1<10) was done
 7153 numerically after evaluating \$1 to the value zero.

7154 The phrase "... also shall have the numeric value of the numeric string" was removed from
 7155 several sections of the ISO POSIX-2:1993 standard because it specifies an unnecessary
 7156 implementation detail. It is not necessary for IEEE Std. 1003.1-200x to specify that these objects
 7157 be assigned two different values. It is only necessary to specify that these objects may evaluate
 7158 to two different values depending on context.

7159 The description of numeric string processing is based on the behavior of the *atof()* function in
 7160 the ISO C standard. While it is not a requirement for an implementation to use this function,
 7161 many historical implementations of *awk* do. In the ISO C standard, floating-point constants use a
 7162 period as a decimal point character for the language itself, independent of the current locale, but
 7163 the *atof()* function and the associated *strtod()* function use the decimal point character of the
 7164 current locale when converting strings to numeric values. Similarly in *awk*, floating point
 7165 constants in an *awk* script use a period independent of the locale, but input strings use the
 7166 decimal point character of the locale.

7167 FUTURE DIRECTIONS

7168 None.

7169 SEE ALSO

7170 *grep*, *lex*, *sed*, the System Interfaces volume of IEEE Std. 1003.1-200x, *atof()*, *setlocale()*, *strtod()*

7171 CHANGE HISTORY

7172 First released in Issue 2.

7173 Issue 4

7174 Aligned with the ISO/IEC 9945-2:1993 standard.

7175 Issue 4, Version 2

7176 The EXAMPLES section is corrected as follows:

- 7177 • In Example 10, the braces are removed.
- 7178 • In Example 17, the invocation of **printf** is corrected.

7179 **Issue 5**

7180 FUTURE DIRECTIONS section added.

7181 **Issue 6**

7182 The *awk* utility is aligned with the IEEE P1003.2b draft standard.

7183 The normative text is reworded to avoid use of the term “must” for application requirements.

7184 **NAME**7185 **basename** — return non-directory portion of a path name7186 **SYNOPSIS**7187 **basename** *string* [*suffix*]7188 **DESCRIPTION**

7189 The *string* operand shall be treated as a path name, as defined in the Base Definitions volume of
 7190 IEEE Std. 1003.1-200x, Section 3.268, Path Name. The string *string* shall be converted to the file
 7191 name corresponding to the last path name component in *string* and then the suffix string *suffix*, if
 7192 present, shall be removed. This shall be done by performing actions equivalent to the following
 7193 steps in order:

- 7194 1. If *string* is a null string, it is unspecified whether the resulting string is ' . ' or a null string.
 7195 In either case, skip steps 2 through 6.
- 7196 2. If *string* is "///", it is implementation-defined whether steps 3 to 6 are skipped or
 7197 processed.
- 7198 3. If *string* consists entirely of slash characters, *string* shall be set to a single slash character. In
 7199 this case, skip steps 4 to 6.
- 7200 4. If there are any trailing slash characters in *string*, they shall be removed.
- 7201 5. If there are any slash characters remaining in *string*, the prefix of *string* up to and including
 7202 the last slash character in *string* shall be removed.
- 7203 6. If the *suffix* operand is present, is not identical to the characters remaining in *string*, and is
 7204 identical to a suffix of the characters remaining in *string*, the suffix *suffix* shall be removed
 7205 from *string*. Otherwise, *string* is modified by this step. It shall not be considered an error if
 7206 *suffix* is not found in *string*.

7207 The resulting string shall be written to standard output.

7208 **OPTIONS**

7209 None.

7210 **OPERANDS**

7211 The following operands shall be supported:

7212 *string* A string.7213 *suffix* A string.7214 **STDIN**

7215 Not used.

7216 **INPUT FILES**

7217 None.

7218 **ENVIRONMENT VARIABLES**7219 The following environment variables shall affect the execution of *basename*:

7220 **LANG** Provide a default value for the internationalization variables that are unset or null.
 7221 If **LANG** is unset or null, the corresponding value from the implementation-
 7222 defined default locale shall be used. If any of the internationalization variables
 7223 contains an invalid setting, the utility shall behave as if none of the variables had
 7224 been defined.

7225 **LC_ALL** If set to a non-empty string value, override the values of all the other
 7226 internationalization variables.

7227 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 7228 characters (for example, single-byte as opposed to multi-byte characters in
 7229 arguments).

7230 *LC_MESSAGES*
 7231 Determine the locale that should be used to affect the format and contents of
 7232 diagnostic messages written to standard error.

7233 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

7234 **ASYNCHRONOUS EVENTS**
 7235 Default.

7236 **STDOUT**
 7237 The *basename* utility shall write a line to the standard output in the following format:
 7238 "%s\n", <resulting string>

7239 **STDERR**
 7240 Used only for diagnostic messages.

7241 **OUTPUT FILES**
 7242 None.

7243 **EXTENDED DESCRIPTION**
 7244 None.

7245 **EXIT STATUS**
 7246 The following exit values shall be returned:
 7247 0 Successful completion.
 7248 >0 An error occurred.

7249 **CONSEQUENCES OF ERRORS**
 7250 Default.

7251 **APPLICATION USAGE**
 7252 The definition of *pathname* specifies implementation-defined behavior for path names starting
 7253 with two slash characters. Therefore, applications shall not arbitrarily add slashes to the
 7254 beginning of a path name unless they can ensure that there are more or less than two or are
 7255 prepared to deal with the implementation-defined consequences.

7256 **EXAMPLES**
 7257 If the string *string* is a valid path name:
 7258 `$(basename "string")`
 7259 produces a file name that could be used to open the file named by *string* in the directory
 7260 returned by:
 7261 `$(dirname "string")`
 7262 If the string *string* is not a valid path name, the same algorithm is used, but the result need not be
 7263 a valid file name. The *basename* utility is not expected to make any judgements about the validity
 7264 of *string* as a path name; it just follows the specified algorithm to produce a result string.
 7265 The following shell script compiles `/usr/src/cmd/cat.c` and moves the output to a file named `cat`
 7266 in the current directory when invoked with the argument `/usr/src/cmd/cat` or with the argument
 7267 `/usr/src/cmd/cat.c`:

7268 c99 \$(dirname "\$1")/\$(basename "\$1" .c).c
7269 mv a.out \$(basename "\$1" .c)

7270 **RATIONALE**

7271 The behaviors of *basename* and *dirname* have been coordinated so that when *string* is a valid path
7272 name:

7273 \$(basename "*string*")

7274 would be a valid file name for the file in the directory:

7275 \$(dirname "*string*")

7276 This would not work for the early proposal versions of these utilities due to the way it specified
7277 handling of trailing slashes.

7278 Since the definition of *pathname* specifies implementation-defined behavior for path names
7279 starting with two slash characters, this volume of IEEE Std. 1003.1-200x specifies similar
7280 implementation-defined behavior for the *basename* and *dirname* utilities.

7281 **FUTURE DIRECTIONS**

7282 None.

7283 **SEE ALSO**

7284 *dirname*, Section 2.5 (on page 2241)

7285 **CHANGE HISTORY**

7286 First released in Issue 2.

7287 **Issue 4**

7288 Aligned with the ISO/IEC 9945-2:1993 standard.

7289 **Issue 6**

7290 IEEE PASC Interpretation 1003.2 #164 has been applied.

7291 The normative text is reworded to avoid use of the term “must” for application requirements.

7292 **NAME**

7293 batch — schedule commands to be executed in a batch queue

7294 **SYNOPSIS**

7295 UP *batch*

7296

7297 **DESCRIPTION**

7298 The *batch* utility shall read commands from standard input and schedule them for execution in a
7299 batch queue. It shall be the equivalent of the command:

7300 at -q b -m now

7301 where queue *b* is a special *at* queue, specifically for batch jobs. Batch jobs shall be submitted to
7302 the batch queue with no time constraints and shall be run by the system using algorithms, based
7303 on unspecified factors, that may vary with each invocation of *batch*.

7304 XSI Users are permitted to use *batch* if their name appears in the file `/usr/lib/cron/at.allow`. If that file
7305 does not exist, the file `/usr/lib/cron/at.deny` is checked to determine whether the user should be
7306 denied access to *batch*. If neither file exists, only a process with the appropriate privileges is
7307 allowed to submit a job. If only `at.deny` exists and is empty, global usage is permitted. The
7308 `at.allow` and `at.deny` files consist of one user name per line.

7309 **OPTIONS**

7310 None.

7311 **OPERANDS**

7312 None.

7313 **STDIN**

7314 The standard input shall be a text file consisting of commands acceptable to the shell command
7315 language described in Chapter 2 (on page 2235).

7316 **INPUT FILES**

7317 XSI The text files `/usr/lib/cron/at.allow` and `/usr/lib/cron/at.deny` contain user names, one per line, of
7318 users who are, respectively, authorized or denied access to the *at* and *batch* utilities.

7319 **ENVIRONMENT VARIABLES**

7320 The following environment variables shall affect the execution of *batch*:

7321 *LANG* Provide a default value for the internationalization variables that are unset or null.
7322 If *LANG* is unset or null, the corresponding value from the implementation-
7323 defined default locale shall be used. If any of the internationalization variables
7324 contains an invalid setting, the utility shall behave as if none of the variables had
7325 been defined.

7326 *LC_ALL* If set to a non-empty string value, override the values of all the other
7327 internationalization variables.

7328 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
7329 characters (for example, single-byte as opposed to multi-byte characters in
7330 arguments and input files).

7331 *LC_MESSAGES*

7332 Determine the locale that should be used to affect the format and contents of
7333 diagnostic messages written to standard error and informative messages written to
7334 standard output.

7335 *LC_TIME* Determine the format and contents for date and time strings written by *batch*.

- 7336 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 7337 **SHELL** Determine the name of a command interpreter to be used to invoke the at-job. If
7338 the variable is unset or null, *sh* shall be used. If it is set to a value other than a name
7339 for *sh*, the implementation shall do one of the following: use that shell; use *sh*; use
7340 the login shell from the user database; any of the preceding accompanied by a
7341 warning diagnostic about which was chosen.
- 7342 **TZ** Determine the timezone. The job shall be submitted for execution at the time
7343 specified by *timespec* or *-t time* relative to the timezone specified by the *TZ*
7344 variable. If *timespec* specifies a timezone, it overrides *TZ*. If *timespec* does not
7345 specify a timezone and *TZ* is unset or null, an unspecified default timezone shall
7346 be used.
- 7347 **ASYNCHRONOUS EVENTS**
- 7348 Default.
- 7349 **STDOUT**
- 7350 When standard input is a terminal, prompts of unspecified format for each line of the user input
7351 described in the STDIN section may be written to standard output.
- 7352 **STDERR**
- 7353 The following shall be written to standard error when a job has been successfully submitted:
- 7354 "job %s at %s\n", *at_job_id*, <date>
- 7355 where *date* shall be equivalent in format to the output of:
- 7356 date +"%a %b %e %T %Y"
- 7357 The date and time written shall be adjusted so that they appear in the timezone of the user (as
7358 determined by the *TZ* variable).
- 7359 Neither this, nor warning messages concerning the selection of the command interpreter, are
7360 considered a diagnostic that changes the exit status.
- 7361 Diagnostic messages, if any, shall be written to standard error.
- 7362 **OUTPUT FILES**
- 7363 None.
- 7364 **EXTENDED DESCRIPTION**
- 7365 None.
- 7366 **EXIT STATUS**
- 7367 The following exit values shall be returned:
- 7368 0 Successful completion.
- 7369 >0 An error occurred.
- 7370 **CONSEQUENCES OF ERRORS**
- 7371 The job shall not be scheduled.

7372 **APPLICATION USAGE**

7373 It may be useful to redirect standard output within the specified commands.

7374 **EXAMPLES**

7375 1. This sequence can be used at a terminal:

```
7376     batch
7377     sort < file >outfile
7378     EOT
```

7379 2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a
7380 command procedure (the sequence of output redirection specifications is significant):7381

```
batch <<! diff file1 file2 2>&1 >outfile | mailx mygroup !
```

7382 **RATIONALE**7383 Early proposals described *batch* in a manner totally separated from *at*, even though the historical
7384 model treated it almost as a synonym for *at -qb*. A number of features were added to list and
7385 control batch work separately from those in *at*. Upon further reflection, it was decided that the
7386 benefit of this did not merit the change to the historical interface.7387 The **-m** option was included on the equivalent *at* command because it is historical practice to
7388 mail results to the submitter, even if all job-produced output is redirected. As explained in the
7389 RATIONALE for *at*, the **now** keyword submits the job for immediate execution (after scheduling
7390 delays), despite some historical systems where *at now* would have been considered an error.7391 **FUTURE DIRECTIONS**

7392 None.

7393 **SEE ALSO**7394 *at*7395 **CHANGE HISTORY**

7396 First released in Issue 2.

7397 **Issue 4**7398 Format reorganized and separated from the *at* description.

7399 Aligned with the ISO/IEC 9945-2:1993 standard.

7400 **Issue 6**

7401 This utility is now marked as part of the User Portability Utilities option.

7402 The NAME is changed to align with the IEEE P1003.2b draft standard.

7403 The normative text is reworded to avoid use of the term “must” for application requirements.

7404 **NAME**

7405 bc — arbitrary-precision arithmetic language

7406 **SYNOPSIS**7407 bc [-l] [*file* ...]7408 **DESCRIPTION**

7409 The *bc* utility shall implement an arbitrary precision calculator. It shall take input from any files
 7410 given, then read from the standard input. If the standard input and standard output to *bc* are
 7411 attached to a terminal, the invocation of *bc* shall be considered to be *interactive*, causing
 7412 behavioral constraints described in the following sections.

7413 **OPTIONS**

7414 The *bc* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,
 7415 Utility Syntax Guidelines.

7416 The following option shall be supported:

7417 -l (The letter ell.) Define the math functions and initialize *scale* to 20, instead of the
 7418 default zero; see the EXTENDED DESCRIPTION section.

7419 **OPERANDS**

7420 The following operand shall be supported:

7421 *file* A path name of a text file containing *bc* program statements. After all *files* have
 7422 been read, *bc* shall read the standard input.

7423 **STDIN**

7424 See the INPUT FILES section.

7425 **INPUT FILES**

7426 Input files shall be text files containing a sequence of comments, statements, and function
 7427 definitions that shall be executed as they are read.

7428 **ENVIRONMENT VARIABLES**7429 The following environment variables shall affect the execution of *bc*:

7430 *LANG* Provide a default value for the internationalization variables that are unset or null.
 7431 If *LANG* is unset or null, the corresponding value from the implementation-
 7432 defined default locale shall be used. If any of the internationalization variables
 7433 contains an invalid setting, the utility shall behave as if none of the variables had
 7434 been defined.

7435 *LC_ALL* If set to a non-empty string value, override the values of all the other
 7436 internationalization variables.

7437 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 7438 characters (for example, single-byte as opposed to multi-byte characters in
 7439 arguments and input files).

7440 *LC_MESSAGES*

7441 Determine the locale that should be used to affect the format and contents of
 7442 diagnostic messages written to standard error.

7443 *XLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

7444 **ASYNCHRONOUS EVENTS**

7445 Default.

7446 **STDOUT**

7447 The output of the *bc* utility shall be controlled by the program read, and consist of zero or more
 7448 lines containing the value of all executed expressions without assignments. The radix and
 7449 precision of the output shall be controlled by the values of the **obase** and **scale** variables; see the
 7450 EXTENDED DESCRIPTION section.

7451 **STDERR**

7452 Used only for diagnostic messages.

7453 **OUTPUT FILES**

7454 None.

7455 **EXTENDED DESCRIPTION**7456 **Grammar**

7457 The grammar in this section and the lexical conventions in the following section shall together
 7458 describe the syntax for *bc* programs. The general conventions for this style of grammar are
 7459 described in Section 1.10 (on page 2223). A valid program can be represented as the non-
 7460 terminal symbol **program** in the grammar. This formal syntax shall take precedence over the text
 7461 syntax description.

```

7462 %token    EOF NEWLINE STRING LETTER NUMBER
7463 %token    MUL_OP
7464 /*      '*' , '/' , '%'                               */
7465 %token    ASSIGN_OP
7466 /*      '=' , '+=' , '-=' , '*=' , '/=' , '%=' , '^=' */
7467 %token    REL_OP
7468 /*      '==' , '<=' , '>=' , '!=' , '<' , '>'           */
7469 %token    INCR_DECR
7470 /*      '++' , '--'                                   */
7471 %token    Define    Break    Quit    Length
7472 /*      'define' , 'break' , 'quit' , 'length'       */
7473 %token    Return    For    If    While    Sqrt
7474 /*      'return' , 'for' , 'if' , 'while' , 'sqrt'   */
7475 %token    Scale    Ibase    Obase    Auto
7476 /*      'scale' , 'ibase' , 'obase' , 'auto'        */
7477 %start    program
7478 %%
7479 program      : EOF
7480              | input_item program
7481              ;
7482 input_item   : semicolon_list NEWLINE
7483              | function
7484              ;
7485 semicolon_list : /* empty */
7486                | statement
7487                | semicolon_list ';' statement
7488                | semicolon_list ';'

```

```

7489             ;
7490     statement_list : /* empty */
7491                   | statement
7492                   | statement_list NEWLINE
7493                   | statement_list NEWLINE statement
7494                   | statement_list ';'
7495                   | statement_list ';' statement
7496             ;
7497     statement      : expression
7498                   | STRING
7499                   | Break
7500                   | Quit
7501                   | Return
7502                   | Return '(' return_expression ')'
7503                   | For '(' expression ';'
7504                       relational_expression ';'
7505                       expression ')' statement
7506                   | If '(' relational_expression ')' statement
7507                   | While '(' relational_expression ')' statement
7508                   | '{' statement_list '}'
7509             ;
7510     function       : Define LETTER '(' opt_parameter_list ')'
7511                   | '{' NEWLINE opt_auto_define_list
7512                       statement_list '}'
7513             ;
7514     opt_parameter_list : /* empty */
7515                   | parameter_list
7516             ;
7517     parameter_list  : LETTER
7518                   | define_list ',' LETTER
7519             ;
7520     opt_auto_define_list : /* empty */
7521                   | Auto define_list NEWLINE
7522                   | Auto define_list ';'
7523             ;
7524     define_list     : LETTER
7525                   | LETTER '[' ']'
7526                   | define_list ',' LETTER
7527                   | define_list ',' LETTER '[' ']'
7528             ;
7529     opt_argument_list : /* empty */
7530                   | argument_list
7531             ;
7532     argument_list   : expression
7533                   | LETTER '[' ']' ',' argument_list"
7534             ;

```

```

7535 relational_expression : expression
7536                        | expression REL_OP expression
7537                        ;
7538 return_expression     : /* empty */
7539                        | expression
7540                        ;
7541 expression            : named_expression
7542                        | NUMBER
7543                        | '(' expression ')'
7544                        | LETTER '(' opt_argument_list ')'
7545                        | '-' expression
7546                        | expression '+' expression
7547                        | expression '-' expression
7548                        | expression MUL_OP expression
7549                        | expression '^' expression
7550                        | INCR_DECR named_expression
7551                        | named_expression INCR_DECR
7552                        | named_expression ASSIGN_OP expression
7553                        | Length '(' expression ')'
7554                        | Sqrt '(' expression ')'
7555                        | Scale '(' expression ')'
7556                        ;
7557 named_expression     : LETTER
7558                        | LETTER '[' expression ']'
7559                        | Scale
7560                        | Ibase
7561                        | Obase
7562                        ;

```

7563 Lexical Conventions in bc

7564 The lexical conventions for *bc* programs, with respect to the preceding grammar, shall be as
7565 follows:

- 7566 1. Except as noted, *bc* shall recognize the longest possible token or delimiter beginning at a
7567 given point.
- 7568 2. A comment shall consist of any characters beginning with the two adjacent characters
7569 `"/*"` and terminated by the next occurrence of the two adjacent characters `"*/"`.
7570 Comments shall have no effect except to delimit lexical tokens.
- 7571 3. The `<newline>` character shall be recognized as the token **NEWLINE**.
- 7572 4. The token **STRING** shall represent a string constant; it shall consist of any characters
7573 beginning with the double-quote character (`'"`) and terminated by another occurrence of
7574 the double-quote character. The value of the string is the sequence of all characters
7575 between, but not including, the two double-quote characters. All characters shall be taken
7576 literally from the input, and there is no way to specify a string containing a double-quote
7577 character. The length of the value of each string shall be limited to `{BC_STRING_MAX}`
7578 bytes.
- 7579 5. A `<blank>` character shall have no effect except as an ordinary character if it appears
7580 within a **STRING** token, or to delimit a lexical token other than **STRING**.

- 7581 6. The combination of a backslash character immediately followed by a <newline> character
 7582 shall have no effect other than to delimit lexical tokens with the following exceptions:
- 7583 • It shall be interpreted as the character sequence "\<newline>" in **STRING** tokens.
 - 7584 • It shall be ignored as part of a multi-line **NUMBER** token.
- 7585 7. The token **NUMBER** shall represent a numeric constant. It shall be recognized by the
 7586 following grammar:
- ```

7587 NUMBER : integer
7588 | '.' integer
7589 | integer '.'
7590 | integer '.' integer
7591 ;

7592 integer : digit
7593 | integer digit
7594 ;

7595 digit : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
7596 | 8 | 9 | A | B | C | D | E | F
7597 ;

```
- 7598 8. The value of a **NUMBER** token shall be interpreted as a numeral in the base specified by  
 7599 the value of the internal register **ibase** (described below). Each of the **digit** characters shall  
 7600 have the value from 0 to 15 in the order listed here, and the period character shall represent  
 7601 the radix point. The behavior is undefined if digits greater than or equal to the value of  
 7602 **ibase** appear in the token. However, note the exception for single-digit values being  
 7603 assigned to **ibase** and **obase** themselves, in **Operations in bc** (on page 2412).
- 7604 9. The following keywords shall be recognized as tokens:
- |      |               |              |               |               |              |
|------|---------------|--------------|---------------|---------------|--------------|
| 7605 | <b>auto</b>   | <b>ibase</b> | <b>length</b> | <b>return</b> | <b>while</b> |
| 7606 | <b>break</b>  | <b>if</b>    | <b>obase</b>  | <b>scale</b>  |              |
| 7607 | <b>define</b> | <b>for</b>   | <b>quit</b>   | <b>sqrt</b>   |              |
- 7608 10. Any of the following characters occurring anywhere except within a keyword shall be  
 7609 recognized as the token **LETTER**:
- ```

7610 a b c d e f g h i j k l m n o p q r s t u v w x y z

```
- 7611 11. The following single-character and two-character sequences shall be recognized as the
 7612 token **ASSIGN_OP**:
- ```

7613 = += -= *= /= %= ^=

```
- 7614 12. If an '=' character, as the beginning of a token, is followed by a '-' character with no  
 7615 intervening delimiter, the behavior is undefined.
- 7616 13. The following single-characters shall be recognized as the token **MUL\_OP**:
- ```

7617 * / %

```
- 7618 14. The following single-character and two-character sequences shall be recognized as the
 7619 token **REL_OP**:
- ```

7620 == <= >= != < >

```
- 7621 15. The following two-character sequences shall be recognized as the token **INCR\_DECR**:

7622            ++    --

7623            16. The following single characters shall be recognized as tokens whose names are the  
7624            character:

7625            <newline> ( ) , + - ; [ ] ^ { }

7626            17. The token **EOF** is returned when the end of input is reached.

### 7627            **Operations in bc**

7628            There are three kinds of identifiers: ordinary identifiers, array identifiers, and function  
7629            identifiers. All three types consist of single lowercase letters. Array identifiers shall be followed  
7630            by square brackets ("[]"). An array subscript is required except in an argument or auto list.  
7631            Arrays are singly dimensioned and can contain up to {BC\_DIM\_MAX} elements. Indexing shall  
7632            begin at zero so an array is indexed from 0 to {BC\_DIM\_MAX}-1. Subscripts shall be truncated  
7633            to integers. The application shall ensure that function identifiers are followed by parentheses,  
7634            possibly enclosing arguments. The three types of identifiers do not conflict.

7635            The following table summarizes the rules for precedence and associativity of all operators.  
7636            Operators on the same line shall have the same precedence; rows are in order of decreasing  
7637            precedence.

7638            **Table 4-3 Operators in bc**

| Operator                  | Associativity |
|---------------------------|---------------|
| ++, --                    | N/A           |
| unary -                   | N/A           |
| ^                         | Right to left |
| *, /, %                   | Left to right |
| +, binary -               | Left to right |
| =, +=, -=, *=, /=, %=, ^= | Right to left |
| ==, <=, >=, !=, <, >      | None          |

7647            Each expression or named expression has a *scale*, which is the number of decimal digits that  
7648            shall be maintained as the fractional portion of the expression.

7649            *Named expressions* are places where values are stored. Named expressions shall be valid on the  
7650            left side of an assignment. The value of a named expression shall be the value stored in the place  
7651            named. Simple identifiers and array elements are named expressions; they have an initial value  
7652            of zero and an initial scale of zero.

7653            The internal registers **scale**, **ibase**, and **obase** are all named expressions. The scale of an  
7654            expression consisting of the name of one of these registers shall be zero; values assigned to any  
7655            of these registers are truncated to integers. The **scale** register shall contain a global value used in  
7656            computing the scale of expressions (as described below). The value of the register **scale** is  
7657            limited to  $0 \leq \text{scale} \leq \{\text{BC\_SCALE\_MAX}\}$  and shall have a default value of zero. The **ibase** and  
7658            **obase** registers are the input and output number radix, respectively. The value of **ibase** shall be  
7659            limited to:

7660             $2 \leq \text{ibase} \leq 16$

7661            The value of **obase** shall be limited to:

7662             $2 \leq \text{obase} \leq \{\text{BC\_BASE\_MAX}\}$

7663            When either **ibase** or **obase** is assigned a single **digit** value from the list in **Lexical Conventions**  
7664            **in bc** (on page 2410), the value shall be assumed in hexadecimal. (For example, **ibase=A** sets to

7665 base ten, regardless of the current **ibase** value.) Otherwise, the behavior is undefined when  
 7666 digits greater than or equal to the value of **ibase** appear in the input. Both **ibase** and **obase** shall  
 7667 have initial values of 10.

7668 Internal computations shall be conducted as if in decimal, regardless of the input and output  
 7669 bases, to the specified number of decimal digits. When an exact result is not achieved, (for  
 7670 example, **scale**=0; 3.2/1) the result shall be truncated.

7671 For all values of **obase** specified by this volume of IEEE Std. 1003.1-200x, *bc* shall output numeric  
 7672 values by performing each of the following steps in order:

- 7673 1. If the value is less than zero, a hyphen ('-') character shall be output.
- 7674 2. One of the following is output, depending on the numerical value:
  - 7675 • If the absolute value of the numerical value is greater than or equal to one, the integer  
 7676 portion of the value shall be output as a series of digits appropriate to **obase** (as  
 7677 described below) most significant digit first. The most significant non-zero digit shall  
 7678 be output next, followed by each successively less significant digit.
  - 7679 • If the absolute value of the numerical value is less than one but greater than zero and  
 7680 the scale of the numerical value is greater than zero, it is unspecified whether the  
 7681 character 0 is output.
  - 7682 • If the numerical value is zero, the character 0 shall be output.
- 7683 3. If the scale of the value is greater than zero and the numeric value is not zero, a period  
 7684 character shall be output, followed by a series of digits appropriate to **obase** (as described  
 7685 below) representing the most significant portion of the fractional part of the value. If *s*  
 7686 represents the scale of the value being output, the number of digits output shall be *s* if  
 7687 **obase** is 10, less than or equal to *s* if **obase** is greater than 10, or greater than or equal to *s* if  
 7688 **obase** is less than 10. For **obase** values other than 10, this should be the number of digits  
 7689 needed to represent a precision of  $10^s$ .

7690 For **obase** values from 2 to 16, valid digits are the first **obase** of the single characters:

7691 0 1 2 3 4 5 6 7 8 9 A B C D E F

7692 which represent the values zero to 15, inclusive, respectively.

7693 For bases greater than 16, each digit shall be written as a separate multi-digit decimal number.  
 7694 Each digit except the most significant fractional digit shall be preceded by a single <space>  
 7695 character. For bases from 17 to 100, *bc* shall write two-digit decimal numbers; for bases from 101  
 7696 to 1 000, three-digit decimal strings, and so on. For example, the decimal number 1 024 in base 25  
 7697 would be written as:

7698 Δ01Δ15Δ24

7699 in base 125, as:

7700 Δ008Δ024

7701 Very large numbers shall be split across lines with 70 characters per line in the POSIX locale;  
 7702 other locales may split at different character boundaries. Lines that are continued shall end with  
 7703 a backslash ('\').

7704 A function call shall consist of a function name followed by parentheses containing a comma-  
 7705 separated list of expressions, which are the function arguments. A whole array passed as an  
 7706 argument shall be specified by the array name followed by empty square brackets. All function  
 7707 arguments shall be passed by value. As a result, changes made to the formal parameters shall  
 7708 have no effect on the actual arguments. If the function terminates by executing a **return**

7709 statement, the value of the function shall be the value of the expression in the parentheses of the  
7710 **return** statement or shall be zero if no expression is provided or if there is no **return** statement.

7711 The result of **sqrt**(*expression*) shall be the square root of the expression. The result shall be  
7712 truncated in the least significant decimal place. The scale of the result shall be the scale of the  
7713 expression or the value of **scale**, whichever is larger.

7714 The result of **length**(*expression*) shall be the total number of significant decimal digits in the  
7715 expression. The scale of the result shall be zero.

7716 The result of **scale**(*expression*) shall be the scale of the expression. The scale of the result shall be  
7717 zero.

7718 A numeric constant shall be an expression. The scale shall be the number of digits that follow the  
7719 radix point in the input representing the constant, or zero if no radix point appears.

7720 The sequence ( *expression* ) shall be an expression with the same value and scale as *expression*.  
7721 The parentheses can be used to alter the normal precedence.

7722 The semantics of the unary and binary operators are as follows:

7723 *-expression*  
7724 The result shall be the negative of the *expression*. The scale of the result shall be the scale of  
7725 *expression*.

7726 The unary increment and decrement operators shall not modify the scale of the named  
7727 expression upon which they operate. The scale of the result shall be the scale of that named  
7728 expression.

7729 *++named-expression*  
7730 The named expression shall be incremented by one. The result shall be the value of the  
7731 named expression after incrementing.

7732 *--named-expression*  
7733 The named expression shall be decremented by one. The result shall be the value of the  
7734 named expression after decrementing.

7735 *named-expression++*  
7736 The named expression shall be incremented by one. The result shall be the value of the  
7737 named expression before incrementing.

7738 *named-expression--*  
7739 The named expression shall be decremented by one. The result shall be the value of the  
7740 named expression before decrementing.

7741 The exponentiation operator, circumflex ( '^ ' ), shall bind right to left.

7742 *expression^expression*  
7743 The result shall be the first *expression* raised to the power of the second *expression*. If the  
7744 second expression is not an integer, the behavior is undefined. If *a* is the scale of the left  
7745 expression and *b* is the absolute value of the right expression, the scale of the result shall be:  
7746 if  $b \geq 0$   $\min(a * b, \max(\text{scale}, a))$  if  $b < 0$  *scale*

7747 The multiplicative operators ( ' \* ' , ' / ' , ' % ' ) shall bind left to right.

7748 *expression\*expression*  
7749 The result shall be the product of the two expressions. If *a* and *b* are the scales of the two  
7750 expressions, then the scale of the result shall be:

7751  $\min(a+b, \max(\text{scale}, a, b))$

7752 *expression/expression*  
 7753 The result shall be the quotient of the two expressions. The scale of the result shall be the  
 7754 value of **scale**.

7755 *expression%expression*  
 7756 For expressions *a* and *b*, *a%b* shall be evaluated equivalent to the steps:

7757 1. Compute *a/b* to current scale.  
 7758 2. Use the result to compute:

7759  $a - (a / b) * b$   
 7760 to scale:  
 7761  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

7762 The scale of the result shall be:  
 7763  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

7764 When **scale** is zero, the '*%*' operator is the mathematical remainder operator.  
 7765 The additive operators ('+', '-') shall bind left to right.

7766 *expression+expression*  
 7767 The result shall be the sum of the two expressions. The scale of the result shall be the  
 7768 maximum of the scales of the expressions.

7769 *expression-expression*  
 7770 The result shall be the difference of the two expressions. The scale of the result shall be the  
 7771 maximum of the scales of the expressions.

7772 The assignment operators ('=', '+=', '-=', '\*=', '/=', '%=', '^=') shall bind right to left.

7773 *named-expression=expression*  
 7774 This expression results in assigning the value of the expression on the right to the named  
 7775 expression on the left. The scale of both the named expression and the result shall be the  
 7776 scale of *expression*.

7777 The compound assignment forms:  
 7778 *named-expression <operator>= expression*  
 7779 shall be equivalent to:  
 7780 *named-expression=named-expression <operator> expression*  
 7781 except that the *named-expression* shall be evaluated only once.

7782 Unlike all other operators, the relational operators ('<', '>', '<=', '>=', '==', '!=') shall be  
 7783 only valid as the object of an **if**, **while**, or inside a **for** statement.

7784 *expression1<expression2*  
 7785 The relation shall be true if the value of *expression1* is strictly less than the value of  
 7786 *expression2*.

7787 *expression1>expression2*  
 7788 The relation shall be true if the value of *expression1* is strictly greater than the value of  
 7789 *expression2*.

7790 *expression1* <= *expression2*  
 7791 The relation shall be true if the value of *expression1* is less than or equal to the value of  
 7792 *expression2*.

7793 *expression1* >= *expression2*  
 7794 The relation shall be true if the value of *expression1* is greater than or equal to the value of  
 7795 *expression2*.

7796 *expression1* = *expression2*  
 7797 The relation shall be true if the values of *expression1* and *expression2* are equal.

7798 *expression1* != *expression2*  
 7799 The relation shall be true if the values of *expression1* and *expression2* are unequal.

7800 There are only two storage classes in *bc*, global and automatic (local). Only identifiers that are  
 7801 local to a function need be declared with the **auto** command. The arguments to a function shall  
 7802 be local to the function. All other identifiers are assumed to be global and available to all  
 7803 functions. All identifiers, global and local, have initial values of zero. Identifiers declared as auto  
 7804 shall be allocated on entry to the function and released on returning from the function. They  
 7805 therefore do not retain values between function calls. Auto arrays shall be specified by the array  
 7806 name followed by empty square brackets. On entry to a function, the old values of the names  
 7807 that appear as parameters and as automatic variables shall be pushed onto a stack. Until the  
 7808 function returns, reference to these names shall refer only to the new values.

7809 References to any of these names from other functions that are called from this function also  
 7810 refer to the new value until one of those functions uses the same name for a local variable.

7811 When a statement is an expression, unless the main operator is an assignment, execution of the  
 7812 statement shall write the value of the expression followed by a <newline> character.

7813 When a statement is a string, execution of the statement shall write the value of the string.

7814 Statements separated by semicolons or <newline> characters shall be executed sequentially. In  
 7815 an interactive invocation of *bc*, each time a <newline> character is read that satisfies the  
 7816 grammatical production:

7817 `input_item : semicolon_list NEWLINE`

7818 the sequential list of statements making up the **semicolon\_list** shall be executed immediately  
 7819 and any output produced by that execution shall be written without any delay due to buffering.

7820 In an **if** statement (**if**(*relation*) *statement*), the *statement* shall be executed if the relation is true.

7821 The **while** statement (**while**(*relation*) *statement*) implements a loop in which the *relation* is tested;  
 7822 each time the *relation* is true, the *statement* shall be executed and the *relation* retested. When the  
 7823 *relation* is false, execution shall resume after *statement*.

7824 A **for** statement (**for**(*expression*; *relation*; *expression*) *statement*) shall be the same as:

7825 `first-expression`  
 7826 `while (relation) {`  
 7827 `statement`  
 7828 `last-expression`  
 7829 `}`

7830 The application shall ensure that all three expressions are present.

7831 The **break** statement shall cause termination of a **for** or **while** statement.

7832 The **auto** statement (**auto** *identifier* [*identifier*] ...) shall cause the values of the identifiers to be  
 7833 pushed down. The identifiers can be ordinary identifiers or array identifiers. Array identifiers

7834 shall be specified by following the array name by empty square brackets. The application shall  
7835 ensure that the **auto** statement is the first statement in a function definition.

7836 A **define** statement:

```
7837 define LETTER (opt_parameter_list) {
7838 opt_auto_define_list
7839 statement_list
7840 }
```

7841 defines a function named **LETTER**. If a function named **LETTER** was previously defined, the  
7842 **define** statement shall replace the previous definition. The expression:

```
7843 LETTER (opt_argument_list)
```

7844 shall invoke the function named **LETTER**. The behavior is undefined if the number of  
7845 arguments in the invocation does not match the number of parameters in the definition.  
7846 Functions shall be defined before they are invoked. A function shall be considered to be defined  
7847 within its own body, so recursive calls are valid. The values of numeric constants within a  
7848 function shall be interpreted in the base specified by the value of the **ibase** register when the  
7849 function is invoked.

7850 The **return** statements (**return** and **return(expression)**) shall cause termination of a function,  
7851 popping of its auto variables, and specification of the result of the function. The first form shall  
7852 be equivalent to **return(0)**. The value and scale of the result returned by the function shall be the  
7853 value and scale of the expression returned.

7854 The **quit** statement (**quit**) shall stop execution of a *bc* program at the point where the statement  
7855 occurs in the input, even if it occurs in a function definition, or in an **if**, **for**, or **while** statement.

7856 The following functions shall be defined when the **-l** option is specified:

```
7857 s(expression)
7858 Sine of argument in radians.
```

```
7859 c(expression)
7860 Cosine of argument in radians.
```

```
7861 a(expression)
7862 Arctangent of argument.
```

```
7863 l(expression)
7864 Natural logarithm of argument.
```

```
7865 e(expression)
7866 Exponential function of argument.
```

```
7867 j(expression, expression)
7868 Bessel function of integer order.
```

7869 The scale of the result returned by these functions shall be the value of the **scale** register at the  
7870 time the function is invoked. The value of the **scale** register after these functions have completed  
7871 their execution shall be the same value it had upon invocation. The behavior is undefined if any  
7872 of these functions is invoked with an argument outside the domain of the mathematical  
7873 function.

#### 7874 EXIT STATUS

7875 The following exit values shall be returned:

7876 0 All input files were processed successfully.

7877 *unspecified* An error occurred.

## 7878 CONSEQUENCES OF ERRORS

7879 If any *file* operand is specified and the named file cannot be accessed, *bc* shall write a diagnostic message to standard error and terminate without any further action.

7881 In an interactive invocation of *bc*, the utility should print an error message and recover following any error in the input. In a non-interactive invocation of *bc*, invalid input causes undefined behavior.

## 7884 APPLICATION USAGE

7885 Automatic variables in *bc* do not work in exactly the same way as in either C or PL/1.

7886 For historical reasons, the exit status from *bc* cannot be relied upon to indicate that an error has occurred. Returning zero after an error is possible. Therefore, *bc* should be used primarily by interactive users (who can react to error messages) or by application programs that can somehow validate the answers returned as not including error messages.

7890 The *bc* utility always uses the period ( `.` ) character to represent a radix point, regardless of any decimal-point character specified as part of the current locale. In languages like C or *awk*, the period character is used in program source, so it can be portable and unambiguous, while the locale-specific character is used in input and output. Because there is no distinction between source and input in *bc*, this arrangement would not be possible. Using the locale-specific character in *bc*'s input would introduce ambiguities into the language; consider the following example in a locale with a comma as the decimal-point character:

```
7897 define f(a,b) {
7898 ...
7899 }
7900 ...
7901 f(1,2,3)
```

7902 Because of such ambiguities, the period character is used in input. Having input follow different conventions from output would be confusing in either pipeline usage or interactive usage, so the period is also used in output.

## 7905 EXAMPLES

7906 In the shell, the following assigns an approximation of the first ten digits of ' $\pi$ ' to the variable *x*:

```
7908 x=$(printf "%s\n" 'scale = 10; 104348/33215' | bc)
```

7909 The following *bc* program prints the same approximation of ' $\pi$ ', with a label, to standard output:

```
7911 scale = 10
7912 "pi equals "
7913 104348 / 33215
```

7914 The following defines a function to compute an approximate value of the exponential function (note that such a function is predefined if the `-l` option is specified):

```
7916 scale = 20
7917 define e(x){
7918 auto a, b, c, i, s
7919 a = 1
7920 b = 1
7921 s = 1
```



```

7922 for (i = 1; 1 == 1; i++){
7923 a = a*x
7924 b = b*i
7925 c = a/b
7926 if (c == 0) {
7927 return(s)
7928 }
7929 s = s+c
7930 }
7931 }

```

7932 The following prints approximate values of the exponential function of the first ten integers:

```

7933 for (i = 1; i <= 10; ++i) {
7934 e(i)
7935 }

```

### 7936 RATIONALE

7937 The *bc* utility is implemented historically as a front-end processor for *dc*; *dc* was not selected to  
 7938 be part of this volume of IEEE Std. 1003.1-200x because *bc* was thought to have a more intuitive  
 7939 programmatic interface. Current implementations that implement *bc* using *dc* are expected to be  
 7940 compliant.

7941 The exit status for error conditions has been left unspecified for several reasons:

- 7942 • The *bc* utility is used in both interactive and non-interactive situations. Different exit codes  
 7943 may be appropriate for the two uses.
- 7944 • It is unclear when a non-zero exit should be given; divide-by-zero, undefined functions, and  
 7945 syntax errors are all possibilities.
- 7946 • It is not clear what utility the exit status has.
- 7947 • In the 4.3 BSD, System V, and Ninth Edition implementations, *bc* works in conjunction with  
 7948 *dc*. The *dc* utility is the parent, *bc* is the child. This was done to cleanly terminate *bc* if *dc*  
 7949 aborted.

7950 The decision to have *bc* exit upon encountering an inaccessible input file is based on the belief  
 7951 that *bc file1 file2* is used most often when at least *file1* contains data/function  
 7952 declarations/initializations. Having *bc* continue with prerequisite files missing is probably not  
 7953 useful. There is no implication in the CONSEQUENCES OF ERRORS section that *bc* must check  
 7954 all its files for accessibility before opening any of them.

7955 There was considerable debate on the appropriateness of the language accepted by *bc*. Several  
 7956 reviewers preferred to see either a pure subset of the C language or some changes to make the  
 7957 language more compatible with C. While the *bc* language has some obvious similarities to C, it  
 7958 has never claimed to be compatible with any version of C. An interpreter for a subset of C might  
 7959 be a very worthwhile utility, and it could potentially make *bc* obsolete. However, no such utility  
 7960 is known in historical practice, and it was not within the scope of this volume of  
 7961 IEEE Std. 1003.1-200x to define such a language and utility. If and when they are defined, it may  
 7962 be appropriate to include them in a future version of this volume of IEEE Std. 1003.1-200x. This  
 7963 left the following alternatives:

- 7964 1. Exclude any calculator language from this volume of IEEE Std. 1003.1-200x.

7965 The consensus of the standard developers was that a simple programmatic calculator  
 7966 language is very useful for both applications and interactive users. The only arguments for  
 7967 excluding any calculator were that it would become obsolete if and when a C-compatible

7968 one emerged, or that the absence would encourage the development of such a C-  
7969 compatible one. These arguments did not sufficiently address the needs of current  
7970 application writers.

7971 2. Standardize the historical *dc*, possibly with minor modifications.

7972 The consensus of the standard developers was that *dc* is a fundamentally less usable  
7973 language and that that would be far too severe a penalty for avoiding the issue of being  
7974 similar to but incompatible with C.

7975 3. Standardize the historical *bc*, possibly with minor modifications.

7976 This was the approach taken. Most of the proponents of changing the language would not  
7977 have been satisfied until most or all of the incompatibilities with C were resolved. Since  
7978 most of the changes considered most desirable would break historical applications and  
7979 require significant modification to historical implementations, almost no modifications  
7980 were made. The one significant modification that was made was the replacement of the  
7981 historical *bc* assignment operators "=", and so on, with the more modern "+=", and so  
7982 on. The older versions are considered to be fundamentally flawed because of the lexical  
7983 ambiguity in uses like  $a=-1$ .

7984 In order to permit implementations to deal with backwards compatibility as they see fit,  
7985 the behavior of this one ambiguous construct was made undefined. (At least three  
7986 implementations have been known to support this change already, so the degree of change  
7987 involved should not be great.)

7988 The '%' operator is the mathematical remainder operator when **scale** is zero. The behavior of  
7989 this operator for other values of **scale** is from historical implementations of *bc*, and has been  
7990 maintained for the sake of historical applications despite its non-intuitive nature.

7991 Historical implementations permit setting **ibase** and **obase** to a broader range of values. This  
7992 includes values less than 2, which were not seen as sufficiently useful to standardize. These  
7993 implementations do not interpret input properly for values of **ibase** that are greater than 16. This  
7994 is because numeric constants are recognized syntactically, rather than lexically, as described in  
7995 this volume of IEEE Std. 1003.1-200x. They are built from lexical tokens of single hexadecimal  
7996 digits and periods. Since <blank>s between tokens are not visible at the syntactic level, it is not  
7997 possible to recognize the multi-digit "digits" used in the higher bases properly. The ability to  
7998 recognize input in these bases was not considered useful enough to require modifying these  
7999 implementations. Note that the recognition of numeric constants at the syntactic level is not a  
8000 problem with conformance to this volume of IEEE Std. 1003.1-200x, as it does not impact the  
8001 behavior of portable applications (and correct *bc* programs). Historical implementations also  
8002 accept input with all of the digits '0'-'9' and 'A'-'F' regardless of the value of **ibase**; since  
8003 digits with value greater than or equal to **ibase** are not really appropriate, the behavior when  
8004 they appear is undefined, except for the common case of:

```
8005 ibase=8;
8006 /* Process in octal base. */
8007 ...
8008 ibase=A
8009 /* Restore decimal base. */
```

8010 In some historical implementations, if the expression to be written is an uninitialized array  
8011 element, a leading <space> character and/or up to four leading 0 characters may be output  
8012 before the character zero. This behavior is considered a bug; it is unlikely that any currently  
8013 portable application relies on:

- 8014 `echo 'b[3]' | bc`
- 8015 returning 0000 rather than 0.
- 8016 Exact calculation of the number of fractional digits to output for a given value in a base other  
8017 than 10 can be computationally expensive. Historical implementations use a faster  
8018 approximation, and this is permitted. Note that the requirements apply only to values of **obase**  
8019 that this volume of IEEE Std. 1003.1-200x requires implementations to support (in particular, not  
8020 to 1, 0, or negative bases, if an implementation supports them as an extension).
- 8021 Historical implementations of *bc* did not allow array parameters to be passed as the last  
8022 parameter to a function. New implementations are encouraged to remove this restriction even  
8023 though it is not required by the grammar.
- 8024 **FUTURE DIRECTIONS**
- 8025 None.
- 8026 **SEE ALSO**
- 8027 *awk*
- 8028 **CHANGE HISTORY**
- 8029 First released in Issue 4.
- 8030 **Issue 5**
- 8031 FUTURE DIRECTIONS section added.
- 8032 **Issue 6**
- 8033 Updated to align with the IEEE P1003.2b draft standard, which included resolution of several  
8034 interpretations of the ISO POSIX-2: 1993 standard.
- 8035 The normative text is reworded to avoid use of the term “must” for application requirements.

8036 **NAME**

8037           bg — run jobs in the background

8038 **SYNOPSIS**8039 UP        bg [ *job\_id* ... ]

8040

8041 **DESCRIPTION**

8042           If job control is enabled (see the description of *set -m*), the *bg* utility shall resume suspended jobs  
 8043           from the current environment (see Section 2.13 (on page 2273)) by running them as background  
 8044           jobs. If the job specified by *job\_id* is already a running background job, the *bg* utility shall have no  
 8045           effect and shall exit successfully.

8046           Using *bg* to place a job into the background shall cause its process ID to become “known in the  
 8047           current shell execution environment”, as if it had been started as an asynchronous list; see  
 8048           Section 2.9.3.1 (on page 2259).

8049 **OPTIONS**

8050           None.

8051 **OPERANDS**

8052           The following operand shall be supported:

8053           *job\_id*       Specify the job to be resumed as a background job. If no *job\_id* operand is given,  
 8054           the most recently suspended job shall be used. The format of *job\_id* is described in  
 8055           the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.205, Job Control  
 8056           Job ID.

8057 **STDIN**

8058           Not used.

8059 **INPUT FILES**

8060           None.

8061 **ENVIRONMENT VARIABLES**8062           The following environment variables shall affect the execution of *bg*:

8063           *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 8064           If *LANG* is unset or null, the corresponding value from the implementation-  
 8065           defined default locale shall be used. If any of the internationalization variables  
 8066           contains an invalid setting, the utility shall behave as if none of the variables had  
 8067           been defined.

8068           *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 8069           internationalization variables.

8070           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 8071           characters (for example, single-byte as opposed to multi-byte characters in  
 8072           arguments).

8073           *LC\_MESSAGES*

8074           Determine the locale that should be used to affect the format and contents of  
 8075           diagnostic messages written to standard error.

8076 XSI        *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

8077 **ASYNCHRONOUS EVENTS**

8078 Default.

8079 **STDOUT**8080 The output of *bg* shall consist of a line in the format:8081 "[%d] %s\n", <*job-number*>, <*command*>

8082 where the fields are as follows:

8083 <*job-number*> A number that can be used to identify the job to the *wait*, *fg*, and *kill* utilities. Using  
8084 these utilities, the job can be identified by prefixing the job number with '% '.8085 <*command*> The associated command that was given to the shell.8086 **STDERR**

8087 Used only for diagnostic messages.

8088 **OUTPUT FILES**

8089 None.

8090 **EXTENDED DESCRIPTION**

8091 None.

8092 **EXIT STATUS**

8093 The following exit values shall be returned:

8094 0 Successful completion.

8095 &gt;0 An error occurred.

8096 **CONSEQUENCES OF ERRORS**8097 If job control is disabled, the *bg* utility shall exit with an error and no job shall be placed in the  
8098 background.8099 **APPLICATION USAGE**8100 A job is generally suspended by typing the SUSP character (<control>-Z on most systems); see  
8101 the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface. At  
8102 that point, *bg* can put the job into the background. This is most effective when the job is  
8103 expecting no terminal input and its output has been redirected to non-terminal files. A  
8104 background job can be forced to stop when it has terminal output by issuing the command:8105 `stty tostop`

8106 A background job can be stopped with the command:

8107 `kill -s stop job ID`8108 The *bg* utility does not work as expected when it is operating in its own utility execution  
8109 environment because that environment has no suspended jobs. In the following examples:8110 ... | xargs bg  
8111 (bg)8112 each *bg* operates in a different environment and does not share its parent shell's understanding  
8113 of jobs. For this reason, *bg* is generally implemented as a shell regular built-in.8114 **EXAMPLES**

8115 None.

8116 **RATIONALE**

8117 The extensions to the shell specified in this volume of IEEE Std. 1003.1-200x have mostly been  
8118 based on features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs*  
8119 are also based on the KornShell. The standard developers examined the characteristics of the C  
8120 shell versions of these utilities and found that differences exist. Despite widespread use of the C  
8121 shell, the KornShell versions were selected for this volume of IEEE Std. 1003.1-200x to maintain a  
8122 degree of uniformity with the rest of the KornShell features selected (such as the very popular  
8123 command line editing features).

8124 The *bg* utility is expected to wrap its output if the output exceeds the number of display  
8125 columns.

8126 **FUTURE DIRECTIONS**

8127 None.

8128 **SEE ALSO**

8129 *fg*, *kill*, *jobs*, *wait*

8130 **CHANGE HISTORY**

8131 First released in Issue 4.

8132 **Issue 6**

8133 This utility is now marked as part of the User Portability Utilities option.

8134 The JC margin marker on the SYNOPSIS is removed since support for Job Control is mandatory  
8135 in this issue. This is a FIPS requirement.

## 8136 NAME

8137 c99 — compile standard C programs

## 8138 SYNOPSIS

```
8139 CD c99 [-c][-D name[=value]]...[-E][-g][-I directory] ... [-L directory]
8140 ... [-o outfile][-O][-s][-U name]... operand ...
8141
```

## 8142 DESCRIPTION

8143 The *c99* utility is an interface to the standard C compilation system; it shall accept source code  
 8144 conforming to the ISO C standard. The system conceptually consists of a compiler and link  
 8145 editor. The files referenced by *operands* shall be compiled and linked to produce an executable  
 8146 file. (It is unspecified whether the linking occurs entirely within the operation of *c99*; some  
 8147 systems may produce objects that are not fully resolved until the file is executed.)

8148 If the **-c** option is specified, for all path name operands of the form *file.c*, the files:

8149  $\$(\text{basename } \textit{pathname} \textit{.c}).\textit{o}$

8150 shall be created as the result of successful compilation. If the **-c** option is not specified, it is  
 8151 unspecified whether such *.o* files are created or deleted for the *file.c* operands.

8152 If there are no options that prevent link editing (such as **-c** or **-E**), and all operands compile and  
 8153 link without error, the resulting executable file shall be written according to the **-o outfile** option  
 8154 (if present) or to the file **a.out**.

8155 The executable file shall be created as specified in Section 1.7.1.4 (on page 2209), except that the  
 8156 file permission bits shall be set to:

8157 S\_IRWXO | S\_IRWXG | S\_IRWXU

8158 and the bits specified by the *umask* of the process shall be cleared.

## 8159 OPTIONS

8160 The *c99* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 8161 12.2, Utility Syntax Guidelines, except that:

- 8162 • The **-I library** operands have the format of options, but their position within a list of  
 8163 operands affects the order in which libraries are searched.
- 8164 • The order of specifying the **-I** and **-L** options is significant.
- 8165 • Portable applications shall specify each option separately; that is, grouping option letters (for  
 8166 example, **-cO**) need not be recognized by all implementations.

8167 The following options shall be supported:

8168 **-c** Suppress the link-edit phase of the compilation, and do not remove any object files  
 8169 that are produced.

8170 **-g** Produce symbolic information in the object or executable files; the nature of this  
 8171 information is unspecified, and may be modified by implementation-defined  
 8172 interactions with other options.

8173 **-s** Produce object or executable files, or both, from which symbolic and other  
 8174 information not required for proper execution using the *exec* family defined in the  
 8175 System Interfaces volume of IEEE Std. 1003.1-200x, has been removed (stripped). If  
 8176 both **-g** and **-s** options are present, the action taken is unspecified.

8177 **-o outfile** Use the path name *outfile*, instead of the default **a.out**, for the executable file  
 8178 produced. If the **-o** option is present with **-c** or **-E**, the result is unspecified.

- 8179        **-D** *name*[=*value*]  
 8180            Define *name* as if by a C-language **#define** directive. If no *=value* is given, a value of  
 8181            1 shall be used. The **-D** option has lower precedence than the **-U** option. That is, if  
 8182            *name* is used in both a **-U** and a **-D** option, *name* shall be undefined regardless of  
 8183            the order of the options. Additional implementation-defined *names* may be  
 8184            provided by the compiler. Implementations shall support at least 2 048 bytes of **-D**  
 8185            definitions and 256 *names*.
- 8186        **-E**            Copy C-language source files to standard output, expanding all preprocessor  
 8187            directives; no compilation shall be performed. If any operand is not a text file, the  
 8188            effects are unspecified.
- 8189        **-I** *directory*   Change the algorithm for searching for headers whose names are not absolute path  
 8190            names to look in the directory named by the *directory* path name before looking in  
 8191            the usual places. Thus, headers whose names are enclosed in double-quotes (" ")  
 8192            shall be searched for first in the directory of the file with the **#include** line, then in  
 8193            directories named in **-I** options, and last in the usual places. For headers whose  
 8194            names are enclosed in angle brackets ("< >"), the header shall be searched for only  
 8195            in directories named in **-I** options and then in the usual places. Directories named  
 8196            in **-I** options shall be searched in the order specified. Implementations shall  
 8197            support at least ten instances of this option in a single *c99* command invocation.
- 8198        **-L** *directory*   Change the algorithm of searching for the libraries named in the **-I** objects to look  
 8199            in the directory named by the *directory* path name before looking in the usual  
 8200            places. Directories named in **-L** options shall be searched in the order specified.  
 8201            Implementations shall support at least ten instances of this option in a single *c99*  
 8202            command invocation. If a directory specified by a **-L** option contains files named  
 8203            **libc.a**, **libm.a**, **libl.a**, or **liby.a**, the results are unspecified.
- 8204        **-O**            Optimize. The nature of the optimization is unspecified.
- 8205        **-U** *name*        Remove any initial definition of *name*.
- 8206        Multiple instances of the **-D**, **-I**, **-U**, and **-L** options can be specified.

## 8207 OPERANDS

8208        An *operand* is either in the form of a path name or the form **-I** *library*. The application shall  
 8209        ensure that at least one operand of the path name form is specified. The following operands shall  
 8210        be supported:

- 8211        *file.c*        A C-language source file to be compiled and optionally linked. The application  
 8212        shall ensure that the operand is of this form if the **-c** option is used.
- 8213        *file.a*        A library of object files typically produced by the *ar* utility, and passed directly to  
 8214        the link editor. Implementations may recognize implementation-defined suffixes  
 8215        other than **.a** as denoting object file libraries.
- 8216        *file.o*        An object file produced by *c99* **-c** and passed directly to the link editor.  
 8217        Implementations may recognize implementation-defined suffixes other than **.o** as  
 8218        denoting object files.

8219        The processing of other files is implementation-defined.

8220        **-I** *library*   (The letter ell.) Search the library named:

8221            *liblibrary.a*

8222        A library shall be searched when its name is encountered, so the placement of a **-I**  
 8223        operand is significant. Several standard libraries can be specified in this manner, as



8224 described in the EXTENDED DESCRIPTION section. Implementations may  
8225 recognize implementation-defined suffixes other than `.a` as denoting libraries.

#### 8226 **STDIN**

8227 Not used.

#### 8228 **INPUT FILES**

8229 The input file shall be one of the following: a text file containing a C-language source program,  
8230 an object file in the format produced by `c99 -c`, or a library of object files, in the format produced  
8231 by archiving zero or more object files, using `ar`. Implementations may supply additional utilities  
8232 that produce files in these formats. Additional input file formats are implementation-defined.

#### 8233 **ENVIRONMENT VARIABLES**

8234 The following environment variables shall affect the execution of `c99`:

8235 **LANG** Provide a default value for the internationalization variables that are unset or null.  
8236 If `LANG` is unset or null, the corresponding value from the implementation-  
8237 defined default locale shall be used. If any of the internationalization variables  
8238 contains an invalid setting, the utility shall behave as if none of the variables had  
8239 been defined.

8240 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
8241 internationalization variables.

8242 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
8243 characters (for example, single-byte as opposed to multi-byte characters in  
8244 arguments and input files).

#### 8245 **LC\_MESSAGES**

8246 Determine the locale that should be used to affect the format and contents of  
8247 diagnostic messages written to standard error.

8248 **XSI** **NLSPATH** Determine the location of message catalogs for the processing of `LC_MESSAGES`.

8249 **TMPDIR** Provide a path name that should override the default directory for temporary files,  
8250 **XSI** if any. On XSI-conforming systems, provide a path name that shall override the  
8251 default directory for temporary files, if any.

#### 8252 **ASYNCHRONOUS EVENTS**

8253 Default.

#### 8254 **STDOUT**

8255 If more than one *file* operand ending in `.c` (or possibly other unspecified suffixes) is given, for  
8256 each such file:

8257 `"%s:\n", <file>`

8258 may be written. These messages, if written, shall precede the processing of each input file; they  
8259 shall not be written to the standard output if they are written to the standard error, as described  
8260 in the `STDERR` section.

8261 If the `-E` option is specified, the standard output shall be a text file that represents the results of  
8262 the preprocessing stage of the language; it may contain extra information appropriate for  
8263 subsequent compilation passes.

#### 8264 **STDERR**

8265 Used only for diagnostic messages. If more than one *file* operand ending in `.c` (or possibly other  
8266 unspecified suffixes) is given, for each such file:

8267 "%s:\n", <file>

8268 may be written to allow identification of the diagnostic and warning messages with the  
8269 appropriate input file. These messages, if written, shall precede the processing of each input file;  
8270 they shall not be written to the standard error if they are written to the standard output, as  
8271 described in the STDOUT section.

8272 This utility may produce warning messages about certain conditions that do not warrant  
8273 returning an error (non-zero) exit value.

## 8274 OUTPUT FILES

8275 Object files or executable files or both are produced in unspecified formats.

## 8276 EXTENDED DESCRIPTION

### 8277 Standard Libraries

8278 The *c99* utility shall recognize the following **-l** operands for standard libraries:

8279 **-l c** This operand shall make visible all library functions referenced in the System  
8280 Interfaces volume of IEEE Std. 1003.1-200x, with the possible exception of those  
8281 functions listed as residing in < aio.h>, < arpa/inet.h>, < math.h>, < mqueue.h>,  
8282 < netdb.h>, < netinet/in.h>, < pthread.h>, < sched.h>, < semaphore.h>,  
8283 < sys/socket.h>, *pthread\_atfork()* in < unistd.h>, and those functions marked as an  
8284 RT extension in < sys/mman.h> and < time.h>. This operand shall not be required  
8285 to be present to cause a search of this library.

8286 **-l l** This operand shall make visible all functions required by the C-language output of  
8287 *lex* that are not made available through the **-l c** operand.

8288 **-l pthread** This operand shall make visible all functions referenced in < pthread.h> and  
8289 *pthread\_atfork()* referenced in < unistd.h>. An implementation may search this  
8290 library in the absence of this operand.

8291 **-l m** This operand shall make visible all functions referenced in < math.h>. An  
8292 implementation may search this library in the absence of this operand.

8293 **-l rt** This operand shall make visible all functions referenced in < aio.h>, < mqueue.h>,  
8294 < sched.h>, and < semaphore.h>, and those functions marked as an RT extension in  
8295 < sys/mman.h> and < time.h>. An implementation may search this library in the  
8296 absence of this operand.

8297 **-l xnet** This operand makes visible all functions referenced in < arpa/inet.h>, < netdb.h>,  
8298 < netinet/in.h>, and < sys/socket.h>. An implementation may search this library in  
8299 the absence of this operand.

8300 **-l y** This operand shall make visible all functions required by the C-language output of  
8301 *yacc* that are not made available through the **-l c** operand.

8302 In the absence of options that inhibit invocation of the link editor, such as **-c** or **-E**, the *c99* utility  
8303 shall cause the equivalent of a **-l c** operand to be passed to the link editor as the last **-l** operand,  
8304 causing it to be searched after all other object files and libraries are loaded.

8305 It is unspecified whether the libraries **libc.a**, **libm.a**, **librt.a**, **libpthread.a**, **libl.a**, **liby.a**, or **libxnet**  
8306 exist as regular files. The implementation may accept as **-l** operands names of objects that do  
8307 not exist as regular files.

8308 **External Symbols**

8309 The C compiler and link editor shall support the significance of external symbols up to a length  
 8310 of at least 31 bytes; the action taken upon encountering symbols exceeding the implementation-  
 8311 defined maximum symbol length is unspecified.

8312 The compiler and link editor shall support a minimum of 511 external symbols per source or  
 8313 object file, and a minimum of 4095 external symbols in total. A diagnostic message shall be  
 8314 written to the standard output if the implementation-defined limit is exceeded; other actions are  
 8315 unspecified.

8316 **Programming Environments**

8317 All implementations shall support one of the following programming environments as a default.  
 8318 Implementations may support more than one of the following programming environments.  
 8319 Applications can use *sysconf()* or *getconf* to determine which programming environments are  
 8320 supported.

8321 **Table 4-4 Programming Environments: Type Sizes**

| 8322 | <b>Programming Environment</b> | <b>Bits in</b> | <b>Bits in</b> | <b>Bits in</b> | <b>Bits in</b> |
|------|--------------------------------|----------------|----------------|----------------|----------------|
| 8323 | <i>getconf</i> Name            | <b>int</b>     | <b>long</b>    | <b>pointer</b> | <b>off_t</b>   |
| 8324 | _POSIX_V6_ILP32_OFF32          | 32             | 32             | 32             | 32             |
| 8325 | _POSIX_V6_ILP32_OFFBIG         | 32             | 32             | 32             | ≥64            |
| 8326 | _POSIX_V6_LP64_OFF64           | 32             | 64             | 64             | 64             |
| 8327 | _POSIX_V6_LP64_OFFBIG          | ≥32            | ≥64            | ≥64            | ≥64            |

8328 **Notes to Reviewers**

8329 *This section with side shading will not appear in the final copy. - Ed.*

8330 The names of the macros above may be changed. This has been added to the issues list.

8331 Implementations provide configuration strings for C compiler flags, linker/loader flags, and  
 8332 libraries for each supported environment. When an application needs to use a specific  
 8333 programming environment rather than the implementation default programming environment  
 8334 while compiling, the application shall first verify that the implementation supports the desired  
 8335 environment. If the desired programming environment is supported, the application shall then  
 8336 invoke *c99* with the appropriate C compiler flags as the first options for the compile, the  
 8337 appropriate linker/loader flags after any other options but before any operands, and the  
 8338 appropriate libraries at the end of the operands.

8339 Portable applications shall not attempt to link together object files compiled for different  
 8340 programming models. Applications shall also be aware that binary data placed in shared  
 8341 memory or in files might not be recognized by applications built for other programming models.

8342 **Table 4-5** Programming Environments: c99 and cc Arguments

| 8343 | <b>Programming Environment</b> |                     | <b>c99 and cc Arguments</b>   |
|------|--------------------------------|---------------------|-------------------------------|
| 8344 | <i>getconf</i> Name            | Use                 | <i>getconf</i> Name           |
| 8345 | _POSIX_V6_ILP32_OFF32          | C Compiler Flags    | POSIX_V6_ILP32_OFF32_CFLAGS   |
| 8346 |                                | Linker/Loader Flags | POSIX_V6_ILP32_OFF32_LDFLAGS  |
| 8347 |                                | Libraries           | POSIX_V6_ILP32_OFF32_LIBS     |
| 8348 | _POSIX_V6_ILP32_OFFBIG         | C Compiler Flags    | POSIX_V6_ILP32_OFFBIG_CFLAGS  |
| 8349 |                                | Linker/Loader Flags | POSIX_V6_ILP32_OFFBIG_LDFLAGS |
| 8350 |                                | Libraries           | POSIX_V6_ILP32_OFFBIG_LIBS    |
| 8351 | _POSIX_V6_LP64_OFF64           | C Compiler Flags    | POSIX_V6_LP64_OFF64_CFLAGS    |
| 8352 |                                | Linker/Loader Flags | POSIX_V6_LP64_OFF64_LDFLAGS   |
| 8353 |                                | Libraries           | POSIX_V6_LP64_OFF64_LIBS      |
| 8354 | _POSIX_V6_LPBIG_OFFBIG         | C Compiler Flags    | POSIX_V6_LPBIG_OFFBIG_CFLAGS  |
| 8355 |                                | Linker/Loader Flags | POSIX_V6_LPBIG_OFFBIG_LDFLAGS |
| 8356 |                                | Libraries           | POSIX_V6_LPBIG_OFFBIG_LIBS    |

8357 **Notes to Reviewers**8358 *This section with side shading will not appear in the final copy. - Ed.*

8359 The names of the macros above may be changed. This has been added to the issues list.

8360 **EXIT STATUS**

8361 The following exit values shall be returned:

8362 0 Successful compilation or link edit.

8363 &gt;0 An error occurred.

8364 **CONSEQUENCES OF ERRORS**

8365 When *c99* encounters a compilation error that causes an object file not to be created, it shall write  
 8366 a diagnostic to standard error and continue to compile other source code operands, but it shall  
 8367 not perform the link phase and return a non-zero exit status. If the link edit is unsuccessful, a  
 8368 diagnostic message shall be written to standard error and *c99* exits with a non-zero status. A  
 8369 portable application shall rely on the exit status of *c99*, rather than on the existence or mode of  
 8370 the executable file.

8371 **APPLICATION USAGE**

8372 Since the *c99* utility usually creates files in the current directory during the compilation process,  
 8373 it is typically necessary to run the *c99* utility in a directory in which a file can be created.

8374 On systems providing POSIX Conformance (see the Base Definitions volume of  
 8375 IEEE Std. 1003.1-200x, Chapter 2, Conformance), *c99* is required only with the the C-Language  
 8376 Development option; XSI-conformant systems always provide *c99*.

8377 Some historical implementations have created *.o* files when *-c* is not specified and more than  
 8378 one source file is given. Since this area is left unspecified, the application cannot rely on *.o* files  
 8379 being created, but it also must be prepared for any related *.o* files that already exist being deleted  
 8380 at the completion of the link edit.

8381 Some historical implementations have permitted *-L* options to be interspersed with *-I* operands  
 8382 on the command line. For an application to compile consistently on systems that do not behave  
 8383 like this, it is necessary for a portable application to supply all *-L* options before any of the *-I*  
 8384 options.

8385 There is the possible implication that if a user supplies versions of the standard library functions  
 8386 (before they would be encountered by an implicit `-l c` or explicit `-l m`), that those versions  
 8387 would be used in place of the standard versions. There are various reasons this might not be  
 8388 true (functions defined as macros, manipulations for clean name space, and so on), so the  
 8389 existence of files named in the same manner as the standard libraries within the `-L` directories is  
 8390 explicitly stated to produce unspecified behavior.

8391 All of the functions specified in the System Interfaces volume of IEEE Std. 1003.1-200x may be  
 8392 made visible by implementations when the Standard C Library is searched. Portable applications  
 8393 must explicitly request searching the other standard libraries when functions made visible by  
 8394 those libraries are used.

#### 8395 EXAMPLES

8396 1. The following usage example compiles `foo.c` and creates the executable file `foo`:

```
8397 c99 -o foo foo.c
```

8398 The following usage example compiles `foo.c` and creates the object file `foo.o`:

```
8399 c99 -c foo.c
```

8400 The following usage example compiles `foo.c` and creates the executable file `a.out`:

```
8401 c99 foo.c
```

8402 The following usage example compiles `foo.c`, links it with `bar.o`, and creates the executable  
 8403 file `a.out`. It also creates and leaves `foo.o`:

```
8404 c99 foo.c bar.o
```

8405 2. The following example shows how an application using threads interfaces can test for  
 8406 support of and use a programming environment supporting 32-bit `int`, `long`, and `pointer`  
 8407 types and an `off_t` type using at least 64 bits:

```
8408 if [$(getconf _POSIX_V6_ILP32_OFFBIG) != "-1"]
8409 then
8410 c99 $(getconf POSIX_V6_ILP32_OFFBIG_CFLAGS) -D_XOPEN_SOURCE=600 \
8411 $(getconf POSIX_V6_ILP32_OFFBIG_LDFLAGS) foo.c -o foo \
8412 $(getconf POSIX_V6_ILP32_OFFBIG_LIBS) -l pthread
8413 else
8414 echo ILP32_OFFBIG programming environment not supported
8415 exit 1
8416 fi
```

#### 8417 **Notes to Reviewers**

8418 *This section with side shading will not appear in the final copy. - Ed.*

8419 The names of the macros above may be changed. This has been added to the issues list.

8420 3. The following examples clarify the use and interactions of `-L` options and `-l` operands.

8421 Consider the case in which module `a.c` calls function `f()` in library `libQ.a`, and module `b.c`  
8422 calls function `g()` in library `libp.a`. Assume that both libraries reside in `/a/b/c`. The  
8423 command line to compile and link in the desired way is:

```
8424 c99 -L /a/b/c main.o a.c -l Q b.c -l p
```

8425 In this case the `-l Q` operand need only precede the first `-l p` operand, since both `libQ.a`  
8426 and `libp.a` reside in the same directory.

8427 Multiple `-L` operands can be used when library name collisions occur. Building on the  
8428 previous example, suppose that the user wants to use a new `libp.a`, in `/a/a/a`, but still wants  
8429 `f()` from `/a/b/c/libQ.a`:

```
8430 c99 -L /a/a/a -L /a/b/c main.o a.c -l Q b.c -l p
```

8431 In this example, the linker searches the `-L` options in the order specified, and finds  
8432 `/a/a/a/libp.a` before `/a/b/c/libp.a` when resolving references for `b.c`. The order of the `-l`  
8433 operands is still important, however.

#### 8434 RATIONALE

8435 The `c99` utility is based on the `c89` utility originally introduced in the ISO POSIX-2: 1993 standard. |

#### 8436 FUTURE DIRECTIONS

8437 None.

#### 8438 SEE ALSO

8439 `ar`, `getconf`, `make`, `nm`, `strip`, `umask`, the System Interfaces volume of IEEE Std. 1003.1-200x,  
8440 `sysconf()`

#### 8441 CHANGE HISTORY

8442 First released in Issue 6. Included for alignment with the ISO/IEC 9899: 1999 standard. |

8443 **NAME**

8444 cal — print a calendar

8445 **SYNOPSIS**8446 xSI cal [[*month*] *year* ]

8447

8448 **DESCRIPTION**

8449 The *cal* utility shall write a calendar to standard output using the Julian calendar for dates from  
 8450 January 1, 1 through September 2, 1752 and the Gregorian calendar for dates from September 14,  
 8451 1752 through December 31, 9999 as though the Gregorian calendar had been adopted on  
 8452 September 14, 1752.

8453 **OPTIONS**

8454 None.

8455 **OPERANDS**

8456 The following operands shall be supported:

8457 *month* Specify the month to be displayed, represented as a decimal integer from 1  
 8458 (January) to 12 (December). The default shall be the current month.

8459 *year* Specify the year for which the calendar is displayed, represented as a decimal  
 8460 integer from 1 to 9999. The default shall be the current year.

8461 **STDIN**

8462 Not used.

8463 **INPUT FILES**

8464 None.

8465 **ENVIRONMENT VARIABLES**8466 The following environment variables shall affect the execution of *cal*:

8467 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 8468 If *LANG* is unset or null, the corresponding value from the implementation-  
 8469 defined default locale shall be used. If any of the internationalization variables  
 8470 contains an invalid setting, the utility shall behave as if none of the variables had  
 8471 been defined.

8472 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 8473 internationalization variables.

8474 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 8475 characters (for example, single-byte as opposed to multi-byte characters in  
 8476 arguments).

8477 *LC\_MESSAGES*

8478 Determine the locale that should be used to affect the format and contents of  
 8479 diagnostic messages written to standard error, and informative messages written  
 8480 to standard output.

8481 *LC\_TIME* Determine the format and contents of the calendar.

8482 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

8483 *TZ* Determine the timezone used to calculate the value of the current month.

8484 **ASYNCHRONOUS EVENTS**

8485 Default.

8486 **STDOUT**

8487 The standard output shall be used to display the calendar, in an unspecified format.

8488 **STDERR**

8489 Used only for diagnostic messages.

8490 **OUTPUT FILES**

8491 None.

8492 **EXTENDED DESCRIPTION**

8493 None.

8494 **EXIT STATUS**

8495 The following exit values shall be returned:

8496 0 Successful completion.

8497 &gt;0 An error occurred.

8498 **CONSEQUENCES OF ERRORS**

8499 Default.

8500 **APPLICATION USAGE**

8501 Note that:

8502 cal 83

8503 refers to A.D. 83, not 1983.

8504 **EXAMPLES**

8505 None.

8506 **RATIONALE**

8507 None.

8508 **FUTURE DIRECTIONS**8509 A future revision of IEEE Std. 1003.1-200x may support locale-specific recognition of the date of  
8510 adoption of the Gregorian calendar.8511 **SEE ALSO**

8512 None.

8513 **CHANGE HISTORY**

8514 First released in Issue 2.

8515 **Issue 4**

8516 Format reorganized.

8517 Internationalized environment variable support mandated.

8518 **Issue 6**8519 The DESCRIPTION is updated to allow for traditional behavior for years before the adoption of  
8520 the Gregorian calendar.



8521 **NAME**8522 `cat` — concatenate and print files8523 **SYNOPSIS**8524 `cat [-u][file ...]`8525 **DESCRIPTION**8526 The `cat` utility reads files in sequence and writes their contents to the standard output in the  
8527 same sequence.8528 **OPTIONS**8529 The `cat` utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
8530 12.2, Utility Syntax Guidelines.

8531 The following option shall be supported:

8532 **-u** Write bytes from the input file to the standard output without delay as each is  
8533 read.8534 **OPERANDS**

8535 The following operand shall be supported:

8536 **file** A path name of an input file. If no *file* operands are specified, the standard input is  
8537 used. If a *file* is '-', the `cat` utility shall read from the standard input at that point  
8538 in the sequence. The `cat` utility shall not close and reopen standard input when it is  
8539 referenced in this way, but shall accept multiple occurrences of '-' as a *file*  
8540 operand.8541 **STDIN**8542 The standard input is used only if no *file* operands are specified, or if a *file* operand is '-'. See  
8543 the INPUT FILES section.8544 **INPUT FILES**

8545 The input files can be any file type.

8546 **ENVIRONMENT VARIABLES**8547 The following environment variables shall affect the execution of `cat`:8548 **LANG** Provide a default value for the internationalization variables that are unset or null.  
8549 If *LANG* is unset or null, the corresponding value from the implementation-  
8550 defined default locale shall be used. If any of the internationalization variables  
8551 contains an invalid setting, the utility shall behave as if none of the variables had  
8552 been defined.8553 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
8554 internationalization variables.8555 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
8556 characters (for example, single-byte as opposed to multi-byte characters in  
8557 arguments).8558 **LC\_MESSAGES**8559 Determine the locale that should be used to affect the format and contents of  
8560 diagnostic messages written to standard error.8561 **XSI** **NLS\_PATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

8562 **ASYNCHRONOUS EVENTS**

8563 Default.

8564 **STDOUT**8565 The standard output shall contain the sequence of bytes read from the input files. Nothing else  
8566 shall be written to the standard output.8567 **STDERR**

8568 Used only for diagnostic messages.

8569 **OUTPUT FILES**

8570 None.

8571 **EXTENDED DESCRIPTION**

8572 None.

8573 **EXIT STATUS**

8574 The following exit values shall be returned:

8575 0 All input files were output successfully.

8576 &gt;0 An error occurred.

8577 **CONSEQUENCES OF ERRORS**

8578 Default.

8579 **APPLICATION USAGE**8580 The **-u** option has value in prototyping non-blocking reads from FIFOs. The intent is to support  
8581 the following sequence:8582 `mkfifo foo`8583 `cat -u foo > /dev/tty13 &`8584 `cat -u > foo`8585 It is unspecified whether standard output is or is not buffered in the default case. This is  
8586 sometimes of interest when standard output is associated with a terminal, since buffering may  
8587 delay the output. The presence of the **-u** option guarantees that unbuffered I/O is available. It is  
8588 implementation-defined whether the *cat* utility buffers output if the **-u** option is not specified.  
8589 Traditionally, the **-u** option is implemented using the equivalent of the *setvbuf()* function  
8590 defined in the System Interfaces volume of IEEE Std. 1003.1-200x.8591 **EXAMPLES**

8592 The following command:

8593 `cat myfile`8594 writes the contents of the file **myfile** to standard output.

8595 The following command:

8596 `cat doc1 doc2 > doc.all`8597 concatenates the files **doc1** and **doc2** and writes the result to **doc.all**.8598 Because of the shell language mechanism used to perform output redirection, a command such  
8599 as this:8600 `cat doc doc.end > doc`8601 causes the original data in **doc** to be lost.

8602 The command:

8603 `cat start - middle - end > file`

8604 when standard input is a terminal, gets two arbitrary pieces of input from the terminal with a  
8605 single invocation of *cat*. Note, however, that if standard input is a regular file, this would be  
8606 equivalent to the command:

8607 `cat start - middle /dev/null end > file`

8608 because the entire contents of the file would be consumed by *cat* the first time '-' was used as a  
8609 *file* operand and an end-of-file condition would be detected immediately when '-' was  
8610 referenced the second time.

#### 8611 RATIONALE

8612 Historical versions of the *cat* utility include the options `-e`, `-t`, and `-v`, which permit the ends of  
8613 lines, `<tab>`s, and invisible characters, respectively, to be rendered visible in the output. The  
8614 standard developers omitted these options because they provide too fine a degree of control  
8615 over what is made visible, and similar output can be obtained using a command such as:

8616 `sed -n -e 's/$/$/' -e l pathname`

8617 The `-s` option was omitted because it corresponds to different functions in BSD and System V-  
8618 based systems. The BSD `-s` option to squeeze blank lines can be accomplished by the shell script  
8619 shown in following example:

```
8620 sed -n '
8621 # Write non-empty lines.
8622 ./ {
8623 p
8624 d
8625 }
8626 # Write a single empty line, then look for more empty lines.
8627 /^$/ p
8628 # Get next line, discard the held <newline> (empty line),
8629 # and look for more empty lines.
8630 :Empty
8631 /^$/ {
8632 N
8633 s/./.
8634 b Empty
8635 }
8636 # Write the non-empty line before going back to search
8637 # for the first in a set of empty lines.
8638 p
8639 '
```

8640 The System V `-s` option to silence error messages can be accomplished by redirecting the  
8641 standard error. Note that the BSD documentation for *cat* uses the term "blank line" to mean the  
8642 same as the POSIX "empty line": a line consisting only of a `<newline>`.

8643 The BSD `-n` option was omitted because similar functionality can be obtained from the `-n`  
8644 option of the *pr* utility.

#### 8645 FUTURE DIRECTIONS

8646 None.

8647 **SEE ALSO**

8648 *more*

8649 **CHANGE HISTORY**

8650 First released in Issue 2.

8651 **Issue 4**

8652 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 8653 NAME

8654 cd — change the working directory

## 8655 SYNOPSIS

8656 cd [-L] [-P] [*directory*]

8657 cd -

## 8658 DESCRIPTION

8659 The *cd* utility shall change the working directory of the current shell execution environment (see  
8660 Section 2.13 (on page 2273)) by executing the following steps in sequence. (In the following  
8661 steps, the symbol **curpath** represents an intermediate value used to simplify the description of  
8662 the algorithm used by *cd*. There is no requirement that **curpath** be made visible to the  
8663 application.)

- 8664 1. If no *directory* operand is given and the *HOME* environment variable is empty or  
8665 undefined, the default behavior is implementation-defined and no further steps shall be  
8666 taken.
- 8667 2. If no *directory* operand is given and the *HOME* environment variable is set to a non-empty  
8668 value, the *cd* utility shall behave as if the directory named in the *HOME* environment  
8669 variable was specified as the *directory* operand.
- 8670 3. If the *directory* operand begins with a slash character, set **curpath** to the operand and  
8671 proceed to step 7.
- 8672 4. If the first component of the *directory* operand is dot or dot-dot, proceed to step 6.
- 8673 5. Starting with the first path name in the colon-separated path names of *CDPATH* (see the  
8674 ENVIRONMENT VARIABLES section) if the path name is non-null, test if the  
8675 concatenation of that path name, a slash character, and the *directory* operand names a  
8676 directory. If the path name is null, test if the concatenation of dot, a slash character, and the  
8677 operand names a directory. In either case, if the resulting string names an existing  
8678 directory, set **curpath** to that string and proceed to step 7. Otherwise, repeat this step with  
8679 the next path name in *CDPATH* until all path names have been tested.
- 8680 6. Set **curpath** to the string formed by the concatenation of the value of *PWD* a slash  
8681 character, and the operand.
- 8682 7. If the **-P** option is in effect, the *cd* utility shall perform actions equivalent to the *chdir*()  
8683 function, called with **curpath** as the *path* argument. If these actions succeed, the *PWD*  
8684 environment variable shall be set to an absolute path name for the current working  
8685 directory and shall not contain file name components that, in the context of path name  
8686 resolution, refer to a file of type symbolic link. If there is insufficient permission on the new  
8687 directory, or on any parent of that directory, to determine the current working directory,  
8688 the value of the *PWD* environment variable is unspecified. If the actions equivalent to  
8689 *chdir*() fail for any reason, the *cd* utility shall display an appropriate error message and not  
8690 alter the *PWD* environment variable. Whether the actions equivalent to *chdir*() succeed or  
8691 fail, no further steps shall be taken.
- 8692 8. The **curpath** value shall then be converted to canonical form as follows, considering each  
8693 component from beginning to end, in sequence:
  - 8694 a. Dot components and any slashes that separate them from the next component shall  
8695 be deleted.
  - 8696 b. For each dot-dot component, if there is a preceding component and it is neither root  
8697 nor dot-dot, the preceding component, all slashes separating the preceding  
8698 component from dot-dot, dot-dot, and all slashes separating dot-dot from the

8699 following component shall be deleted.

8700 c. An implementation may further simplify **curpath** by removing any trailing slash  
 8701 characters that are not also leading slashes, replacing multiple non-leading  
 8702 consecutive slashes with a single slash, and replacing three or or more leading  
 8703 slashes with a single slash. If, as a result of this canonicalization, the **curpath** variable  
 8704 is null, no further steps shall be taken.

8705 9. The *cd* utility shall then perform actions equivalent to the *chdir()* function called with  
 8706 **curpath** as the *path* argument. If these actions failed for any reason, the *cd* utility shall  
 8707 display an appropriate error message and no further steps shall be taken. The *PWD*  
 8708 environment variable shall be set to **curpath**.

8709 If, during the execution of the above steps, the *PWD* environment variable is changed, the  
 8710 *OLDPWD* environment variable shall also be changed to the value of the old working directory  
 8711 (that is the current working directory immediately prior to the call to *cd*).

### 8712 OPTIONS

8713 The *cd* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 8714 Utility Syntax Guidelines.

8715 The following options shall be supported by the implementation:

8716 **-L** Handle the operand dot-dot logically; symbolic link components shall not be  
 8717 resolved before dot-dot components are processed (see steps 5. and 6. in the  
 8718 DESCRIPTION).

8719 **-P** Handle the operand dot-dot physically; symbolic link components shall be  
 8720 resolved before dot-dot components are processed (see step 4. in the  
 8721 DESCRIPTION).

8722 If both **-L** and **-P** options are specified, the last of these options shall be used and all others  
 8723 ignored. If neither **-L** nor **-P** is specified, the operand shall be handled dot-dot logically; see the  
 8724 DESCRIPTION.

### 8725 OPERANDS

8726 The following operands shall be supported:

8727 *directory* An absolute or relative path name of the directory that shall become the new  
 8728 working directory. The interpretation of a relative path name by *cd* depends on the  
 8729 **-L** option and the *CDPATH* and *PWD* environment variables. If *directory* is an  
 8730 empty string, the results are unspecified.

8731 **-** When a hyphen is used as the operand, this is equivalent to the command:

```
8732 cd "$OLDPWD" && pwd
```

8733 which changes to the previous working directory and then writes its name.

### 8734 STDIN

8735 Not used.

### 8736 INPUT FILES

8737 None.

### 8738 ENVIRONMENT VARIABLES

8739 The following environment variables shall affect the execution of *cd*:

8740 *CDPATH* A colon-separated list of path names that refer to directories. The *cd* utility shall  
 8741 use this list in its attempt to change the directory, as described in the  
 8742 DESCRIPTION. An empty string in place of a directory path name represents the

- 8743 current directory. If *CDPATH* is not set, it shall be treated as if it were an empty  
8744 string.
- 8745 *HOME* The name of the directory, used when no *directory* operand is specified.
- 8746 *LANG* Provide a default value for the internationalization variables that are unset or null.  
8747 If *LANG* is unset or null, the corresponding value from the implementation-  
8748 defined default locale shall be used. If any of the internationalization variables  
8749 contains an invalid setting, the utility shall behave as if none of the variables had  
8750 been defined.
- 8751 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
8752 internationalization variables.
- 8753 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
8754 characters (for example, single-byte as opposed to multi-byte characters in  
8755 arguments).
- 8756 *LC\_MESSAGES*  
8757 Determine the locale that should be used to affect the format and contents of  
8758 diagnostic messages written to standard error.
- 8759 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 8760 *OLDPWD* A path name of the previous working directory, used by *cd -*.
- 8761 *PWD* This variable shall be set as specified in the DESCRIPTION. If an application sets  
8762 or unsets the value of *PWD*, the behavior of *cd* is unspecified.
- 8763 **ASYNCHRONOUS EVENTS**
- 8764 Default.
- 8765 **STDOUT**
- 8766 If a non-empty directory name from *CDPATH* is used, or if *cd -* is used, an absolute path name of  
8767 the new working directory shall be written to the standard output as follows:
- 8768 "%s\n", <new directory>
- 8769 Otherwise, there shall be no output.
- 8770 **STDERR**
- 8771 Used only for diagnostic messages.
- 8772 **OUTPUT FILES**
- 8773 None.
- 8774 **EXTENDED DESCRIPTION**
- 8775 None.
- 8776 **EXIT STATUS**
- 8777 The following exit values shall be returned:
- 8778 0 The directory was successfully changed.
- 8779 >0 An error occurred.
- 8780 **CONSEQUENCES OF ERRORS**
- 8781 The working directory shall remain unchanged.

**8782 APPLICATION USAGE**

8783 Since *cd* affects the current shell execution environment, it is always provided as a shell regular  
8784 built-in. If it is called in a subshell or separate utility execution environment, such as one of the  
8785 following:

```
8786 (cd /tmp)
8787 nohup cd
8788 find . -exec cd {} \;
```

8789 it does not affect the working directory of the caller's environment.

8790 The user must have execute (search) permission in *directory* in order to change to it.

**8791 EXAMPLES**

8792 None.

**8793 RATIONALE**

8794 The use of the *CDPATH* was introduced in the System V shell. Its use is analogous to the use of  
8795 the *PATH* variable in the shell. The BSD C shell used a shell parameter *cdpath* for this purpose.

8796 A common extension when *HOME* is undefined is to get the login directory from the user  
8797 database for the invoking user. This does not occur on System V implementations.

8798 Some historical shells, such as the KornShell, took special actions when the directory name  
8799 contained a dot-dot component, selecting the logical parent of the directory, rather than the  
8800 actual parent directory; that is, it moved up one level toward the '/' in the path name,  
8801 remembering what the user typed, rather than performing the equivalent of:

```
8802 chdir("../");
```

8803 In such a shell, the following commands would not necessarily produce equivalent output for all  
8804 directories:

```
8805 cd .. && ls ls ..
```

8806 This behavior is not permitted by default because it is not consistent with the definition of dot-  
8807 dot in most historical practice; that is, while this behavior has been optionally available in the  
8808 KornShell, other shells have historically not supported this functionality. The logical path name  
8809 is stored in the *PWD* environment variable when the *cd* utility completes and this value is used  
8810 to construct the next directory name if *cd* is invoked with the *-L* option.

**8811 FUTURE DIRECTIONS**

8812 None.

**8813 SEE ALSO**

8814 *pwd*, the System Interfaces volume of IEEE Std. 1003.1-200x, *chdir()*

**8815 CHANGE HISTORY**

8816 First released in Issue 2.

**8817 Issue 4**

8818 Aligned with the ISO/IEC 9945-2:1993 standard.

8819 Extensions added for *cd-*, *PWD*, and *OLDPWD*.

**8820 Issue 6**

8821 The following new requirements on POSIX implementations derive from alignment with the  
8822 Single UNIX Specification:

- 8823 • The *cd-*, *PWD*, and *OLDPWD* are added.



8824  
8825

The **-L** and **-P** options are added to align with the IEEE P1003.2b draft standard. This also includes the introduction of a new description to include the effect of these options.

## 8826 NAME

8827 cflow — generate a C-language flowgraph (DEVELOPMENT)

## 8828 SYNOPSIS

```
8829 xSI cflow [-r][-d num][-D name[=def]] ... [-i incl][-I dir] ...
8830 [-U dir] ... file ...
```

8831

## 8832 DESCRIPTION

8833 The *cflow* utility shall analyse a collection of object files or assembler, C-language, *lex* or *yacc*  
 8834 source files, and attempt to build a graph, written to standard output, charting the external  
 8835 references.

## 8836 OPTIONS

8837 The *cflow* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 8838 12.2, Utility Syntax Guidelines, except that the order of the **-D**, **-I**, and **-U** options (which are  
 8839 identical to their interpretation by *c99*) is significant.

8840 The following options shall be supported:

8841 **-d num** Indicate the depth at which the flowgraph is cut off. The application shall ensure  
 8842 that the argument *num* is a decimal integer. By default this is a very large number  
 8843 (typically greater than 32 000). Attempts to set the cut-off depth to a non-positive  
 8844 integer are ignored.

8845 **-i incl** Increase the number of included symbols. The *incl* option-argument is one of the  
 8846 following characters:

8847 *x* Include external and static data symbols. The default shall be to include only  
 8848 functions in the flowgraph.

8849 *\_* (Underscore) Include names that begin with an underscore. The default shall  
 8850 be to exclude these functions (and data if **-i x** is used).

8851 **-r** Reverse the caller:callee relationship, producing an inverted listing showing the  
 8852 callers of each function. The listing is also sorted in lexicographical order by callee.

## 8853 OPERANDS

8854 The following operand is supported:

8855 *file* The path name of a file for which a graph is to be generated. Files suffixed in **.l**, **.y**,  
 8856 **.c**, and **.i** shall be processed by *lex* and *yacc* and preprocessed by the *c99*  
 8857 preprocessor phase (bypassed for **.i** files) as appropriate, and then run through the  
 8858 first pass of *lint*. Files suffixed with **.s** shall be assembled and information shall be  
 8859 extracted (as in **.o** files) from the symbol table.

## 8860 STDIN

8861 Not used.

## 8862 INPUT FILES

8863 The input files shall be object files or assembler, C-language, *lex* or *yacc* source files.

## 8864 ENVIRONMENT VARIABLES

8865 The following environment variables shall affect the execution of *cflow*:

8866 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 8867 If *LANG* is unset or null, the corresponding value from the implementation-  
 8868 defined default locale shall be used. If any of the internationalization variables  
 8869 contains an invalid setting, the utility shall behave as if none of the variables had  
 8870 been defined.

- 8871 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
8872 internationalization variables.
- 8873 *LC\_COLLATE*  
8874 Determine the locale for the ordering of the output when the *-r* option is used.
- 8875 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
8876 characters (for example, single-byte as opposed to multi-byte characters in  
8877 arguments and input files).
- 8878 *LC\_MESSAGES*  
8879 Determine the locale that should be used to affect the format and contents of  
8880 diagnostic messages written to standard error.
- 8881 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 8882 **ASYNCHRONOUS EVENTS**  
8883 Default.
- 8884 **STDOUT**  
8885 The flowgraph written to standard output shall be formatted as follows:  
8886 "%d %s:%s\n", <reference number>, <global>, <definition>
- 8887 Each line of output begins with a reference (that is, line) number, followed by indentation of at  
8888 least one column position per level. This is followed by the name of the global, a colon, and its  
8889 definition. Normally globals are only functions not defined as an external or beginning with an  
8890 underscore; see the **OPTIONS** section for the *-i* inclusion option. For information extracted from  
8891 C-language source, the definition consists of an abstract type declaration (for example, **char\***)  
8892 and, delimited by angle brackets, the name of the source file and the line number where the  
8893 definition was found. Definitions extracted from object files indicate the file name and location  
8894 counter under which the symbol appeared (for example, *text*).
- 8895 Once a definition of a name has been written, subsequent references to that name contain only  
8896 the reference number of the line where the definition can be found. For undefined references,  
8897 only "< >" shall be written.
- 8898 **STDERR**  
8899 Used only for diagnostic messages.
- 8900 **OUTPUT FILES**  
8901 None.
- 8902 **EXTENDED DESCRIPTION**  
8903 None.
- 8904 **EXIT STATUS**  
8905 The following exit values shall be returned:  
8906 0 Successful completion.  
8907 >0 An error occurred.
- 8908 **CONSEQUENCES OF ERRORS**  
8909 Default.

**8910 APPLICATION USAGE**

8911 Files produced by *lex* and *yacc* cause the reordering of line number declarations, and this can  
8912 confuse *cflow*. To obtain proper results, the input of *yacc* or *lex* must be directed to *cflow*.

**8913 EXAMPLES**

8914 Given the following in **file.c**:

```
8915 int i;
8916 main()
8917 {
8918 f();
8919 g();
8920 f();
8921 }
8922 f()
8923 {
8924 i = h();
8925 }
```

8926 The command:

```
8927 cflow -i x file.c
```

8928 produces the output:

```
8929 1 main: int(), <file.c 2>
8930 2 f: int(), <file.c 8>
8931 3 h: <>
8932 4 i: int, <file.c 1>
8933 5 g: <>
```

**8934 RATIONALE**

8935 None.

**8936 FUTURE DIRECTIONS**

8937 None.

**8938 SEE ALSO**

8939 *c99*, *lex*, *yacc*

**8940 CHANGE HISTORY**

8941 First released in Issue 2.

**8942 Issue 4**

8943 Format reorganized.

8944 Internationalized environment variable support mandated.

**8945 Issue 6**

8946 The normative text is reworded to avoid use of the term “must” for application requirements.

8947 **NAME**

8948 chgrp — change the file group ownership

8949 **SYNOPSIS**8950 chgrp -hR *group file ...*8951 chgrp -R [-H | -L | -P ] *group file ...*8952 **DESCRIPTION**8953 The *chgrp* utility shall set the group ID of the file named by each *file* operand to the group ID  
8954 specified by the *group* operand.8955 For each *file* operand, it shall perform actions equivalent to the *chown()* function defined in the  
8956 System Interfaces volume of IEEE Std. 1003.1-200x, called with the following arguments:

- 8957 • The *file* operand shall be used as the *path* argument.
- 8958 • The user ID of the file shall be used as the *owner* argument.
- 8959 • The specified group ID shall be used as the *group* argument.

8960 Unless *chgrp* is invoked by a process with appropriate privileges, the set-user-ID and set-group-  
8961 ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and set-  
8962 group-ID bits of other file types may be cleared.8963 **OPTIONS**8964 The *chgrp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
8965 12.2, Utility Syntax Guidelines.

8966 The following options shall be supported by the implementation:

8967 **-h** If the system supports group IDs for symbolic links, for each *file* operand that  
8968 names a file of type symbolic link, *chgrp* shall attempt to set the group ID of the  
8969 symbolic link instead of the file referenced by the symbolic link. If the system does  
8970 not support group IDs for symbolic links, for each *file* operand that names a file of  
8971 type symbolic link, *chgrp* shall do nothing more with the current file and shall go  
8972 on to any remaining files.

8973 **-H** If the **-R** option is specified and a symbolic link referencing a file of type directory  
8974 is specified on the command line, *chgrp* shall change the group of the directory  
8975 referenced by the symbolic link and all files in the file hierarchy below it.

8976 **-L** If the **-R** option is specified and a symbolic link referencing a file of type directory  
8977 is specified on the command line or encountered during the traversal of a file  
8978 hierarchy, *chgrp* shall change the group of the directory referenced by the symbolic  
8979 link and all files in the file hierarchy below it.

8980 **-P** If the **-R** option is specified and a symbolic link is specified on the command line  
8981 or encountered during the traversal of a file hierarchy, *chgrp* shall change the  
8982 group ID of the symbolic link if the system supports this operation. The *chgrp*  
8983 utility shall not follow the symbolic link to any other part of the file hierarchy.

8984 **-R** Recursively change file group IDs. For each *file* operand that names a directory,  
8985 *chgrp* shall change the group of the directory and all files in the file hierarchy below  
8986 it. Unless a **-H**, **-L**, or **-P** option is specified, it is unspecified which of these  
8987 options will be used as the default.

8988 Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be  
8989 considered an error. The last option specified shall determine the behavior of the utility.

8990 **OPERANDS**

8991 The following operands shall be supported:

8992 *group* A group name from the group database or a numeric group ID. Either specifies a  
8993 group ID to be given to each file named by one of the *file* operands. If a numeric  
8994 *group* operand exists in the group database as a group name, the group ID number  
8995 associated with that group name is used as the group ID.

8996 *file* A path name of a file whose group ID is to be modified.

8997 **STDIN**

8998 Not used.

8999 **INPUT FILES**

9000 None.

9001 **ENVIRONMENT VARIABLES**9002 The following environment variables shall affect the execution of *chgrp*:

9003 *LANG* Provide a default value for the internationalization variables that are unset or null.  
9004 If *LANG* is unset or null, the corresponding value from the implementation-  
9005 defined default locale shall be used. If any of the internationalization variables  
9006 contains an invalid setting, the utility shall behave as if none of the variables had  
9007 been defined.

9008 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
9009 internationalization variables.

9010 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
9011 characters (for example, single-byte as opposed to multi-byte characters in  
9012 arguments).

9013 *LC\_MESSAGES*

9014 Determine the locale that should be used to affect the format and contents of  
9015 diagnostic messages written to standard error.

9016 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

9017 **ASYNCHRONOUS EVENTS**

9018 Default.

9019 **STDOUT**

9020 Not used.

9021 **STDERR**

9022 Used only for diagnostic messages.

9023 **OUTPUT FILES**

9024 None.

9025 **EXTENDED DESCRIPTION**

9026 None.

9027 **EXIT STATUS**

9028 The following exit values shall be returned:

9029 0 The utility executed successfully and all requested changes were made.

9030 >0 An error occurred.

9031 **CONSEQUENCES OF ERRORS**

9032 Default.

9033 **APPLICATION USAGE**9034 Only the owner of a file or the user with appropriate privileges may change the owner or group  
9035 of a file.9036 Some systems restrict the use of *chgrp* to a user with appropriate privileges when the *group*  
9037 specified is not the effective group ID or one of the supplementary group IDs of the calling  
9038 process.9039 **EXAMPLES**

9040 None.

9041 **RATIONALE**9042 The System V and BSD versions use different exit status codes. Some implementations used the  
9043 exit status as a count of the number of errors that occurred; this practice is unworkable since it  
9044 can overflow the range of valid exit status values. The standard developers chose to mask these  
9045 by specifying only 0 and >0 as exit values.9046 The functionality of *chgrp* is described substantially through references to *chown()*. In this way,  
9047 there is no duplication of effort required for describing the interactions of permissions, multiple  
9048 groups, and so on.9049 **FUTURE DIRECTIONS**

9050 None.

9051 **SEE ALSO**9052 *chmod*, *chown*, the System Interfaces volume of IEEE Std. 1003.1-200x, *chown()*9053 **CHANGE HISTORY**

9054 First released in Issue 2.

9055 **Issue 4**

9056 Aligned with the ISO/IEC 9945-2:1993 standard.

9057 **Issue 6**9058 New options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
9059 options affect the processing of symbolic links.9060 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
9061 section to "Default."

9062 **NAME**

9063 chmod — change the file modes

9064 **SYNOPSIS**9065 chmod [-R] *mode file ...*9066 **DESCRIPTION**9067 The *chmod* utility shall change any or all of the file mode bits of the file named by each *file*  
9068 operand in the way specified by the *mode* operand.9069 It is implementation-defined whether and how the *chmod* utility affects any alternate or  
9070 additional file access control mechanism (see the Base Definitions volume of  
9071 IEEE Std. 1003.1-200x, Section 4.1, File Access Permissions) being used for the specified file.9072 Only a process whose effective user ID matches the user ID of the file, or a process with the  
9073 appropriate privileges, shall be permitted to change the file mode bits of a file.9074 **OPTIONS**9075 The *chmod* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
9076 12.2, Utility Syntax Guidelines.

9077 The following option shall be supported:

9078 **-R** Recursively change file mode bits. For each *file* operand that names a directory,  
9079 *chmod* shall change the file mode bits of the directory and all files in the file  
9080 hierarchy below it.9081 **OPERANDS**

9082 The following operands shall be supported:

9083 *mode* Represents the change to be made to the file mode bits of each file named by one of  
9084 the *file* operands; see the EXTENDED DESCRIPTION section.9085 *file* A path name of a file whose file mode bits shall be modified.9086 **STDIN**

9087 Not used.

9088 **INPUT FILES**

9089 None.

9090 **ENVIRONMENT VARIABLES**9091 The following environment variables shall affect the execution of *chmod*:9092 *LANG* Provide a default value for the internationalization variables that are unset or null.  
9093 If *LANG* is unset or null, the corresponding value from the implementation-  
9094 defined default locale shall be used. If any of the internationalization variables  
9095 contains an invalid setting, the utility shall behave as if none of the variables had  
9096 been defined.9097 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
9098 internationalization variables.9099 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
9100 characters (for example, single-byte as opposed to multi-byte characters in  
9101 arguments).9102 *LC\_MESSAGES*9103 Determine the locale that should be used to affect the format and contents of  
9104 diagnostic messages written to standard error.



- 9105 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 9106 **ASYNCHRONOUS EVENTS**
- 9107 Default.
- 9108 **STDOUT**
- 9109 Not used.
- 9110 **STDERR**
- 9111 Used only for diagnostic messages.
- 9112 **OUTPUT FILES**
- 9113 None.
- 9114 **EXTENDED DESCRIPTION**
- 9115 The *mode* operand shall be either a *symbolic\_mode* expression or a non-negative octal integer. The
- 9116 *symbolic\_mode* form is described by the grammar later in this section.
- 9117 Each **clause** shall specify an operation to be performed on the current file mode bits of each *file*.
- 9118 The operations shall be performed on each *file* in the order in which the **clauses** are specified.
- 9119 The **who** symbols **u**, **g**, and **o** shall specify the *user*, *group*, and *other* parts of the file mode bits,
- 9120 respectively. A **who** consisting of the symbol **a** shall be equivalent to **ugo**.
- 9121 The **perm** symbols **r**, **w**, and **x** represent the *read*, *write*, and *execute/search* portions of file mode
- 9122 bits, respectively. The **perm** symbol **s** shall represent the *set-user-ID-on-execution* (when **who**
- 9123 contains or implies **u**) and *set-group-ID-on-execution* (when **who** contains or implies **g**) bits.
- 9124 The **perm** symbol **X** shall represent the execute/search portion of the file mode bits if the file is a
- 9125 directory or if the current (unmodified) file mode bits have at least one of the execute bits
- 9126 (S\_IXUSR, S\_IXGRP, or S\_IXOTH) set. It shall be ignored if the file is not a directory and none of
- 9127 the execute bits are set in the current file mode bits.
- 9128 The **permcop** symbols **u**, **g**, and **o** shall represent the current permissions associated with the
- 9129 user, group, and other parts of the file mode bits, respectively. For the remainder of this section,
- 9130 **perm** refers to the non-terminals **perm** and **permcop** in the grammar.
- 9131 If multiple **actionlists** are grouped with a single **wholist** in the grammar, each **actionlist** shall be
- 9132 applied in the order specified with that **wholist**. The *op* symbols shall represent the operation
- 9133 performed, as follows:
- 9134 + If **perm** is not specified, the '+' operation shall not change the file mode bits.
- 9135 If **who** is not specified, the file mode bits represented by **perm** for the owner, group, and
- 9136 other permissions, except for those with corresponding bits in the file mode creation mask
- 9137 of the invoking process, shall be set.
- 9138 Otherwise, the file mode bits represented by the specified **who** and **perm** values shall be set.
- 9139 - If **perm** is not specified, the '-' operation shall not change the file mode bits.
- 9140 If **who** is not specified, the file mode bits represented by **perm** for the owner, group, and
- 9141 other permissions, except for those with corresponding bits in the file mode creation mask
- 9142 of the invoking process, shall be cleared.
- 9143 Otherwise, the file mode bits represented by the specified **who** and **perm** values shall be
- 9144 cleared.
- 9145 = Clear the file mode bits specified by the **who** value, or, if no **who** value is specified, all of the
- 9146 file mode bits specified in this volume of IEEE Std. 1003.1-200x.

9147 If **perm** is not specified, the '=' operation shall make no further modifications to the file  
 9148 mode bits.

9149 If **who** is not specified, the file mode bits represented by **perm** for the owner, group, and  
 9150 other permissions, except for those with corresponding bits in the file mode creation mask  
 9151 of the invoking process, shall be set.

9152 Otherwise, the file mode bits represented by the specified **who** and **perm** values shall be set.

9153 When using the symbolic mode form on a regular file, it is implementation-defined whether or  
 9154 not:

- 9155 • Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all  
 9156 execute bits are currently clear and none are being set are ignored.
- 9157 • Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-  
 9158 on-execution bits.
- 9159 • Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all  
 9160 execute bits are currently clear are ignored. However, if the command *ls -l file* writes an *s* in  
 9161 the position indicating that the set-user-ID-on-execution or set-group-ID-on-execution is set,  
 9162 the commands *chmod u-s file* or *chmod g-s file*, respectively, shall not be ignored.

9163 When using the symbolic mode form on other file types, it is implementation-defined whether  
 9164 or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are  
 9165 honored.

9166 If the **who** symbol **o** is used in conjunction with the **perm** symbol **s** with no other **who** symbols  
 9167 being specified, the set-user-ID-on-execution and set-group-ID-on-execution bits shall not be  
 9168 modified. It shall not be an error to specify the **who** symbol **o** in conjunction with the **perm**  
 9169 symbol **s**.

9170 For an octal integer *mode* operand, the file mode bits shall be set absolutely.

9171 For each bit set in the octal number, the corresponding file permission bit shown in the following  
 9172 table shall be set; all other file permission bits shall be cleared. For regular files, for each bit set in  
 9173 the octal number corresponding to the set-user-ID-on-execution or the set-group-ID-on-  
 9174 execution, bits shown in the following table shall be set; if these bits are not set in the octal  
 9175 number, they are cleared. For other file types, it is implementation-defined whether or not  
 9176 requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are  
 9177 honored.

| 9178 | Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit |
|------|-------|----------|-------|----------|-------|----------|-------|----------|
| 9179 | 4000  | S_ISUID  | 0400  | S_IRUSR  | 0040  | S_IRGRP  | 0004  | S_IROTH  |
| 9180 | 2000  | S_ISGID  | 0200  | S_IWUSR  | 0020  | S_IWGRP  | 0002  | S_IWOTH  |
| 9181 |       |          | 0100  | S_IXUSR  | 0010  | S_IXGRP  | 0001  | S_IXOTH  |

9182 When bits are set in the octal number other than those listed in the table above, the behavior is  
 9183 unspecified.

9184 **Grammar for chmod**

9185 The grammar and lexical conventions in this section describe the syntax for the *symbolic\_mode*  
 9186 operand. The general conventions for this style of grammar are described in Section 1.10 (on  
 9187 page 2223). A valid *symbolic\_mode* can be represented as the non-terminal symbol *symbolic\_mode*  
 9188 in the grammar. This formal syntax shall take precedence over the preceding text syntax  
 9189 description.

9190 The lexical processing is based entirely on single characters. Implementations need not allow  
 9191 blank characters within the single argument being processed.

```

9192 %start symbolic_mode
9193 %%

9194 symbolic_mode : section
9195 | symbolic_mode ',' clause
9196 ;

9197 clause : actionlist
9198 | wholist actionlist
9199 ;

9200 wholist : who
9201 | wholist who
9202 ;

9203 who : 'u' | 'g' | 'o' | 'a'
9204 ;

9205 actionlist : action
9206 | actionlist action
9207 ;

9208 action : op
9209 | op permlist
9210 | op permcopy
9211 ;

9212 permcopy : 'u' | 'g' | 'o'
9213 ;

9214 op : '+' | '-' | '='
9215 ;

9216 permlist : perm
9217 | perm permlist
9218 ;

9219 perm : 'r' | 'w' | 'x' | 'X' | 's'
9220 ;

```

9221 **EXIT STATUS**

9222 The following exit values shall be returned:

9223 0 The utility executed successfully and all requested changes were made.

9224 >0 An error occurred.

9225 CONSEQUENCES OF ERRORS

9226 Default.

9227 APPLICATION USAGE

9228 Some implementations of the *chmod* utility change the mode of a directory before the files in the  
 9229 directory when performing a recursive (**-R** option) change; others change the directory mode  
 9230 after the files in the directory. If an application tries to remove read or search permission for a  
 9231 file hierarchy, the removal attempt fails if the directory is changed first; on the other hand, trying  
 9232 to re-enable permissions to a restricted hierarchy fails if directories are changed last. Users  
 9233 should not try to make a hierarchy inaccessible to themselves.

9234 Some implementations of *chmod* never used the process' *umask* when changing modes; systems  
 9235 conformant with this volume of IEEE Std. 1003.1-200x do so when **who** is not specified. Note the  
 9236 difference between:

9237 `chmod a-w file`

9238 which removes all write permissions, and:

9239 `chmod -- -w file`

9240 which removes write permissions that would be allowed if **file** was created with the same  
 9241 *umask*.

9242 Portable applications should never assume that they know how the set-user-ID and set-group-  
 9243 ID bits on directories are interpreted.

9244 EXAMPLES

9245

9246

9247

9248

9249

9250

9251

9252

9253

9254

| Mode         | Results                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------|
| <i>a+=</i>   | Equivalent to <i>a+,a=</i> ; clears all file mode bits.                                            |
| <i>go+-w</i> | Equivalent to <i>go+,go-w</i> ; clears group and other write bits.                                 |
| <i>g=o-w</i> | Equivalent to <i>g=o,g-w</i> ; sets group bit to match other bits and then clears group write bit. |
| <i>g-r+w</i> | Equivalent to <i>g-r,g+w</i> ; clears group read bit and sets group write bit.                     |
| <i>=g</i>    | Sets owner bits to match group bits and sets other bits to match group bits.                       |

9255 RATIONALE

9256 The functionality of *chmod* is described substantially through references to concepts defined in  
 9257 the System Interfaces volume of IEEE Std. 1003.1-200x. In this way, there is less duplication of  
 9258 effort required for describing the interactions of permissions. However, the behavior of this  
 9259 utility is not described in terms of the *chmod()* function from the System Interfaces volume of  
 9260 IEEE Std. 1003.1-200x because that specification requires certain side effects upon alternate file  
 9261 access control mechanisms that might not be appropriate, depending on the implementation.

9262 Implementations that support mandatory file and record locking as specified by the 1984  
 9263 /usr/group standard historically used the combination of set-group-ID bit set and group  
 9264 execute bit clear to indicate mandatory locking. This condition is usually set or cleared with the  
 9265 symbolic mode **perm** symbol **l** instead of the **perm** symbols **s** and **x** so that the mandatory  
 9266 locking mode is not changed without explicit indication that that was what the user intended.  
 9267 Therefore, the details on how the implementation treats these conditions must be defined in the  
 9268 documentation. This volume of IEEE Std. 1003.1-200x does not require mandatory locking (nor  
 9269 does the System Interfaces volume of IEEE Std. 1003.1-200x), but does allow it as an extension.  
 9270 However, this volume of IEEE Std. 1003.1-200x does require that the *ls* and *chmod* utilities work

- 9271 consistently in this area. If *ls -l file* indicates that the set-group-ID bit is set, *chmod g-s file* must  
9272 clear it (assuming appropriate privileges exist to change modes).
- 9273 The System V and BSD versions use different exit status codes. Some implementations used the  
9274 exit status as a count of the number of errors that occurred; this practice is unworkable since it  
9275 can overflow the range of valid exit status values. This problem is avoided here by specifying  
9276 only 0 and >0 as exit values.
- 9277 The System Interfaces volume of IEEE Std. 1003.1-200x indicates that implementation-defined  
9278 restrictions may cause the S\_ISUID and S\_ISGID bits to be ignored. This volume of  
9279 IEEE Std. 1003.1-200x allows the *chmod* utility to choose to modify these bits before calling  
9280 *chmod()* (or some function providing equivalent capabilities) for non-regular files. Among other  
9281 things, this allows implementations that use the set-user-ID and set-group-ID bits on directories  
9282 to enable extended features to handle these extensions in an intelligent manner.
- 9283 The **X perm** symbol was adopted from BSD-based systems because it provides commonly  
9284 desired functionality when doing recursive (**-R** option) modifications. Similar functionality is  
9285 not provided by the *find* utility. Historical BSD versions of *chmod*, however, only supported **X**  
9286 with *op+*; it has been extended in this volume of IEEE Std. 1003.1-200x because it is also useful  
9287 with *op=*. (It has also been added for *op-* even though it duplicates **x**, in this case, because it is  
9288 intuitive and easier to explain.)
- 9289 The grammar was extended with the *permcop* non-terminal to allow historical-practice forms of  
9290 symbolic modes like **o=u -g** (that is, set the “other” permissions to the permissions of “owner”  
9291 minus the permissions of “group”).
- 9292 **FUTURE DIRECTIONS**
- 9293 None.
- 9294 **SEE ALSO**
- 9295 *ls*, *umask*, the System Interfaces volume of IEEE Std. 1003.1-200x, *chmod()*
- 9296 **CHANGE HISTORY**
- 9297 First released in Issue 2.
- 9298 **Issue 4**
- 9299 Aligned with the ISO/IEC 9945-2:1993 standard.
- 9300 **Issue 6**
- 9301 The following new requirements on POSIX implementations derive from alignment with the  
9302 Single UNIX Specification:
- 9303 • Octal modes have been kept and made mandatory despite being marked obsolescent in the  
9304 previous version of this volume of IEEE Std. 1003.1-200x.
- 9305 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
9306 section to “Default.”.

9307 **NAME**

9308 chown — change the file ownership

9309 **SYNOPSIS**

9310 chown -hR owner[:group] file ...

9311 chown -R [-H | -L | -P ] owner[:group] file ...

9312 **DESCRIPTION**9313 The *chown* utility shall set the user ID of the file named by each *file* operand to the user ID  
9314 specified by the *owner* operand.9315 For each *file* operand, it shall perform actions equivalent to the *chown()* function defined in the  
9316 System Interfaces volume of IEEE Std. 1003.1-200x, called with the following arguments:

- 9317 1. The
- file*
- operand shall be used as the
- path*
- argument.
- 
- 9318 2. The user ID indicated by the
- owner*
- portion of the first operand shall be used as the
- owner*
- 
- 9319 argument.
- 
- 9320 3. If the
- group*
- portion of the first operand is given, the group ID indicated by it shall be used
- 
- 9321 as the
- group*
- argument; otherwise, the group ID of the file shall be used as the
- group*
- 
- 9322 argument.

9323 Unless *chown* is invoked by a process with appropriate privileges, the set-user-ID and set-  
9324 group-ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and  
9325 set-group-ID bits of other file types may be cleared.9326 **OPTIONS**9327 The *chown* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
9328 12.2, Utility Syntax Guidelines.

9329 The following options shall be supported by the implementation:

9330 **-h** If the system supports user IDs for symbolic links, for each *file* operand that names  
9331 a file of type symbolic link, *chown* shall attempt to set the user ID of the symbolic  
9332 link. If the system supports group IDs for symbolic links, and a group ID was  
9333 specified, for each *file* operand that names a file of type symbolic link, *chown* shall  
9334 attempt to set the group ID of the symbolic link. If the system does not support  
9335 user or group IDs for symbolic links, for each *file* operand that names a file of type  
9336 symbolic link, *chown* shall do nothing more with the current file and shall go on to  
9337 any remaining files.9338 **-H** If the **-R** option is specified and a symbolic link referencing a file of type directory  
9339 is specified on the command line, *chown* shall change the user ID (and group ID, if  
9340 specified) of the directory referenced by the symbolic link and all files in the file  
9341 hierarchy below it.9342 **-L** If the **-R** option is specified and a symbolic link referencing a file of type directory  
9343 is specified on the command line or encountered during the traversal of a file  
9344 hierarchy, *chown* shall change the user ID (and group ID, if specified) of the  
9345 directory referenced by the symbolic link and all files in the file hierarchy below it.9346 **-P** If the **-R** option is specified and a symbolic link is specified on the command line  
9347 or encountered during the traversal of a file hierarchy, *chown* shall change the  
9348 owner ID (and group ID, if specified) of the symbolic link if the system supports  
9349 this operation. The *chown* utility shall not follow the symbolic link to any other  
9350 part of the file hierarchy.

9351           **-R**           Recursively change file user and group IDs. For each *file* operand that names a  
 9352                           directory, *chown* shall change the user ID (and group ID, if specified) of the  
 9353                           directory and all files in the file hierarchy below it. Unless a **-H**, **-L**, or **-P** option is  
 9354                           specified, it is unspecified which of these options will be used as the default.

9355           Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be  
 9356           considered an error. The last option specified shall determine the behavior of the utility.

#### 9357 **OPERANDS**

9358           The following operands shall be supported:

9359           *owner[:group]* A user ID and optional group ID to be assigned to *file*. The application shall  
 9360                           ensure that the *owner* portion of this operand is a user name from the user database  
 9361                           or a numeric user ID. Either specifies a user ID to be given to each file named by  
 9362                           one of the *file* operands. If a numeric *owner* operand exists in the user database as a  
 9363                           user name, the user ID number associated with that user name is used as the user  
 9364                           ID. Similarly, if the *group* portion of this operand is present, it shall be a group  
 9365                           name from the group database or a numeric group ID. Either specifies a group ID  
 9366                           to be given to each file. If a numeric group operand exists in the group database as a  
 9367                           group name, the group ID number associated with that group name shall be used  
 9368                           as the group ID.

9369           *file*           A path name of a file whose user ID is to be modified.

#### 9370 **STDIN**

9371           Not used.

#### 9372 **INPUT FILES**

9373           None.

#### 9374 **ENVIRONMENT VARIABLES**

9375           The following environment variables shall affect the execution of *chown*:

9376           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 9377                           If **LANG** is unset or null, the corresponding value from the implementation-  
 9378                           defined default locale shall be used. If any of the internationalization variables  
 9379                           contains an invalid setting, the utility shall behave as if none of the variables had  
 9380                           been defined.

9381           **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
 9382                           internationalization variables.

9383           **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 9384                           characters (for example, single-byte as opposed to multi-byte characters in  
 9385                           arguments).

9386           **LC\_MESSAGES**

9387                           Determine the locale that should be used to affect the format and contents of  
 9388                           diagnostic messages written to standard error.

9389 **XSI**           **NLSPATH**       Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

#### 9390 **ASYNCHRONOUS EVENTS**

9391           Default.

#### 9392 **STDOUT**

9393           Not used.

9394 **STDERR**

9395           Used only for diagnostic messages.

9396 **OUTPUT FILES**

9397           None.

9398 **EXTENDED DESCRIPTION**

9399           None.

9400 **EXIT STATUS**

9401           The following exit values shall be returned:

9402           0   The utility executed successfully and all requested changes were made.

9403           >0  An error occurred.

9404 **CONSEQUENCES OF ERRORS**

9405           Default.

9406 **APPLICATION USAGE**

9407           Only the owner of a file or the user with appropriate privileges may change the owner or group of a file.

9409           Some systems restrict the use of *chown* to a user with appropriate privileges.

9410 **EXAMPLES**

9411           None.

9412 **RATIONALE**

9413           The System V and BSD versions use different exit status codes. Some implementations used the exit status as a count of the number of errors that occurred; this practice is unworkable since it can overflow the range of valid exit status values. These are masked by specifying only 0 and >0 as exit values.

9417           The functionality of *chown* is described substantially through references to functions in the System Interfaces volume of IEEE Std. 1003.1-200x. In this way, there is no duplication of effort required for describing the interactions of permissions, multiple groups, and so on.

9420           The 4.3 BSD method of specifying both owner and group was included in this volume of IEEE Std. 1003.1-200x because:

9422           • There are cases where the desired end condition could not be achieved using the *chgrp* and *chown* (that only changed the user ID) utilities. (If the current owner is not a member of the desired group and the desired owner is not a member of the current group, the *chown()* function could fail unless both owner and group are changed at the same time.)

9426           • Even if they could be changed independently, in cases where both are being changed, there is a 100% performance penalty caused by being forced to invoke both utilities.

9428           The BSD syntax *user[.group]* was changed to *user[:group]* in this volume of IEEE Std. 1003.1-200x because the period is a valid character in login names (as specified by the Base Definitions volume of IEEE Std. 1003.1-200x, login names consist of characters in the portable file name character set). The colon character was chosen as the replacement for the period character because it would never be allowed as a character in a user name or group name on historical implementations.

9434           The **-R** option is considered by some observers as an undesirable departure from the historical UNIX system tools approach; since a tool, *find*, already exists to recurse over directories, there seemed to be no good reason to require other tools to have to duplicate that functionality. However, the **-R** option was deemed an important user convenience, is far more efficient than



9438 forking a separate process for each element of the directory hierarchy, and is in widespread  
9439 historical use.

9440 **FUTURE DIRECTIONS**

9441 None.

9442 **SEE ALSO**

9443 *chmod*, *chgrp*, the System Interfaces volume of IEEE Std. 1003.1-200x, *chown()*

9444 **CHANGE HISTORY**

9445 First released in Issue 2.

9446 **Issue 4**

9447 Aligned with the ISO/IEC 9945-2:1993 standard.

9448 **Issue 6**

9449 New options **-h**, **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
9450 options affect the processing of symbolic links.

9451 The normative text is reworded to avoid use of the term “must” for application requirements.

9452 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
9453 section is changed to “Default.”.

9454 **NAME**

9455 cksum — write file checksums and sizes

9456 **SYNOPSIS**9457 cksum [*file* ...]9458 **DESCRIPTION**

9459 The *cksum* utility shall calculate and write to standard output a cyclic redundancy check (CRC)  
 9460 for each input file, and also write to standard output the number of octets in each file. The CRC  
 9461 used is based on the polynomial used for CRC error checking in the ISO/IEC 8802-3:1996  
 9462 standard (Ethernet).

9463 The encoding for the CRC checksum is defined by the generating polynomial:

$$9464 G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

9465 Mathematically, the CRC value corresponding to a given file shall be defined by the following  
 9466 procedure:

- 9467 1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial *M(x)*  
 9468 of degree *n*−1. These *n* bits are the bits from the file, with the most significant bit being the  
 9469 most significant bit of the first octet of the file and the last bit being the least significant bit  
 9470 of the last octet, padded with zero bits (if necessary) to achieve an integral number of  
 9471 octets, followed by one or more octets representing the length of the file as a binary value,  
 9472 least significant octet first. The smallest number of octets capable of representing this  
 9473 integer shall be used.
- 9474 2. *M(x)* is multiplied by  $x^{32}$  (that is, shifted left 32 bits) and divided by *G(x)* using mod 2  
 9475 division, producing a remainder *R(x)* of degree ≤ 31.
- 9476 3. The coefficients of *R(x)* are considered to be a 32-bit sequence.
- 9477 4. The bit sequence is complemented and the result is the CRC.

9478 **OPTIONS**

9479 None.

9480 **OPERANDS**

9481 The following operand shall be supported:

9482 *file* A path name of a file to be checked. If no *file* operands are specified, the standard  
 9483 input is used.

9484 **STDIN**9485 The standard input is used only if no *file* operands are specified. See the INPUT FILES section.9486 **INPUT FILES**

9487 The input files can be any file type.

9488 **ENVIRONMENT VARIABLES**9489 The following environment variables shall affect the execution of *cksum*:

9490 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 9491 If *LANG* is unset or null, the corresponding value from the implementation-  
 9492 defined default locale shall be used. If any of the internationalization variables  
 9493 contains an invalid setting, the utility shall behave as if none of the variables had  
 9494 been defined.

9495 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 9496 internationalization variables.

- 9497            *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 9498 characters (for example, single-byte as opposed to multi-byte characters in  
 9499 arguments).
- 9500            *LC\_MESSAGES*  
 9501                    Determine the locale that should be used to affect the format and contents of  
 9502 diagnostic messages written to standard error.
- 9503 *XSI*            *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 9504 **ASYNCHRONOUS EVENTS**  
 9505            Default.
- 9506 **STDOUT**  
 9507            For each file processed successfully, the *cksum* utility shall write in the following format:  
 9508            "%u %d %s\n", <checksum>, <# of octets>, <pathname>  
 9509            If no *file* operand was specified, the path name and its leading <space> shall be omitted.
- 9510 **STDERR**  
 9511            Used only for diagnostic messages.
- 9512 **OUTPUT FILES**  
 9513            None.
- 9514 **EXTENDED DESCRIPTION**  
 9515            None.
- 9516 **EXIT STATUS**  
 9517            The following exit values shall be returned:  
 9518            0 All files were processed successfully.  
 9519            >0 An error occurred.
- 9520 **CONSEQUENCES OF ERRORS**  
 9521            Default.
- 9522 **APPLICATION USAGE**  
 9523            The *cksum* utility is typically used to quickly compare a suspect file against a trusted version of  
 9524 the same, such as to ensure that files transmitted over noisy media arrive intact. However, this  
 9525 comparison cannot be considered cryptographically secure. The chances of a damaged file  
 9526 producing the same CRC as the original are small; deliberate deception is difficult, but probably  
 9527 not impossible.
- 9528            Although input files to *cksum* can be any type, the results need not be what would be expected  
 9529 on character special device files or on file types not described by the System Interfaces volume of  
 9530 IEEE Std. 1003.1-200x. Since this volume of IEEE Std. 1003.1-200x does not specify the block size  
 9531 used when doing input, checksums of character special files need not process all of the data in  
 9532 those files.
- 9533            The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted  
 9534 between two systems and undergoes any data transformation (such as moving 8-bit characters  
 9535 into 9-bit bytes or changing little-endian byte ordering to big-endian), identical CRC values  
 9536 cannot be expected. Implementations performing such transformations may extend *cksum* to  
 9537 handle such situations.

9538 **EXAMPLES**

9539 None.

9540 **RATIONALE**

9541 The following C-language program can be used as a model to describe the algorithm. It assumes  
 9542 that a **char** is one octet. It also assumes that the entire file is available for one pass through the  
 9543 function. This was done for simplicity in demonstrating the algorithm, rather than as an  
 9544 implementation model.

```

9545 static unsigned long crctab[] = {
9546 0x00000000,
9547 0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b,
9548 0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6,
9549 0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbd8d,
9550 0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac,
9551 0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f,
9552 0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3, 0x709f7b7a,
9553 0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,
9554 0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58,
9555 0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033,
9556 0xa4ad16ea, 0xa06c0b5d, 0xd4326d90, 0xd0f37027, 0xddb056fe,
9557 0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
9558 0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4,
9559 0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,
9560 0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5,
9561 0x2ac12072, 0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcd8bb16,
9562 0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca, 0x7897ab07,
9563 0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93ddd8, 0x6f52c06c,
9564 0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08, 0x571d7dd1,
9565 0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
9566 0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e, 0xbf1b04b,
9567 0xbb60adfc, 0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698,
9568 0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d,
9569 0x94ea7b2a, 0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,
9570 0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2, 0xc6bcf05f,
9571 0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34,
9572 0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcdfd59, 0x608edb80,
9573 0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
9574 0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f, 0x5c007b8a,
9575 0x58c1663d, 0x558240e4, 0x51435d53, 0x251d3b9e, 0x21dc2629,
9576 0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5, 0x3f9b762c,
9577 0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,
9578 0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623, 0xf12f560e,
9579 0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65,
9580 0xeba91bbc, 0xef68060b, 0xd727bbb6, 0xd3e6a601, 0xdea580d8,
9581 0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
9582 0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7, 0xae3afba2,
9583 0xaafb615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,
9584 0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74,
9585 0x857130c3, 0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,
9586 0x4e8ee645, 0x4a4ffb2, 0x470cdd2b, 0x43cdc09c, 0x7b827d21,
9587 0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a,
9588 0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e, 0x18197087,

```

```

9589 0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
9590 0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088, 0x2497d08d,
9591 0x2056cd3a, 0x2d15ebe3, 0x29d4f654, 0xc5a92679, 0xc1683bce,
9592 0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb,
9593 0xdbee767c, 0xe3alc1, 0xe760d676, 0xea23f0af, 0xee2ed18,
9594 0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4, 0x89b8fd09,
9595 0x8d79e0be, 0x803ac667, 0x84fbdbd0, 0x9abc8bd5, 0x9e7d9662,
9596 0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06, 0xa6322bdf,
9597 0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4
9598 };

9599 unsigned long memcrc(const unsigned char *b, size_t n)
9600 {
9601 /* Input arguments:
9602 * const char* b == byte sequence to checksum
9603 * size_t n == length of sequence
9604 */

9605 register unsigned i, c, s = 0;

9606 for (i = n; i > 0; --i) {
9607 c = (unsigned)(*b++);
9608 s = (s << 8) ^ crctab[(s >> 24) ^ c];
9609 }

9610 /* Extend with the length of the string. */
9611 while (n != 0) {
9612 c = n & 0377;
9613 n >>= 8;
9614 s = (s << 8) ^ crctab[(s >> 24) ^ c];
9615 }

9616 return ~s;
9617 }

```

9618 The historical practice of writing the number of “blocks” has been changed to writing the  
9619 number of octets, since the latter is not only more useful, but also since historical  
9620 implementations have not been consistent in defining what a “block” meant. Octets are used  
9621 instead of bytes because bytes can differ in size between systems.

9622 The algorithm used was selected to increase the operational robustness of *cksum*. Neither the  
9623 System V nor BSD *sum* algorithm was selected. Since each of these was different and each was  
9624 the default behavior on those systems, no realistic compromise was available if either were  
9625 selected—some set of historical applications would break. Therefore, the name was changed to  
9626 *cksum*. Although the historical *sum* commands will probably continue to be provided for many  
9627 years, programs designed for portability across systems should use the new name.

9628 The algorithm selected is based on that used by the ISO/IEC 8802-3: 1996 standard (Ethernet) for  
9629 the frame check sequence field. The algorithm used does not match the technical definition of a  
9630 *checksum*; the term is used for historical reasons. The length of the file is included in the CRC  
9631 calculation because this parallels inclusion of a length field by Ethernet in its CRC, but also  
9632 because it guards against inadvertent collisions between files that begin with different series of  
9633 zero octets. The chance that two different files produce identical CRCs is much greater when  
9634 their lengths are not considered. Keeping the length and the checksum of the file itself separate  
9635 would yield a slightly more robust algorithm, but historical usage has always been that a single  
9636 number (the checksum as printed) represents the signature of the file. It was decided that

9637 historical usage was the more important consideration.

9638 Early proposals contained modifications to the Ethernet algorithm that involved extracting table  
9639 values whenever an intermediate result became zero. This was demonstrated to be less robust  
9640 than the current method and mathematically difficult to describe or justify.

9641 The calculation used is identical to that given in pseudo-code in the referenced Sarwate article.  
9642 The pseudo-code rendition is:

```
9643 X ← 0; Y ← 0;
9644 for i ← m - 1 step -1 until 0 do
9645 begin
9646 T ← X(1) ^ A[i];
9647 X(1) ← X(0); X(0) ← Y(1); Y(1) ← Y(0); Y(0) ← 0;
9648 comment: f[T] and f'[T] denote the T-th words in the
9649 table f and f' ;
9650 X ← X ^ f[T]; Y ← Y ^ f'[T];
9651 end
```

9652 The pseudo-code is reproduced exactly as given; however, note that in the case of *cksum*, **A[i]**  
9653 represents a byte of the file, the words **X** and **Y** are treated as a single 32-bit value, and the tables  
9654 **f** and **f'** are a single table containing 32-bit values.

9655 The referenced Sarwate article also discusses generating the table.

9656 **FUTURE DIRECTIONS**

9657 None.

9658 **SEE ALSO**

9659 None.

9660 **CHANGE HISTORY**

9661 First released in Issue 4.

9662 **NAME**

9663 cmp — compare two files

9664 **SYNOPSIS**9665 cmp [ -l | -s ] *file1 file2*9666 **DESCRIPTION**

9667 The *cmp* utility shall compare two files. The *cmp* utility writes no output if the files are the same.  
 9668 Under default options, if they differ, it shall write to standard output the byte and line number at  
 9669 which the first difference occurred. Bytes and lines shall be numbered beginning with 1.

9670 **OPTIONS**

9671 The *cmp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 9672 12.2, Utility Syntax Guidelines.

9673 The following options shall be supported:

9674 **-l** (Lowercase ell.) Write the byte number (decimal) and the differing bytes (octal) for  
 9675 each difference.

9676 **-s** Write nothing for differing files; return exit status only.

9677 **OPERANDS**

9678 The following operands shall be supported:

9679 *file1* A path name of the first file to be compared. If *file1* is '-', the standard input shall  
 9680 be used.

9681 *file2* A path name of the second file to be compared. If *file2* is '-', the standard input  
 9682 shall be used.

9683 If both *file1* and *file2* refer to standard input or refer to the same FIFO special, block special, or  
 9684 character special file, the results are undefined.

9685 **STDIN**

9686 The standard input shall be used only if the *file1* or *file2* operand refers to standard input. See the  
 9687 INPUT FILES section.

9688 **INPUT FILES**

9689 The input files can be any file type.

9690 **ENVIRONMENT VARIABLES**9691 The following environment variables shall affect the execution of *cmp*:

9692 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 9693 If **LANG** is unset or null, the corresponding value from the implementation-  
 9694 defined default locale shall be used. If any of the internationalization variables  
 9695 contains an invalid setting, the utility shall behave as if none of the variables had  
 9696 been defined.

9697 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 9698 internationalization variables.

9699 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 9700 characters (for example, single-byte as opposed to multi-byte characters in  
 9701 arguments).

9702 **LC\_MESSAGES**

9703 Determine the locale that should be used to affect the format and contents of  
 9704 diagnostic messages written to standard error and informative messages written to  
 9705 standard output.

9706 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

9707 **ASYNCHRONOUS EVENTS**

9708 Default.

9709 **STDOUT**

9710 In the POSIX locale, results of the comparison shall be written to standard output. When no  
9711 options are used, the format shall be:

9712 "%s %s differ: char %d, line %d\n", *file1*, *file2*,  
9713 <byte number>, <line number>

9714 When the **-I** option is used, the format shall be:

9715 "%d %o %o\n", <byte number>, <differing byte>,  
9716 <differing byte>

9717 for each byte that differs. The first <differing byte> number is from *file1* while the second is from  
9718 *file2*. In both cases, <byte number> shall be relative to the beginning of the file, beginning with 1.

9719 No output shall be written to standard output when the **-s** option is used.

9720 **STDERR**

9721 Used only for diagnostic messages. If *file1* and *file2* are identical for the entire length of the  
9722 shorter file, in the POSIX locale the following diagnostic message shall be written, unless the **-s**  
9723 option is specified:

9724 "cmp: EOF on %s%s\n", <name of shorter file>, <additional info>

9725 The <additional info> field shall either be null or a string that starts with a <blank> character and  
9726 contains no <newline> characters. Some systems report on the number of lines in this case.

9727 **OUTPUT FILES**

9728 None.

9729 **EXTENDED DESCRIPTION**

9730 None.

9731 **EXIT STATUS**

9732 The following exit values shall be returned:

9733 0 The files are identical.

9734 1 The files are different; this includes the case where one file is identical to the first part of the  
9735 other.

9736 >1 An error occurred.

9737 **CONSEQUENCES OF ERRORS**

9738 Default.

9739 **APPLICATION USAGE**

9740 Although input files to *cmp* can be any type, the results might not be what would be expected on  
9741 character special device files or on file types not described by the System Interfaces volume of  
9742 IEEE Std. 1003.1-200x. Since this volume of IEEE Std. 1003.1-200x does not specify the block size  
9743 used when doing input, comparisons of character special files need not compare all of the data  
9744 in those files.

9745 For files which are not text files, line numbers simply reflect the presence of a <newline>  
9746 character, without any implication that the file is organized into lines.



9747 **EXAMPLES**

9748 None.

9749 **RATIONALE**

9750 The global language in Section 1.11 (on page 2224) indicates that using two mutually-exclusive  
9751 options together produces unspecified results. Some System V implementations consider the  
9752 option usage:

9753 `cmp -l -s ...`

9754 to be an error. They also treat:

9755 `cmp -s -l ...`

9756 as if no options were specified. Both of these behaviors are considered bugs, but are allowed.

9757 The word **char** in the standard output format comes from historical usage, even though it is  
9758 actually a byte number. When *cmp* is supported in other locales, implementations are  
9759 encouraged to use the word *byte* or its equivalent in another language. Users should not  
9760 interpret this difference to indicate that the functionality of the utility changed between locales.

9761 Some systems report on the number of lines in the identical-but-shorter file case. This is allowed  
9762 by the inclusion of the *<additional info>* fields in the output format. The restriction on having a  
9763 leading *<blank>* and no *<newline>*s is to make parsing for the file name easier. It is recognized  
9764 that some file names containing white-space characters make parsing difficult anyway, but the  
9765 restriction does aid programs used on systems where the names are predominantly well  
9766 behaved.

9767 **FUTURE DIRECTIONS**

9768 None.

9769 **SEE ALSO**9770 *comm, diff*9771 **CHANGE HISTORY**

9772 First released in Issue 2.

9773 **Issue 4**

9774 Aligned with the ISO/IEC 9945-2:1993 standard.

9775 **NAME**

9776 comm — select or reject lines common to two files

9777 **SYNOPSIS**

9778 comm [-123] *file1 file2*

9779 **DESCRIPTION**

9780 The *comm* utility shall read *file1* and *file2*, which should be ordered in the current collating  
9781 sequence, and produce three text columns as output: lines only in *file1*, lines only in *file2*, and  
9782 lines in both files.

9783 If the lines in both files are not ordered according to the collating sequence of the current locale,  
9784 the results are unspecified.

9785 **OPTIONS**

9786 The *comm* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
9787 12.2, Utility Syntax Guidelines.

9788 The following options shall be supported:

9789 -1 Suppress the output column of lines unique to *file1*.

9790 -2 Suppress the output column of lines unique to *file2*.

9791 -3 Suppress the output column of lines duplicated in *file1* and *file2*.

9792 **OPERANDS**

9793 The following operands shall be supported:

9794 *file1* A path name of the first file to be compared. If *file1* is '-', the standard input is  
9795 used.

9796 *file2* A path name of the second file to be compared. If *file2* is '-', the standard input is  
9797 used.

9798 If both *file1* and *file2* refer to standard input or to the same FIFO special, block special, or  
9799 character special file, the results are undefined.

9800 **STDIN**

9801 The standard input shall be used only if one of the *file1* or *file2* operands refers to standard input.  
9802 See the INPUT FILES section.

9803 **INPUT FILES**

9804 The input files shall be text files.

9805 **ENVIRONMENT VARIABLES**

9806 The following environment variables shall affect the execution of *comm*:

9807 *LANG* Provide a default value for the internationalization variables that are unset or null.  
9808 If *LANG* is unset or null, the corresponding value from the implementation-  
9809 defined default locale shall be used. If any of the internationalization variables  
9810 contains an invalid setting, the utility shall behave as if none of the variables had  
9811 been defined.

9812 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
9813 internationalization variables.

9814 *LC\_COLLATE*

9815 Determine the locale for the collating sequence *comm* expects to have been used  
9816 when the input files were sorted.

9817 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 9818 characters (for example, single-byte as opposed to multi-byte characters in  
 9819 arguments and input files).

9820 **LC\_MESSAGES**  
 9821 Determine the locale that should be used to affect the format and contents of  
 9822 diagnostic messages written to standard error.

9823 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

9824 **ASYNCHRONOUS EVENTS**  
 9825 Default.

9826 **STDOUT**  
 9827 The *comm* utility shall produce output depending on the options selected. If the **-1**, **-2**, and **-3**  
 9828 options are all selected, *comm* shall write nothing to standard output.

9829 If the **-1** option is not selected, lines contained only in *file1* shall be written using the format:  
 9830 "%s\n", <line in file1>

9831 If the **-2** option is not selected, lines contained only in *file2* are written using the format:  
 9832 "%s%s\n", <lead>, <line in file2>

9833 where the string <lead> is as follows:  
 9834 <tab> The **-1** option is not selected.  
 9835 null string The **-1** option is selected.

9836 If the **-3** option is not selected, lines contained in both files shall be written using the format:  
 9837 "%s%s\n", <lead>, <line in both>

9838 where the string <lead> is as follows:  
 9839 <tab><tab> Neither the **-1** nor the **-2** option is selected.  
 9840 <tab> Exactly one of the **-1** and **-2** options is selected.  
 9841 null string Both the **-1** and **-2** options are selected.

9842 If the input files were ordered according to the collating sequence of the current locale, the lines  
 9843 written shall be in the collating sequence of the original lines.

9844 **STDERR**  
 9845 Used only for diagnostic messages.

9846 **OUTPUT FILES**  
 9847 None.

9848 **EXTENDED DESCRIPTION**  
 9849 None.

9850 **EXIT STATUS**  
 9851 The following exit values shall be returned:  
 9852 0 All input files were successfully output as specified.  
 9853 >0 An error occurred.

9854 **CONSEQUENCES OF ERRORS**

9855 Default.

9856 **APPLICATION USAGE**9857 If the input files are not properly presorted, the output of *comm* might not be useful.9858 **EXAMPLES**

9859 If a file named **xcu** contains a sorted list of the utilities in this volume of IEEE Std. 1003.1-200x, a  
9860 file named **xpg3** contains a sorted list of the utilities specified in the X/Open Portability Guide,  
9861 Issue 3, and a file named **svid89** contains a sorted list of the utilities in the System V Interface  
9862 Definition Third Edition:

9863 `comm -23 xcu xpg3 | comm -23 - svid89`9864 would print a list of utilities in this volume of IEEE Std. 1003.1-200x not specified by either of the  
9865 other documents:9866 `comm -12 xcu xpg3 | comm -12 - svid89`

9867 would print a list of utilities specified by all three documents, and:

9868 `comm -12 xpg3 svid89 | comm -23 - xcu`9869 would print a list of utilities specified by both XPG3 and the SVID, but not specified in this  
9870 volume of IEEE Std. 1003.1-200x.9871 **RATIONALE**

9872 None.

9873 **FUTURE DIRECTIONS**

9874 None.

9875 **SEE ALSO**9876 *cmp, diff, sort, uniq*9877 **CHANGE HISTORY**

9878 First released in Issue 2.

9879 **Issue 4**

9880 Aligned with the ISO/IEC 9945-2:1993 standard.

9881 **Issue 6**

9882 The normative text is reworded to avoid use of the term “must” for application requirements.

9883 **NAME**9884 `command` — execute a simple command9885 **SYNOPSIS**9886 `command [-p] command_name [argument ...]`9887 UP `command [ -v | -V ] command_name`

9888

9889 **DESCRIPTION**9890 The *command* utility shall cause the shell to treat the arguments as a simple command, suppressing the shell function lookup that is described in Section 2.9.1.1 (on page 2257), item 1b.9892 If the *command\_name* is the same as the name of one of the special built-in utilities, the special properties in the enumerated list at the beginning of Section 2.15 (on page 2276) shall not occur. In every other respect, if *command\_name* is not the name of a function, the effect of *command* shall be the same as omitting *command*.9896 On systems supporting the User Portability Utilities option, the *command* utility also shall provide information concerning how a command name is interpreted by the shell; see `-v` and `-V`.9899 **OPTIONS**9900 The *command* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

9902 The following options shall be supported:

9903 `-p` Perform the command search using a default value for *PATH* that is guaranteed to find all of the standard utilities.9905 `-v` (On systems supporting the User Portability Utilities option.) Write a string to standard output that indicates the path name or command that will be used by the shell, in the current shell execution environment (see Section 2.13 (on page 2273)), to invoke *command\_name*.9909 • Utilities, regular built-in utilities, *command\_names* including a slash character, and any implementation-defined functions that are found using the *PATH* variable (as described in Section 2.9.1.1 (on page 2257)), shall be written as absolute path names.9913 • Shell functions, special built-in utilities, regular built-in utilities not associated with a *PATH* search, and shell reserved words shall be written as just their names.

9916 • An alias shall be written as a command line that represents its alias definition.

9917 • Otherwise, no output shall be written and the exit status shall reflect that the name was not found.

9919 `-V` (On systems supporting the User Portability Utilities option.) Write a string to standard output that indicates how the name given in the *command\_name* operand will be interpreted by the shell, in the current shell execution environment (see Section 2.13 (on page 2273)). Although the format of this string is unspecified, it shall indicate in which of the following categories *command\_name* falls and shall include the information stated:9925 • Utilities, regular built-in utilities, and any implementation-defined functions that are found using the *PATH* variable (as described in Section 2.9.1.1 (on page 2257)), shall be identified as such and include the absolute path name in the

- 9928 string.
- 9929 • Other shell functions shall be identified as functions.
- 9930 • Aliases shall be identified as aliases and their definitions included in the string.
- 9931 • Special built-in utilities shall be identified as special built-in utilities.
- 9932 • Regular built-in utilities not associated with a *PATH* search shall be identified
- 9933 as regular built-in utilities. (The term “regular” need not be used.)
- 9934 • Shell reserved words shall be identified as reserved words.
- 9935 **OPERANDS**
- 9936 The following operands shall be supported:
- 9937 *argument* One of the strings treated as an argument to *command\_name*.
- 9938 *command\_name*
- 9939 The name of a utility or a special built-in utility.
- 9940 **STDIN**
- 9941 Not used.
- 9942 **INPUT FILES**
- 9943 None.
- 9944 **ENVIRONMENT VARIABLES**
- 9945 The following environment variables shall affect the execution of *command*:
- 9946 *LANG* Provide a default value for the internationalization variables that are unset or null.
- 9947 If *LANG* is unset or null, the corresponding value from the implementation-
- 9948 defined default locale shall be used. If any of the internationalization variables
- 9949 contains an invalid setting, the utility shall behave as if none of the variables had
- 9950 been defined.
- 9951 *LC\_ALL* If set to a non-empty string value, override the values of all the other
- 9952 internationalization variables.
- 9953 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
- 9954 characters (for example, single-byte as opposed to multi-byte characters in
- 9955 arguments).
- 9956 *LC\_MESSAGES*
- 9957 Determine the locale that should be used to affect the format and contents of
- 9958 diagnostic messages written to standard error and informative messages written to
- 9959 standard output.
- 9960 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 9961 *PATH* Determine the search path used during the command search described in Section
- 9962 2.9.1.1 (on page 2257), except as described under the *-p* option.
- 9963 **ASYNCHRONOUS EVENTS**
- 9964 Default.
- 9965 **STDOUT**
- 9966 When the *-v* option is specified, standard output shall be formatted as:
- 9967 "%s\n", *<pathname or command>*
- 9968 When the *-V* option is specified, standard output shall be formatted as:

9969 "%s\n", <unspecified>

#### 9970 **STDERR**

9971 Used only for diagnostic messages.

#### 9972 **OUTPUT FILES**

9973 None.

#### 9974 **EXTENDED DESCRIPTION**

9975 None.

#### 9976 **EXIT STATUS**

9977 When the `-v` or `-V` options are specified, the following exit values shall be returned:

9978 0 Successful completion.

9979 >0 The *command\_name* could not be found or an error occurred.

9980 Otherwise, the following exit values shall be returned:

9981 126 The utility specified by *command\_name* was found but could not be invoked.

9982 127 An error occurred in the *command* utility or the utility specified by *command\_name* could not  
9983 be found.

9984 Otherwise, the exit status of *command* shall be that of the simple command specified by the  
9985 arguments to *command*.

#### 9986 **CONSEQUENCES OF ERRORS**

9987 Default.

#### 9988 **APPLICATION USAGE**

9989 The order for command search allows functions to override regular built-ins and path searches.  
9990 This utility is necessary to allow functions that have the same name as a utility to call the utility  
9991 (instead of a recursive call to the function).

9992 The system default path is available using *getconf*; however, since *getconf* may need to have the  
9993 *PATH* set up before it can be called itself, the following can be used:

9994 `command -p getconf _CS_PATH`

9995 There are some advantages to suppressing the special characteristics of special built-ins on  
9996 occasion. For example:

9997 `command exec > unwritable-file`

9998 does not cause a non-interactive script to abort, so that the output status can be checked by the  
9999 script.

10000 The *command*, *env*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if an  
10001 error occurs so that applications can distinguish “failure to find a utility” from “invoked utility  
10002 exited with an error indication”. The value 127 was chosen because it is not commonly used for  
10003 other meanings; most utilities use small values for “normal error conditions” and the values  
10004 above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen  
10005 in a similar manner to indicate that the utility could be found, but not invoked. Some scripts  
10006 produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
10007 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
10008 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
10009 any other reason.

10010 Since the `-v` and `-V` options of *command* produce output in relation to the current shell execution  
10011 environment, *command* is generally provided as a shell regular built-in. If it is called in a subshell

10012 or separate utility execution environment, such as one of the following:

```
10013 (PATH=foo command -v)
10014 nohup command -v
```

10015 it does not necessarily produce correct results. For example, when called with *nohup* or an *exec*  
10016 function, in a separate utility execution environment, most implementations are not able to  
10017 identify aliases, functions, or special built-ins.

10018 Two types of regular built-ins could be encountered on a system and these are described  
10019 separately by *command*. The description of command search in Section 2.9.1.1 (on page 2257)  
10020 allows for a standard utility to be implemented as a regular built-in as long as it is found in the  
10021 appropriate place in a *PATH* search. So, for example, *command -v true* might yield */bin/true* or  
10022 some similar path name. Other implementation-defined utilities that are not defined by this  
10023 volume of IEEE Std. 1003.1-200x might exist only as built-ins and have no path name associated  
10024 with them. These produce output identified as (regular) built-ins. Applications encountering  
10025 these are not able to count on *execing* them, using them with *nohup*, overriding them with a  
10026 different *PATH*, and so on.

#### 10027 EXAMPLES

10028 1. Make a version of *cd* that always prints out the new working directory exactly once:

```
10029 cd() {
10030 command cd "$@" >/dev/null
10031 pwd
10032 }
```

10033 2. Start off a “secure shell script” in which the script avoids being spoofed by its parent:

```
10034 IFS='
10035 '
10036 # The preceding value should be <space><tab><newline>.
10037 # Set IFS to its default value.
```

```
10038 \unalias -a
10039 # Unset all possible aliases.
10040 # Note that unalias is escaped to prevent an alias
10041 # being used for unalias.
```

```
10042 unset -f command
10043 # Ensure command is not a user function.
```

```
10044 PATH="$(command -p getconf _CS_PATH):$PATH"
10045 # Put on a reliable PATH prefix.
```

```
10046 # ...
```

10047 At this point, given correct permissions on the directories called by *PATH*, the script has  
10048 the ability to ensure that any utility it calls is the intended one. It is being very cautious  
10049 because it assumes that implementation extensions may be present that would allow user  
10050 functions to exist when it is invoked; this capability is not specified by this volume of  
10051 IEEE Std. 1003.1-200x, but it is not prohibited as an extension. For example, the *ENV*  
10052 variable precedes the invocation of the script with a user start-up script. Such a script  
10053 could define functions to spoof the application.



10054 **RATIONALE**

10055 Since *command* is a regular built-in utility it is always found prior to the *PATH* search.

10056 There is nothing in the description of *command* that implies the command line is parsed any  
10057 differently from that of any other simple command. For example:

10058 `command a | b ; c`

10059 is not parsed in any special way that causes ' | ' or ' ; ' to be treated other than a pipe operator  
10060 or semicolon or that prevents function lookup on **b** or **c**.

10061 The *command* utility is somewhat similar to the Eighth Edition shell *builtin* command, but since  
10062 *command* also goes to the file system to search for utilities, the name *builtin* would not be  
10063 intuitive.

10064 The *command* utility is most likely to be provided as a regular built-in. It is not listed as a special  
10065 built-in for the following reasons:

- 10066 • The removal of exportable functions made the special precedence of a special built-in  
10067 unnecessary.
- 10068 • A special built-in has special properties (see Section 2.15 (on page 2276)) that were  
10069 inappropriate for invoking other utilities. For example, two commands such as:

10070 `date > unwritable-file`

10071 `command date > unwritable-file`

10072 would have entirely different results; in a non-interactive script, the former would continue  
10073 to execute the next command, the latter would abort. Introducing this semantic difference  
10074 along with suppressing functions was seen to be non-intuitive.

10075 The **-p** option is present because it is useful to be able to ensure a safe path search that finds all  
10076 the POSIX Shell and Utilities standard utilities. This search might not be identical to the one that  
10077 occurs through one of the POSIX System Interfaces *exec* functions when *PATH* is unset. At the  
10078 very least, this feature is required to allow the script to access the correct version of *getconf* so  
10079 that the value of the default path can be accurately retrieved.

10080 The *command* **-v** and **-V** options were added to satisfy requirements from users that are  
10081 currently accomplished by three different historical utilities: *type* in the System V shell, *whence*  
10082 in the KornShell, and *which* in the C shell. Since there is no historical agreement on how and what  
10083 to accomplish here, the POSIX *command* utility was enhanced and the historical utilities were left  
10084 unmodified. The C shell *which* merely conducts a path search. The KornShell *whence* is more  
10085 elaborate—in addition to the categories required by POSIX, it also reports on tracked aliases,  
10086 exported aliases, and undefined functions.

10087 The output format of **-V** was left mostly unspecified because human users are its only audience.  
10088 Applications should not be written to care about this information; they can use the output of **-v**  
10089 to differentiate between various types of commands, but the additional information that may be  
10090 emitted by the more verbose **-V** is not needed and should not be arbitrarily constrained in its  
10091 verbosity or localization for application parsing reasons.

10092 **FUTURE DIRECTIONS**

10093 None.

10094 **SEE ALSO**

10095 *sh*, *type*

10096 **CHANGE HISTORY**

10097 First released in Issue 4.

## 10098 NAME

10099 compress — compress data

## 10100 SYNOPSIS

10101 xSI compress [-fv][-b *bits*][*file* ...]10102 compress [-cfv][-b *bits*][*file*]

10103

## 10104 DESCRIPTION

10105 **Notes to Reviewers**10106 *This section with side shading will not appear in the final copy. - Ed.*10107 We need to cite the patent number for Lempel-Ziv coding; if anyone knows what it is, please let  
10108 us know.10109 The *compress* utility shall attempt to reduce the size of the named files by using adaptive  
10110 Lempel-Ziv coding algorithm. On systems not supporting adaptive Lempel-Ziv coding  
10111 algorithm, the input files shall not be changed and an error value greater than two shall be  
10112 returned. Except when the output is to the standard output, each file shall be replaced by one  
10113 with the extension *.Z*. If the invoking process has appropriate privileges, the ownership, modes,  
10114 access time, and modification time of the original file are preserved. If appending the *.Z* to the  
10115 file name would make the name exceed {NAME\_MAX} bytes, the command shall fail. If no files  
10116 are specified, the standard input shall be compressed to the standard output.

## 10117 OPTIONS

10118 The *compress* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,  
10119 Section 12.2, Utility Syntax Guidelines.

10120 The following options shall be supported:

10121 **-b *bits*** Specify the maximum number of bits to use in a code. For a portable application,  
10122 the *bits* argument shall be:10123  $9 \leq bits \leq 14$ 10124 The implementation may allow *bits* values of greater than 14. The default is 14, 15,  
10125 or 16.10126 **-c** Cause *compress* to write to the standard output; the input file is not changed, and  
10127 no *.Z* files are created.10128 **-f** Force compression of *file*, even if it does not actually reduce the size of the file, or if  
10129 the corresponding *file.Z* file already exists. If the **-f** option is not given, and the  
10130 process is not running in the background, the user is prompted as to whether an  
10131 existing *file.Z* file should be overwritten.10132 **-v** Write the percentage reduction of each file to standard error.

## 10133 OPERANDS

10134 The following operand shall be supported:

10135 *file* A path name of a file to be compressed.

## 10136 STDIN

10137 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '- '.

10138 **INPUT FILES**

10139 If *file* operands are specified, the input files contain the data to be compressed.

10140 **ENVIRONMENT VARIABLES**

10141 The following environment variables shall affect the execution of *compress*:

10142 *LANG* Provide a default value for the internationalization variables that are unset or null.  
10143 If *LANG* is unset or null, the corresponding value from the implementation-  
10144 defined default locale shall be used. If any of the internationalization variables  
10145 contains an invalid setting, the utility shall behave as if none of the variables had  
10146 been defined.

10147 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
10148 internationalization variables.

10149 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
10150 characters (for example, single-byte as opposed to multi-byte characters in  
10151 arguments).

10152 *LC\_MESSAGES*

10153 Determine the locale that should be used to affect the format and contents of  
10154 diagnostic messages written to standard error.

10155 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

10156 **ASYNCHRONOUS EVENTS**

10157 Default.

10158 **STDOUT**

10159 If no *file* operands are specified, or if a *file* operand is '-', or if the -c option is specified, the  
10160 standard output contains the compressed output.

10161 **STDERR**

10162 Used for all diagnostic and prompt messages and the output from -v.

10163 **OUTPUT FILES**

10164 The output files shall contain the compressed output. The format of compressed files is  
10165 unspecified and interchange of such files between implementations (including access via  
10166 unspecified file sharing mechanisms) is not required by IEEE Std. 1003.1-200x.

10167 **EXTENDED DESCRIPTION**

10168 None.

10169 **EXIT STATUS**

10170 The following exit values shall be returned:

10171 0 Successful completion.

10172 1 An error occurred.

10173 2 One or more files were not compressed because they would have increased in size (and the  
10174 -f option was not specified).

10175 >2 An error occurred.

10176 **CONSEQUENCES OF ERRORS**

10177 The input file shall remain unmodified.

10178 **APPLICATION USAGE**

10179 The amount of compression obtained depends on the size of the input, the number of *bits* per  
10180 code, and the distribution of common substrings. Typically, text such as source code or English  
10181 is reduced by 50-60%. Compression is generally much better than that achieved by Huffman  
10182 coding or adaptive Huffman coding (*compact*), and takes less time to compute.

10183 Although *compress* strictly follows the default actions upon receipt of a signal or when an error  
10184 occurs, some unexpected results may occur. In some implementations it is likely that a partially  
10185 compressed file is left in place, alongside its uncompressed input file. Since the general  
10186 operation of *compress* is to delete the uncompressed file only after the *.Z* file has been  
10187 successfully filled, an application should always carefully check the exit status of *compress* before  
10188 arbitrarily deleting files that have like-named neighbors with *.Z* suffixes.

10189 The limit of 14 on the *bits* option-argument is to achieve portability to all systems (within the  
10190 restrictions imposed by the lack of an explicit published file format). Some systems based on  
10191 16-bit architectures cannot support 15 or 16-bit uncompression.

10192 **EXAMPLES**

10193 None.

10194 **RATIONALE**

10195 None.

10196 **FUTURE DIRECTIONS**

10197 None.

10198 **SEE ALSO**

10199 *uncompress, zcat*

10200 **CHANGE HISTORY**

10201 First released in Issue 4.

10202 **Issue 4, Version 2**

10203 The DESCRIPTION section is clarified to state that the ownership, modes, access time, and  
10204 modification time of the original file are preserved if the invoking process has appropriate  
10205 privileges.

10206 The STDOUT section includes the case where a *file* operand is *'-'*.

10207 **Issue 6**

10208 The normative text is reworded to avoid use of the term “must” for application requirements.

10209 An error case is added for systems not supporting adaptive Lempel-Ziv coding.

## 10210 NAME

10211 cp — copy files

## 10212 SYNOPSIS

10213 cp [-fip] *source\_file target\_file*10214 cp [-fip] *source\_file ... target*10215 cp -R [-H | -L | -P][-fip] *source\_file ... target*10216 OB cp -r [-H | -L | -P][-fip] *source\_file ... target*

## 10217 DESCRIPTION

10218 The first synopsis form is denoted by two operands, neither of which are existing files of type  
 10219 directory. The *cp* utility shall copy the contents of *source\_file* (or, if *source\_file* is a file of type  
 10220 symbolic link, the contents of the file referenced by *source\_file*) to the destination path named by  
 10221 *target\_file*.

10222 The second synopsis form is denoted by two or more operands where the **-R** or **-r** options are  
 10223 not specified and the first synopsis form is not applicable. It shall be an error if any *source\_file* is a  
 10224 file of type directory, if *target* does not exist, or if *target* is a file of a type defined by the System  
 10225 Interfaces volume of IEEE Std. 1003.1-200x, but is not a file of type directory. The *cp* utility shall  
 10226 copy the contents of each *source\_file* (or, if *source\_file* is a file of type symbolic link, the contents  
 10227 of the file referenced by *source\_file*) to the destination path named by the concatenation of *target*,  
 10228 a slash character, and the last component of *source\_file*.

10229 The third and fourth synopsis forms are denoted by two or more operands where the **-R** or **-r**  
 10230 options are specified. The *cp* utility shall copy each file in the file hierarchy rooted in each  
 10231 *source\_file* to a destination path named as follows.

10232 If *target* exists and is a file of type directory, the name of the corresponding destination path for  
 10233 each file in the file hierarchy shall be the concatenation of *target*, a slash character, and the path  
 10234 name of the file relative to the directory containing *source\_file*.

10235 If *target* does not exist and two operands are specified, the name of the corresponding  
 10236 destination path for *source\_file* shall be *target*; the name of the corresponding destination path for  
 10237 all other files in the file hierarchy shall be the concatenation of *target*, a slash character, and the  
 10238 path name of the file relative to *source\_file*.

10239 It shall be an error if *target* does not exist and more than two operands are specified, or if *target*  
 10240 exists and is a file of a type defined by the System Interfaces volume of IEEE Std. 1003.1-200x,  
 10241 but is not a file of type directory.

10242 In the following description, the term *dest\_file* refers to the file named by the destination path.  
 10243 The term *source\_file* refers to the file that is being copied, whether specified as an operand or a  
 10244 file in a file hierarchy rooted in a *source\_file* operand. If *source\_file* is a file of type symbolic link:

- 10245 • If neither the **-R** nor **-r** options were specified, *cp* shall take actions based on the type and  
 10246 contents of the file referenced by the symbolic link, and not by the symbolic link itself.
- 10247 • If the **-R** option was specified:
  - 10248 — If none of the options **-H**, **-L**, nor **-P** were specified, it is unspecified which of **-H**, **-L**, or  
 10249 **-P** will be used as a default.
  - 10250 — If the **-H** option was specified, *cp* shall take actions based on the type and contents of the  
 10251 file referenced by any symbolic link specified as a *source\_file* operand.
  - 10252 — If the **-L** option was specified, *cp* shall take actions based on the type and contents of the  
 10253 file referenced by any symbolic link specified as a *source\_file* operand or any symbolic

- 10254 links encountered during traversal of a file hierarchy.
- 10255 — If the **-P** option was specified, *cp* shall copy any symbolic link specified as a *source\_file*  
 10256 operand and any symbolic links encountered during traversal of a file hierarchy, and shall  
 10257 not follow any symbolic links.
- 10258 • If the **-r** option was specified, the behavior is implementation-defined.
- 10259 For each *source\_file*, the following steps shall be taken:
- 10260 1. If *source\_file* references the same file as *dest\_file*, *cp* may write a diagnostic message to  
 10261 standard error; it shall do nothing more with *source\_file* and shall go on to any remaining  
 10262 files.
- 10263 2. If *source\_file* is of type directory, the following steps shall be taken:
- 10264 a. If neither the **-R** or **-r** options were specified, *cp* shall write a diagnostic message to  
 10265 standard error, do nothing more with *source\_file*, and go on to any remaining files.
- 10266 b. If *source\_file* was not specified as an operand and *source\_file* is dot or dot-dot, *cp* shall  
 10267 do nothing more with *source\_file* and go on to any remaining files.
- 10268 c. If *dest\_file* exists and it is a file type not specified by the System Interfaces volume of  
 10269 IEEE Std. 1003.1-200x, the behavior is implementation-defined.
- 10270 d. If *dest\_file* exists and it is not of type directory, *cp* shall write a diagnostic message to  
 10271 standard error, do nothing more with *source\_file* or any files below *source\_file* in the  
 10272 file hierarchy, and go on to any remaining files.
- 10273 e. If the directory *dest\_file* does not exist, it shall be created with file permission bits set  
 10274 to the same value as those of *source\_file*, modified by the file creation mask of the  
 10275 user if the **-p** option was not specified, and then bitwise-inclusively OR'ed with  
 10276 S\_IRWXU. If *dest\_file* cannot be created, *cp* shall write a diagnostic message to  
 10277 standard error, do nothing more with *source\_file*, and go on to any remaining files. It  
 10278 is unspecified if *cp* attempts to copy files in the file hierarchy rooted in *source\_file*.
- 10279 f. The files in the directory *source\_file* shall be copied to the directory *dest\_file*, taking  
 10280 the four steps [1-4] listed here with the files as *source\_files*.
- 10281 g. If *dest\_file* was created, its file permission bits shall be changed (if necessary) to be the  
 10282 same as those of *source\_file*, modified by the file creation mask of the user if the **-p**  
 10283 option was not specified.
- 10284 h. The *cp* utility shall do nothing more with *source\_file* and go on to any remaining files.
- 10285 3. If *source\_file* is of type regular file, the following steps shall be taken:
- 10286 a. If *dest\_file* exists, the following steps shall be taken:
- 10287 i. If the **-i** option is in effect, the *cp* utility shall write a prompt to the standard  
 10288 error and read a line from the standard input. If the response is not affirmative,  
 10289 *cp* shall do nothing more with *source\_file* and go on to any remaining files.
- 10290 ii. A file descriptor for *dest\_file* shall be obtained by performing actions equivalent  
 10291 to the *open()* function defined in the System Interfaces volume of  
 10292 IEEE Std. 1003.1-200x called using *dest\_file* as the *path* argument, and the  
 10293 bitwise-inclusive OR of O\_WRONLY and O\_TRUNC as the *oflag* argument.
- 10294 iii. If the attempt to obtain a file descriptor fails and the **-f** option is in effect, *cp*  
 10295 shall attempt to remove the file by performing actions equivalent to the  
 10296 *unlink()* function defined in the System Interfaces volume of

- 10297 IEEE Std. 1003.1-200x called using *dest\_file* as the *path* argument. If this attempt  
10298 succeeds, *cp* shall continue with step 3b.
- 10299 b. If *dest\_file* does not exist, a file descriptor shall be obtained by performing actions  
10300 equivalent to the *open()* function defined in the System Interfaces volume of  
10301 IEEE Std. 1003.1-200x called using *dest\_file* as the *path* argument, and the bitwise-  
10302 inclusive OR of *O\_WRONLY* and *O\_CREAT* as the *oflag* argument. The file  
10303 permission bits of *source\_file* shall be the *mode* argument.
- 10304 c. If the attempt to obtain a file descriptor fails, *cp* shall write a diagnostic message to  
10305 standard error, do nothing more with *source\_file*, and go on to any remaining files.
- 10306 d. The contents of *source\_file* shall be written to the file descriptor. Any write errors  
10307 shall cause *cp* to write a diagnostic message to standard error and continue to step 3e.
- 10308 e. The file descriptor shall be closed.
- 10309 f. The *cp* utility shall do nothing more with *source\_file*. If a write error occurred in step  
10310 3d, it is unspecified if *cp* continues with any remaining files. If no write error  
10311 occurred in step 3d, *cp* shall go on to any remaining files.
- 10312 4. Otherwise, the following steps shall be taken:
- 10313 a. If the **-r** option was specified, the behavior is implementation-defined.
- 10314 b. If the **-R** option was specified, the following steps shall be taken:
- 10315 i. The *dest\_file* shall be created with the same file type as *source\_file*.
- 10316 ii. If *source\_file* is a file of type FIFO, the file permission bits shall be the same as  
10317 those of *source\_file*, modified by the file creation mask of the user if the **-p**  
10318 option was not specified. Otherwise, the permissions, owner ID, and group ID  
10319 of *dest\_file* are implementation-defined.
- 10320 If this creation fails for any reason, *cp* shall write a diagnostic message to  
10321 standard error, do nothing more with *source\_file*, and go on to any remaining  
10322 files.
- 10323 iii. If *source\_file* is a file of type symbolic link, the path name contained in *dest\_file*  
10324 shall be the same as the path name contained in *source\_file*.
- 10325 If this fails for any reason, *cp* shall write a diagnostic message to standard error,  
10326 do nothing more with *source\_file*, and go on to any remaining files.
- 10327 If the implementation provides additional or alternate access control mechanisms (see the Base  
10328 Definitions volume of IEEE Std. 1003.1-200x, Section 4.1, File Access Permissions), their effect on  
10329 copies of files is implementation-defined.

#### 10330 OPTIONS

- 10331 The *cp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
10332 Utility Syntax Guidelines.
- 10333 The following options shall be supported:
- 10334 **-f** If a file descriptor for a destination file cannot be obtained, as described in step  
10335 3.a.ii., attempt to unlink the destination file and proceed.
- 10336 **-H** Take actions based on the type and contents of the file referenced by any symbolic  
10337 link specified as a *source\_file* operand.
- 10338 **-i** Write a prompt to standard error before copying to any existing destination file. If  
10339 the response from the standard input is affirmative, the copy shall be attempted;



10340 otherwise, it shall not.

10341 **-L** Take actions based on the type and contents of the file referenced by any symbolic  
10342 link specified as a *source\_file* operand or any symbolic links encountered during  
10343 traversal of a file hierarchy.

### 10344 **Notes to Reviewers**

10345 *This section with side shading will not appear in the final copy. - Ed.*

10346 A description of the **-P** option is needed. This is not in the IEEE P1003.2b draft  
10347 standard.

10348 **-p** Duplicate the following characteristics of each source file in the corresponding  
10349 destination file:

10350 1. The time of last data modification and time of last access. If this duplication  
10351 fails for any reason, *cp* shall write a diagnostic message to standard error.

10352 2. The user ID and group ID. If this duplication fails for any reason, it is  
10353 unspecified whether *cp* writes a diagnostic message to standard error.

10354 3. The file permission bits and the S\_ISUID and S\_ISGID bits. Other,  
10355 implementation-defined, bits may be duplicated as well. If this duplication  
10356 fails for any reason, *cp* shall write a diagnostic message to standard error.

10357 If the user ID or the group ID cannot be duplicated, the file permission bits  
10358 S\_ISUID and S\_ISGID shall be cleared. If these bits are present in the source file but  
10359 are not duplicated in the destination file, it is unspecified whether *cp* writes a  
10360 diagnostic message to standard error.

10361 The order in which the preceding characteristics are duplicated is unspecified. The  
10362 *dest\_file* shall not be deleted if these characteristics cannot be preserved.

10363 **-R** Copy file hierarchies.

10364 OB **-r** Copy file hierarchies. The treatment of special files is implementation-defined.

10365 Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be  
10366 considered an error. The last option specified shall determine the behavior of the utility.

### 10367 **OPERANDS**

10368 The following operands shall be supported:

10369 *source\_file* A path name of a file to be copied.

10370 *target\_file* A path name of an existing or nonexistent file, used for the output when a single  
10371 file is copied.

10372 *target* A path name of a directory to contain the copied files.

### 10373 **STDIN**

10374 Used to read an input line in response to each prompt specified in the STDERR section.  
10375 Otherwise, the standard input shall not be used.

### 10376 **INPUT FILES**

10377 The input files specified as operands may be of any file type.

10378 **ENVIRONMENT VARIABLES**

10379 The following environment variables shall affect the execution of *cp*:

10380 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 10381 If *LANG* is unset or null, the corresponding value from the implementation-  
 10382 defined default locale shall be used. If any of the internationalization variables  
 10383 contains an invalid setting, the utility shall behave as if none of the variables had  
 10384 been defined.

10385 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 10386 internationalization variables.

10387 *LC\_COLLATE*  
 10388 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 10389 character collating elements used in the extended regular expression defined for  
 10390 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

10391 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 10392 characters (for example, single-byte as opposed to multi-byte characters in  
 10393 arguments and input files) and the behavior of character classes used in the  
 10394 extended regular expression defined for the **yesexpr** locale keyword in the  
 10395 *LC\_MESSAGES* category.

10396 *LC\_MESSAGES*  
 10397 Determine the locale for the processing of affirmative responses that should be  
 10398 used to affect the format and contents of diagnostic messages written to standard  
 10399 error.

10400 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

10401 **ASYNCHRONOUS EVENTS**

10402 Default.

10403 **STDOUT**

10404 Not used.

10405 **STDERR**

10406 A prompt shall be written to standard error under the conditions specified in the *DESCRIPTION*  
 10407 section. The prompt shall contain the destination path name, but its format is otherwise  
 10408 unspecified. Otherwise, the standard error shall be used only for diagnostic messages.

10409 **OUTPUT FILES**

10410 The output files may be of any type.

10411 **EXTENDED DESCRIPTION**

10412 None.

10413 **EXIT STATUS**

10414 The following exit values shall be returned:

10415 0 All files were copied successfully.

10416 >0 An error occurred.

10417 **CONSEQUENCES OF ERRORS**

10418 If *cp* is prematurely terminated by a signal or error, files or file hierarchies may be only partially  
 10419 copied and files and directories may have incorrect permissions or access and modification  
 10420 times.

10421 **APPLICATION USAGE**

10422 The difference between **-R** and **-r** is in the treatment by *cp* of file types other than regular and  
10423 directory. The original **-r** flag, for historic reasons, does not handle special files any differently  
10424 from regular files, but always reads the file and copies its contents. This has obvious problems in  
10425 the presence of special file types; for example, character devices, FIFOs, and sockets. The **-R**  
10426 option is intended to recreate the file hierarchy and the **-r** option supports historical practice. It  
10427 was anticipated that a future version of this volume of IEEE Std. 1003.1-200x would deprecate  
10428 the **-r** option, and for that reason, there has been no attempt to fix its behavior with respect to  
10429 FIFOs or other file types where copying the file is clearly wrong. However, some systems  
10430 support **-r** with the same abilities as the **-R** defined in this volume of IEEE Std. 1003.1-200x. To  
10431 accommodate them as well as systems that do not, the differences between **-r** and **-R** are  
10432 implementation-defined. Implementations may make them identical. The **-r** option is now  
10433 marked obsolescent.

10434 The set-user-ID and set-group-ID bits are explicitly cleared when files are created. This is to  
10435 prevent users from creating programs that are set-user-ID or set-group-ID to them when  
10436 copying files or to make set-user-ID or set-group-ID files accessible to new groups of users. For  
10437 example, if a file is set-user-ID and the copy has a different group ID than the source, a new  
10438 group of users has execute permission to a set-user-ID program than did previously. In  
10439 particular, this is a problem for superusers copying users' trees.

10440 **EXAMPLES**

10441 None.

10442 **RATIONALE**

10443 The **-i** option exists on BSD systems, giving applications and users a way to avoid accidentally  
10444 removing files when copying. Although the 4.3 BSD version does not prompt if the standard  
10445 input is not a terminal, the standard developers decided that use of **-i** is a request for interaction,  
10446 so when the destination path exists, the utility takes instructions from whatever responds on  
10447 standard input.

10448 The exact format of the interactive prompts is unspecified. Only the general nature of the  
10449 contents of prompts are specified because implementations may desire more descriptive  
10450 prompts than those used on historical implementations. Therefore, an application using the **-i**  
10451 option relies on the system to provide the most suitable dialog directly with the user, based on  
10452 the behavior specified.

10453 The **-p** option is historical practice on BSD systems, duplicating the time of last data  
10454 modification and time of last access. This volume of IEEE Std. 1003.1-200x extends it to preserve  
10455 the user and group IDs, as well as the file permissions. This requirement has obvious problems  
10456 in that the directories are almost certainly modified after being copied. This volume of  
10457 IEEE Std. 1003.1-200x requires that the modification times be preserved. The statement that the  
10458 order in which the characteristics are duplicated is unspecified is to permit implementations to  
10459 provide the maximum amount of security for the user. Implementations should take into  
10460 account the obvious security issues involved in setting the owner, group, and mode in the  
10461 wrong order or creating files with an owner, group, or mode different from the final value.

10462 It is unspecified whether *cp* writes diagnostic messages when the user and group IDs cannot be  
10463 set due to the widespread practice of users using **-p** to duplicate some portion of the file  
10464 characteristics, indifferent to the duplication of others. Historic implementations only write  
10465 diagnostic messages on errors other than [EPERM].

10466 The **-r** option is historical practice on BSD and BSD-derived systems, copying file hierarchies as  
10467 opposed to single files. This functionality is used heavily in historical applications, and its loss  
10468 would significantly decrease consensus. The **-R** option was added as a close synonym to the **-r**  
10469 option, selected for consistency with all other options in this volume of IEEE Std. 1003.1-200x

10470 that do recursive directory descent.

10471 When a failure occurs during the copying of a file hierarchy, *cp* is required to attempt to copy  
10472 files that are on the same level in the hierarchy or above the file where the failure occurred. It is  
10473 unspecified if *cp* shall attempt to copy files below the file where the failure occurred (which  
10474 cannot succeed in any case).

10475 Permissions, owners, and groups of created special file types have been deliberately left as  
10476 implementation-defined. This is to allow systems to satisfy special requirements (for example,  
10477 allowing users to create character special devices, but requiring them to be owned by a certain  
10478 group). In general, it is strongly suggested that the permissions, owner, and group be the same  
10479 as if the user had run the historical *mknod*, *ln*, or other utility to create the file. It is also probable  
10480 that additional privileges are required to create block, character, or other implementation-  
10481 defined special file types.

10482 Additionally, the *-p* option explicitly requires that all set-user-ID and set-group-ID permissions  
10483 be discarded if any of the owner or group IDs cannot be set. This is to keep users from  
10484 unintentionally giving away special privilege when copying programs.

10485 When creating regular files, historical versions of *cp* use the mode of the source file as modified  
10486 by the file mode creation mask. Other choices would have been to use the mode of the source file  
10487 unmodified by the creation mask or to use the same mode as would be given to a new file  
10488 created by the user (plus the execution bits of the source file) and then modify it by the file mode  
10489 creation mask. In the absence of any strong reason to change historic practice, it was in large part  
10490 retained.

10491 When creating directories, historical versions of *cp* use the mode of the source directory, plus  
10492 read, write, and search bits for the owner, as modified by the file mode creation mask. This is  
10493 done so that *cp* can copy trees where the user has read permission, but the owner does not. A  
10494 side effect is that if the file creation mask denies the owner permissions, *cp* fails. Also, once the  
10495 copy is done, historical versions of *cp* set the permissions on the created directory to be the same  
10496 as the source directory, unmodified by the file creation mask.

10497 This behavior has been modified so that *cp* is always able to create the contents of the directory,  
10498 regardless of the file creation mask. After the copy is done, the permissions are set to be the same  
10499 as the source directory, as modified by the file creation mask. This latter change from historical  
10500 behavior is to prevent users from accidentally creating directories with permissions beyond  
10501 those they would normally set and for consistency with the behavior of *cp* in creating files.

10502 It is not a requirement that *cp* detect attempts to copy a file to itself; however, implementations  
10503 are strongly encouraged to do so. Historical implementations have detected the attempt in most  
10504 cases.

10505 There are two methods of copying subtrees in this volume of IEEE Std. 1003.1-200x. The other  
10506 method is described as part of the *pax* utility (see *pax* (on page 2910)). Both methods are  
10507 historical practice. The *cp* utility provides a simpler, more intuitive interface, while *pax* offers a  
10508 finer granularity of control. Each provides additional functionality to the other; in particular, *pax*  
10509 maintains the hard-link structure of the hierarchy, while *cp* does not. It is the intention of the  
10510 standard developers that the results be similar (using appropriate option combinations in both  
10511 utilities). The results are not required to be identical; there seemed insufficient gain to  
10512 applications to balance the difficulty of implementations having to guarantee that the results  
10513 would be exactly identical.

10514 The wording allowing *cp* to copy a directory to implementation-defined file types not specified  
10515 by the System Interfaces volume of IEEE Std. 1003.1-200x is provided so that implementations  
10516 supporting symbolic links are not required to prohibit copying directories to symbolic links.  
10517 Other extensions to the System Interfaces volume of IEEE Std. 1003.1-200x file types may need to

10518 use this loophole as well.

10519 **FUTURE DIRECTIONS**

10520 The `-r` option may be removed; use `-R` instead.

10521 **SEE ALSO**

10522 *mv, find, ln, pax*

10523 **CHANGE HISTORY**

10524 First released in Issue 2.

10525 **Issue 4**

10526 Aligned with the ISO/IEC 9945-2:1993 standard.

10527 **Issue 6**

10528 The `-r` option is marked obsolescent.

10529 The new options `-H`, `-L`, and `-P` are added to align with the IEEE P1003.2b draft standard. These  
10530 options affect the processing of symbolic links.

## 10531 NAME

10532 crontab — schedule periodic background work

## 10533 SYNOPSIS

10534 UP crontab [*file*]

10535 crontab [ *-e* | *-l* | *-r* ]

10536

## 10537 DESCRIPTION

10538 The *crontab* utility shall create, replace, or edit a user's *crontab* entry; a *crontab* entry is a list of  
 10539 commands and the times at which they shall be executed. The new *crontab* entry can be input by  
 10540 specifying *file* or input from standard input if no *file* operand is specified, or by using an editor, if  
 10541 *-e* is specified.

10542 Upon execution of a command from a *crontab* entry, the implementation shall supply a default  
 10543 environment, defining at least the following environment variables:

10544 *HOME* A path name of the user's home directory.

10545 *LOGNAME* The user's login name.

10546 *PATH* A string representing a search path guaranteed to find all of the standard utilities.

10547 *SHELL* A path name of the command interpreter. When *crontab* is invoked as specified by  
 10548 this volume of IEEE Std. 1003.1-200x, the value shall be a path name for *sh*.

10549 The values of these variables when *crontab* is invoked as specified by this volume of  
 10550 IEEE Std. 1003.1-200x shall not affect the default values provided when the scheduled command  
 10551 is run.

10552 If standard output and standard error are not redirected by commands executed from the  
 10553 *crontab* entry, any generated output or errors shall be mailed, via an implementation-defined  
 10554 method, to the user.

10555 XSI Users are permitted to use *crontab* if their names appear in the file */usr/lib/cron/cron.allow*. If  
 10556 that file does not exist, the file */usr/lib/cron/cron.deny* is checked to determine whether the user  
 10557 should be denied access to *crontab*. If neither file exists, only a process with appropriate  
 10558 privileges is allowed to submit a job. If only *cron.deny* exists and is empty, global usage is  
 10559 permitted. The *cron.allow* and *cron.deny* files consist of one user name per line.

## 10560 OPTIONS

10561 The *crontab* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 10562 12.2, Utility Syntax Guidelines.

10563 The following options shall be supported:

10564 *-e* Edit a copy of the invoking user's *crontab* entry, or create an empty entry to edit if  
 10565 the *crontab* entry does not exist. When editing is complete, the entry shall be  
 10566 installed as the user's *crontab* entry.

10567 *-l* (The letter ell.) List the invoking user's *crontab* entry.

10568 *-r* Remove the invoking user's *crontab* entry.

## 10569 OPERANDS

10570 The following operand shall be supported:

10571 *file* The path name of a file that contains specifications, in the format defined in the  
 10572 INPUT FILES section, for *crontab* entries.

10573 **STDIN**

10574 See the INPUT FILES section.

10575 **INPUT FILES**

10576 In the POSIX locale, the user or application shall ensure that a crontab entry is a text file  
 10577 consisting of lines of six fields each. The fields shall be separated by <blank> characters. The first  
 10578 five fields shall be integer patterns that specify the following:

- 10579 1. Minute (0-59)
- 10580 2. Hour (0-23)
- 10581 3. Day of the month (1-31)
- 10582 4. Month of the year (1-12)
- 10583 5. Day of the week (0-6 with 0=Sunday)

10584 Each of these patterns can be either an asterisk (meaning all valid values), an element, or a list of  
 10585 elements separated by commas. An element shall be either a number or two numbers separated  
 10586 by a hyphen (meaning an inclusive range). The specification of days can be made by two fields  
 10587 (day of the month and day of the week). If month, day of month, and day of week are all  
 10588 asterisks, every day shall be matched. If either the month or day of month is specified as an  
 10589 element or list, but the day of week is an asterisk, the month and day of month fields shall  
 10590 specify the days that match. If both month and day of month are specified as asterisk, but day of  
 10591 week is an element or list, then only the specified days of the week match. Finally, if either the  
 10592 month or day of month is specified as an element or list, and the day of week is also specified as  
 10593 an element or list, then any day matching either the month and day of month, or the day of  
 10594 week, shall be matched.

10595 The sixth field of a line in a crontab entry is a string that shall be executed by *sh* at the specified  
 10596 times. A percent sign character in this field shall be translated to a <newline> character. Any  
 10597 character preceded by a backslash (including the '%' ) shall cause that character to be treated  
 10598 literally. Only the first line (up to a '%' or end-of-line) of the command field shall be executed  
 10599 by the command interpreter. The other lines shall be made available to the command as  
 10600 standard input.

10601 Blank lines and those whose first non-<blank> character is '#' shall be ignored.

10602 XSI The text files `/usr/lib/cron/cron.allow` and `/usr/lib/cron/cron.deny` contain user names, one per  
 10603 line, of users who are, respectively, authorized or denied access to the service underlying the  
 10604 *crontab* utility.

10605 **ENVIRONMENT VARIABLES**

10606 The following environment variables shall affect the execution of *crontab*:

- |                                           |                 |                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10607<br>10608                            | <i>EDITOR</i>   | Determine the editor to be invoked when the <code>-e</code> option is specified. The default editor shall be <i>vi</i> .                                                                                                                                                                                                                                       |
| 10609<br>10610<br>10611<br>10612<br>10613 | <i>LANG</i>     | Provide a default value for the internationalization variables that are unset or null. If <i>LANG</i> is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined. |
| 10614<br>10615                            | <i>LC_ALL</i>   | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                       |
| 10616<br>10617                            | <i>LC_CTYPE</i> | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in                                                                                                                                                                                                  |

- 10618 arguments and input files).
- 10619 **LC\_MESSAGES**
- 10620 Determine the locale that should be used to affect the format and contents of
- 10621 diagnostic messages written to standard error.
- 10622 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 10623 **ASYNCHRONOUS EVENTS**
- 10624 Default.
- 10625 **STDOUT**
- 10626 If the `-l` option is specified, the crontab entry shall be written to the standard output.
- 10627 **STDERR**
- 10628 Used only for diagnostic messages.
- 10629 **OUTPUT FILES**
- 10630 None.
- 10631 **EXTENDED DESCRIPTION**
- 10632 None.
- 10633 **EXIT STATUS**
- 10634 The following exit values shall be returned:
- 10635 0 Successful completion.
- 10636 >0 An error occurred.
- 10637 **CONSEQUENCES OF ERRORS**
- 10638 The user's crontab entry is not submitted, removed, edited, or listed.
- 10639 **APPLICATION USAGE**
- 10640 The format of the *crontab* entry shown here is guaranteed only for the POSIX locale. Other
- 10641 cultures may be supported with substantially different interfaces, although implementations are
- 10642 encouraged to provide comparable levels of functionality.
- 10643 The default settings of the *HOME*, *LOGNAME*, *PATH*, and *SHELL* variables that are given to the
- 10644 scheduled job are not affected by the settings of those variables when *crontab* is run; as stated,
- 10645 they are defaults. The text about "invoked as specified by this volume of IEEE Std. 1003.1-200x"
- 10646 means that the implementation may provide extensions that allow these variables to be affected
- 10647 at runtime, but that the user has to take explicit action in order to access the extension, such as
- 10648 give a new option flag or modify the format of the crontab entry.
- 10649 A typical user error is to type only *crontab*; this causes the system to wait for the new crontab
- 10650 entry on standard input. If end-of-file is typed (generally `<control>-D`), the crontab entry is
- 10651 replaced by an empty file. In this case, the user should type the interrupt character, which
- 10652 prevents the crontab entry from being replaced.
- 10653 **EXAMPLES**
- 10654 1. Clean up **core** files every weekday morning at 3:15 am:
- 10655 15 3 \* \* 1-5 find \$HOME -name core 2>/dev/null | xargs rm -f
- 10656 2. Mail a birthday greeting:
- 10657 0 12 14 2 \* mailx john%Happy Birthday!%Time for lunch.
- 10658 3. As an example of specifying the two types of days:



10659           0 0 1,15 \* 1

10660           would run a command on the first and fifteenth of each month, as well as on every  
10661           Monday. To specify days by only one field, the other field should be set to '\*'; for  
10662           example:

10663           0 0 \* \* 1

10664           would run a command only on Mondays.

10665 **RATIONALE**

10666           All references to a *cron* daemon and to *cron files* have been omitted. Although historical  
10667           implementations have used this arrangement, there is no reason to limit future implementations.

10668           This description of *crontab* is designed to support only users with normal privileges. The format  
10669           of the input is based on the System V *crontab*; however, there is no requirement here that the  
10670           actual system database used by the *cron* daemon (or a similar mechanism) use this format  
10671           internally. For example, systems derived from BSD are likely to have an additional field  
10672           appended that indicates the user identity to be used when the job is submitted.

10673           The *-e* option was adopted from the SVID as a user convenience, although it does not exist in all  
10674           historical implementations.

10675 **FUTURE DIRECTIONS**

10676           None.

10677 **SEE ALSO**

10678           *at*

10679 **CHANGE HISTORY**

10680           First released in Issue 2.

10681 **Issue 4**

10682           Aligned with the ISO/IEC 9945-2:1993 standard.

10683 **Issue 6**

10684           This utility is now marked as part of the User Portability Utilities option.

10685           The normative text is reworded to avoid use of the term “must” for application requirements.

## 10686 NAME

10687 csplit — split files based on context

## 10688 SYNOPSIS

10689 UP csplit [-ks][-f *prefix*][-n *number*] *file arg1 ... argn*

10690

## 10691 DESCRIPTION

10692 The *csplit* utility shall read the file named by the *file* operand, write all or part of that file into  
10693 other files as directed by the *arg* operands, and write the sizes of the files.

## 10694 OPTIONS

10695 The *csplit* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
10696 12.2, Utility Syntax Guidelines.

10697 The following options shall be supported:

10698 **-f *prefix*** Name the created files *prefix00*, *prefix01*, ..., *prefixn*. The default is **xx00 ... xxn**. If  
10699 the *prefix* argument would create a file name exceeding {NAME\_MAX} bytes, an  
10700 error shall result, *csplit* shall exit with a diagnostic message and no files shall be  
10701 created.10702 **-k** Leave previously created files intact. By default, *csplit* shall remove created files if  
10703 an error occurs.10704 **-n *number*** Use *number* decimal digits to form file names for the file pieces. The default shall be  
10705 2.10706 **-s** Suppress the output of file size messages.

## 10707 OPERANDS

10708 The following operands shall be supported:

10709 ***file*** The path name of a text file to be split. If *file* is '-', the standard input shall be  
10710 used.10711 The operands *arg1 ... argn* can be a combination of the following:10712 ***/rexp/[offset]***10713 A file shall be created using the content of the lines from the current line up to, but  
10714 not including, the line that results from the evaluation of the regular expression  
10715 with *offset*, if any, applied. The regular expression *rexp* shall follow the rules for  
10716 basic regular expressions described in the Base Definitions volume of  
10717 IEEE Std. 1003.1-200x, Section 9.3, Basic Regular Expressions. The application shall  
10718 use the sequence "\/" to specify a slash character within the *rexp*. The optional  
10719 offset shall be a positive or negative integer value representing a number of lines.  
10720 A positive integer value can be preceded by '+'. If the selection of lines from an  
10721 *offset* expression of this type would create a file with zero lines, or one with greater  
10722 than the number of lines left in the input file, the results are unspecified. After the  
10723 section is created, the current line shall be set to the line that results from the  
10724 evaluation of the regular expression with any offset applied. If the current line is  
10725 the first line in the file and a regular expression operation has not yet been  
10726 performed, the pattern match of *rexp* shall be applied from the current line to the  
10727 end of the file. Otherwise, the pattern match of *rexp* shall be applied from the line  
10728 following the current line to the end of the file.10729 ***%rexp%[offset]***10730 Equivalent to */rexp/[offset]*, except that no file shall be created for the selected  
10731 section of the input file. The application shall use the sequence "\%" to specify a

- 10732 percent-sign character within the *rexp*.
- 10733 *line\_no* Create a file from the current line up to (but not including) the line number *line\_no*.  
 10734 Lines in the file shall be numbered starting at one. The current line becomes  
 10735 *line\_no*.
- 10736 *{num}* Repeat operand. This operand can follow any of the operands described  
 10737 previously. If it follows a *rexp* type operand, that operand shall be applied *num*  
 10738 more times. If it follows a *line\_no* operand, the file shall be split every *line\_no* lines,  
 10739 *num* times, from that point.
- 10740 An error shall be reported if an operand does not reference a line between the current position  
 10741 and the end of the file.
- 10742 **STDIN**
- 10743 See the INPUT FILES section.
- 10744 **INPUT FILES**
- 10745 The input file shall be a text file.
- 10746 **ENVIRONMENT VARIABLES**
- 10747 The following environment variables shall affect the execution of *csplit*:
- 10748 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 10749 If *LANG* is unset or null, the corresponding value from the implementation-  
 10750 defined default locale shall be used. If any of the internationalization variables  
 10751 contains an invalid setting, the utility shall behave as if none of the variables had  
 10752 been defined.
- 10753 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 10754 internationalization variables.
- 10755 *LC\_COLLATE*  
 10756 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 10757 character collating elements within regular expressions.
- 10758 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 10759 characters (for example, single-byte as opposed to multi-byte characters in  
 10760 arguments and input files) and the behavior of character classes within regular  
 10761 expressions.
- 10762 *LC\_MESSAGES*  
 10763 Determine the locale that should be used to affect the format and contents of  
 10764 diagnostic messages written to standard error.
- 10765 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 10766 **ASYNCHRONOUS EVENTS**
- 10767 If the *-k* option is specified, created files shall be retained. Otherwise, the default action occurs.
- 10768 **STDOUT**
- 10769 Unless the *-s* option is used, the standard output shall consist of one line per file created, with a  
 10770 format as follows:  
 10771 "%d\n", <file size in bytes>

10772 **STDERR**

10773 Used only for diagnostic messages.

10774 **OUTPUT FILES**

10775 The output files shall contain portions of the original input file; otherwise, unchanged.

10776 **EXTENDED DESCRIPTION**

10777 None.

10778 **EXIT STATUS**

10779 The following exit values shall be returned:

10780 0 Successful completion.

10781 >0 An error occurred.

10782 **CONSEQUENCES OF ERRORS**

10783 By default, created files shall be removed if an error occurs. When the **-k** option is specified,

10784 created files shall not be removed if an error occurs.

10785 **APPLICATION USAGE**

10786 None.

10787 **EXAMPLES**

10788 1. This example creates four files, **cobol00 ... cobol03**:

10789 `csplit -f cobol file '/procedure division/' /par5./ /par16./`

10790 After editing the split files, they can be recombined as follows:

10791 `cat cobol0[0-3] > file`

10792 Note that this example overwrites the original file.

10793 2. This example would split the file after the first 99 lines, and every 100 lines thereafter, up  
10794 to 9999 lines; this is because lines in the file are numbered from 1 rather than zero, for  
10795 historical reasons:

10796 `csplit -k file 100 {99}`

10797 3. Assuming that **prog.c** follows the C-language coding convention of ending routines with a  
10798 `'}'` at the beginning of the line, this example creates a file containing each separate C  
10799 routine (up to 21) in **prog.c**:

10800 `csplit -k prog.c '%main(%' '/^}/+1' {20}`

10801 **RATIONALE**

10802 The **-n** option was added to extend the range of file names that could be handled.

10803 Consideration was given to adding a **-a** flag to use the alphabetic file name generation used by  
10804 the historical *split* utility, but the functionality added by the **-n** option was deemed to make  
10805 alphabetic naming unnecessary.

10806 **FUTURE DIRECTIONS**

10807 None.

10808 **SEE ALSO**

10809 *sed*, *split*

10810 **CHANGE HISTORY**

10811 First released in Issue 2.

10812 **Issue 4**

10813 Aligned with the ISO/IEC 9945-2: 1993 standard.

10814 **Issue 5**

10815 FUTURE DIRECTIONS section added.

10816 **Issue 6**

10817 This utility is now marked as part of the User Portability Utilities option.

10818 The APPLICATION USAGE section is added.

10819 The description of regular expression operands is changed to align with the IEEE P1003.2b draft standard.

10820

10821 The normative text is reworded to avoid use of the term “must” for application requirements.

## 10822 NAME

10823 ctags — create a tags file (DEVELOPMENT, FORTRAN)

## 10824 SYNOPSIS

10825 UP `ctags [-a][-f tagsfile] pathname ...`

10826 `ctags -x pathname ...`

10827

## 10828 DESCRIPTION

10829 The *ctags* utility shall be provided on systems that support the User Portability Utilities option,  
10830 the Software Development Utilities option, and either or both of the C-Language Development  
10831 Utilities option and FORTRAN Development Utilities option. On other systems, it is optional.

10832 The *ctags* utility shall write a *tags* file or an index of objects from C-language or FORTRAN  
10833 source files specified by the *pathname* operands. The tags file shall list the locators of language-  
10834 specific objects within the source files. A locator consists of a name, path name, and either a  
10835 search pattern or a line number that can be used in searching for the object definition. The  
10836 objects that shall be recognized are specified in the EXTENDED DESCRIPTION section.

## 10837 OPTIONS

10838 The *ctags* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
10839 12.2, Utility Syntax Guidelines.

10840 The following options shall be supported:

10841 **-a** Append to tags file.

10842 **-f *tagsfile*** Write the object locator lists into *tagsfile* instead of the default file named **tags** in  
10843 the current directory.

10844 **-x** Produce a list of object names, the line number, and file name in which each is  
10845 defined, as well as the text of that line, and write this to the standard output. A  
10846 **tags** file shall not be created when **-x** is specified.

## 10847 OPERANDS

10848 The following *pathname* operands are supported:

10849 ***file.c*** Files with basenames ending with the **.c** suffix shall be treated as C-language  
10850 source code. Such files that are not valid input to *c99* produce unspecified results.

10851 ***file.h*** Files with basenames ending with the **.h** suffix shall be treated as C-language  
10852 source code. Such files that are not valid input to *c99* produce unspecified results.

10853 ***file.f*** Files with basenames ending with the **.f** suffix shall be treated as FORTRAN-  
10854 language source code. Such files that are not valid input to *fort77* produce  
10855 unspecified results.

10856 The handling of other files is implementation-defined.

## 10857 STDIN

10858 See the INPUT FILES section.

## 10859 INPUT FILES

10860 The input files shall be text files containing source code in the language indicated by the operand  
10861 file name suffixes.

10862 **ENVIRONMENT VARIABLES**

10863 The following environment variables shall affect the execution of *ctags*:

10864 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 10865 If *LANG* is unset or null, the corresponding value from the implementation-  
 10866 defined default locale shall be used. If any of the internationalization variables  
 10867 contains an invalid setting, the utility shall behave as if none of the variables had  
 10868 been defined.

10869 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 10870 internationalization variables.

10871 *LC\_COLLATE*  
 10872 Determine the order in which output is sorted for the *-x* option. The POSIX locale  
 10873 determines the order in which the tags file is written.

10874 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 10875 characters (for example, single-byte as opposed to multi-byte characters in  
 10876 arguments and input files). When processing C-language source code, if the locale  
 10877 is not compatible with the C locale described by the ISO C standard, the results are  
 10878 unspecified.

10879 *LC\_MESSAGES*  
 10880 Determine the locale that should be used to affect the format and contents of  
 10881 diagnostic messages written to standard error.

10882 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

10883 **ASYNCHRONOUS EVENTS**

10884 Default.

10885 **STDOUT**

10886 The list of object name information produced by the *-x* option shall be written to standard  
 10887 output in the following format:

```
10888 "%s %d %s %s", <object-name>, <line-number>, <filename>,
10889 <text>
```

10890 where *<text>* is the text of line *<line-number>* of file *<filename>*.

10891 **STDERR**

10892 Used only for diagnostic messages.

10893 **OUTPUT FILES**

10894 When the *-x* option is not specified, the format of the output file shall be:

```
10895 "%s\t%s\t\t/%s/\n", <identifier>, <filename>, <pattern>
```

10896 where *<pattern>* is a search pattern that could be used by an editor to find the defining instance  
 10897 of *<identifier>* in *<filename>* (where *defining instance* is indicated by the declarations listed in the  
 10898 EXTENDED DESCRIPTION).

10899 An optional circumflex (*'^'*) can be added as a prefix to *<pattern>*, and an optional dollar sign  
 10900 can be appended to *<pattern>* to indicate that the pattern is anchored to the beginning (end) of a  
 10901 line of text. Any slash or backslash characters in *<pattern>* shall be preceded by a backslash  
 10902 character. The anchoring circumflex, dollar sign, and escaping backslash characters shall not be  
 10903 considered part of the search pattern. All other characters in the search pattern shall be  
 10904 considered literal characters.

10905 An alternative format is:

10906 "%s\t%s\t?%s?\n", <identifier>, <filename>, <pattern>

10907 which is identical to the first format except that slashes in <pattern> shall not be preceded by  
10908 escaping backslash characters, and question mark characters in <pattern> shall be preceded by  
10909 backslash characters.

10910 A second alternative format is:

10911 "%s\t%s\t%d\n", <identifier>, <filename>, <lineno>

10912 where <lineno> is a decimal line number that could be used by an editor to find <identifier> in  
10913 <filename>.

10914 Neither alternative format shall be produced by *ctags* when it is used as described by  
10915 IEEE Std. 1003.1-200x, but the standard utilities that process tags files shall be able to process  
10916 those formats as well as the first format.

10917 In any of these formats, the file shall be sorted by identifier, based on the collation sequence in  
10918 the POSIX locale.

#### 10919 EXTENDED DESCRIPTION

10920 If the operand identifies C-language source, the *ctags* utility shall attempt to produce an output  
10921 line for each of the following objects:

- 10922 • Function definitions
- 10923 • Type definitions
- 10924 • Macros with arguments

10925 It may also produce output for any of the following objects:

- 10926 • Function prototypes
- 10927 • Structures
- 10928 • Unions
- 10929 • Global variable definitions
- 10930 • Enumeration types
- 10931 • Macros without arguments
- 10932 • **#define** statements
- 10933 • **#line** statements

10934 Any **#if** and **#ifdef** statements shall produce no output. The tag **main** is treated specially in C  
10935 programs. The tag formed shall be created by prefixing **M** to the name of the file, with the  
10936 trailing **.c**, and leading path name components (if any) removed.

10937 On systems that do not support the C-Language Development Utilities option, *ctags* produces  
10938 undefined results for C-language source code files.

10939 If the operand identifies FORTRAN source, the *ctags* utility shall produce an output line for each  
10940 function definition. It may also produce output for any of the following objects:

- 10941 • Subroutine definitions
- 10942 • COMMON statements



- 10943       • PARAMETER statements
  - 10944       • DATA and BLOCK DATA statements
  - 10945       • Statement numbers
- 10946       On systems that do not support the FORTRAN Development Utilities option, *ctags* produces  
 10947       unspecified results for FORTRAN source code files. It should write to standard error a message  
 10948       identifying this condition and cause a non-zero exit status to be produced.
- 10949       It is implementation-defined what other objects (including duplicate identifiers) produce output.
- 10950 **EXIT STATUS**
- 10951       The following exit values shall be returned:
- 10952       0   Successful completion.
  - 10953       >0  An error occurred.
- 10954 **CONSEQUENCES OF ERRORS**
- 10955       Default.
- 10956 **APPLICATION USAGE**
- 10957       The output with *-x* is meant to be a simple index that can be written out as an off-line readable  
 10958       function index. If the input files to *ctags* (such as *.c* files) were not created using the same locales  
 10959       as those in effect when *ctags -x* is run, results might not be as expected.
- 10960       The description of C-language processing says “attempts to” because the C language can be  
 10961       greatly confused, especially through the use of *#defines*, and this utility would be of no use if  
 10962       the real C preprocessor were run to identify them. The output from *ctags* may be fooled and  
 10963       incorrect for various constructs.
- 10964 **EXAMPLES**
- 10965       None.
- 10966 **RATIONALE**
- 10967       The option list was significantly reduced from that provided by historical implementations. The  
 10968       *-F* option was omitted as redundant, since it is the default. The *-B* option was omitted as being  
 10969       of very limited usefulness. The *-t* option was omitted since the recognition of typedefs is now  
 10970       required for C source files. The *-u* option was omitted because the update function was judged  
 10971       to be not only inefficient, but also rarely needed.
- 10972       An early proposal included a *-w* option to suppress warning diagnostics. Since the types of such  
 10973       diagnostics could not be described, the option was omitted as being not useful.
- 10974       The text for *LC\_CTYPE* about compatibility with the C locale acknowledges that the ISO C  
 10975       standard imposes requirements on the locale used to process C source. This could easily be a  
 10976       superset of that known as “the C locale” by way of implementation extensions, or one of a few  
 10977       alternative locales for systems supporting different codesets. No statement is made for  
 10978       FORTRAN because the ANSI X3.9-1978 standard (FORTRAN 77) does not (yet) define a similar  
 10979       locale concept. However, a general rule in this volume of IEEE Std. 1003.1-200x is that any time  
 10980       that locales do not match (preparing a file for one locale and processing it in another), the results  
 10981       are suspect.
- 10982       The collation sequence of the tags file is not affected by *LC\_COLLATE* because it is typically not  
 10983       used by human readers, but only by programs such as *vi* to locate the tag within the source files.  
 10984       Using the POSIX locale eliminates some of the problems of coordinating locales between the  
 10985       *ctags* file creator and the *vi* file reader.

- 10986 Historically, the tags file has been used only by *ex* and *vi*. However, the format of the tags file  
 10987 has been published to encourage other programs to use the tags in new ways. The format allows  
 10988 either patterns or line numbers to find the identifiers because the historical *vi* recognizes either.  
 10989 The *ctags* utility does not produce the format using line numbers because it is not useful  
 10990 following any source file changes that add or delete lines. The documented search patterns  
 10991 match historical practice. It should be noted that literal leading circumflex or trailing dollar-sign  
 10992 characters in the search pattern will only behave correctly if anchored to the beginning of the  
 10993 line or end of the line by an additional circumflex or dollar-sign character.
- 10994 Historical implementations also understand the objects used by the languages Pascal and  
 10995 sometimes LISP, and they understand the C source output by *lex* and *yacc*. The *ctags* utility is  
 10996 not required to accommodate these languages, although implementors are encouraged to do so.
- 10997 The following historical option was not specified, as *vgrind* is not included in this volume of  
 10998 IEEE Std. 1003.1-200x:
- 10999        -v            If the -v flag is given, an index of the form expected by *vgrind* is produced on the  
 11000                      standard output. This listing contains the function name, file name, and page  
 11001                      number (assuming 64-line pages). Since the output is sorted into lexicographic  
 11002                      order, it may be desired to run the output through *sort -f*. Sample use:
- 11003                      ctags -v files | sort -f > index vgrind -x index
- 11004        The special treatment of the tag **main** makes the use of *ctags* practical in directories with more  
 11005        than one program.
- 11006 **FUTURE DIRECTIONS**
- 11007        None.
- 11008 **SEE ALSO**
- 11009        *c99*, *fort77*, *vi*
- 11010 **CHANGE HISTORY**
- 11011        First released in Issue 4.
- 11012 **Issue 5**
- 11013        FUTURE DIRECTIONS section added.
- 11014 **Issue 6**
- 11015        This utility is now marked as part of the User Portability Utilities option.
- 11016        The OUTPUT FILES section is changed to align with the IEEE P1003.2b draft standard.
- 11017        The normative text is reworded to avoid use of the term “must” for application requirements.
- 11018        IEEE PASC Interpretation 1003.2 #168 is applied, changing “create” to “write” in the  
 11019        DESCRIPTION.

## 11020 NAME

11021 cut — cut out selected fields of each line of a file

## 11022 SYNOPSIS

11023 cut -b *list* [-n] [*file* ...]

11024 cut -c *list* [*file* ...]

11025 cut -f *list* [-d *delim*][-s][*file* ...]

## 11026 DESCRIPTION

11027 The *cut* utility shall cut out bytes (-b option), characters (-c option) or character-delimited fields  
 11028 (-f option) from each line in one or more files, concatenate them, and write them to standard  
 11029 output.

## 11030 OPTIONS

11031 The *cut* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 11032 12.2, Utility Syntax Guidelines.

11033 The application shall ensure that the option-argument *list* (see options -b, -c, and -f below) is a  
 11034 comma-separated list or <blank> character-separated list of positive numbers and ranges.  
 11035 Ranges can be in three forms. The first is two positive numbers separated by a hyphen  
 11036 (*low-high*), which represents all fields from the first number to the second number. The second is  
 11037 a positive number preceded by a hyphen (-*high*), which represents all fields from field number 1  
 11038 to that number. The third is a positive number followed by a hyphen (*low-*), which represents  
 11039 that number to the last field, inclusive. The elements in *list* can be repeated, can overlap, and can  
 11040 be specified in any order, but the bytes, characters, or fields selected shall be written in the order  
 11041 of the input data. If an element appears in the selection list more than once, it shall be written  
 11042 exactly once.

11043 The following options shall be supported:

11044 -b *list* Cut based on a *list* of bytes. Each selected byte shall be output unless the -n option  
 11045 is also specified. It shall not be an error to select bytes not present in the input line.

11046 -c *list* Cut based on a *list* of characters. Each selected character shall be output. It shall  
 11047 not be an error to select characters not present in the input line.

11048 -d *delim* Set the field delimiter to the character *delim*. The default is the <tab> character.

11049 -f *list* Cut based on a *list* of fields, assumed to be separated in the file by a delimiter  
 11050 character (see -d). Each selected field shall be output. Output fields shall be  
 11051 separated by a single occurrence of the field delimiter character. Lines with no field  
 11052 delimiters shall be passed through intact, unless -s is specified. It shall not be an  
 11053 error to select fields not present in the input line.

11054 -n Do not split characters. When specified with the -b option, each element in *list* of  
 11055 the form *low-high* (hyphen-separated numbers) shall be modified as follows:

- 11056 • If the byte selected by *low* is not the first byte of a character, *low* shall be  
 11057 decremented to select the first byte of the character originally selected by *low*.  
 11058 If the byte selected by *high* is not the last byte of a character, *high* shall be  
 11059 decremented to select the last byte of the character prior to the character  
 11060 originally selected by *high*, or zero if there is no prior character. If the resulting  
 11061 range element has *high* equal to zero or *low* greater than *high*, the list element  
 11062 shall be dropped from *list* for that input line without causing an error.

11063 Each element in *list* of the form *low-* shall be treated as above with *high* set to the  
 11064 number of bytes in the current line, not including the terminating <newline>

11065 character. Each element in *list* of the form *-high* shall be treated as above with *low*  
 11066 set to 1. Each element in *list* of the form *num* (a single number) shall be treated as  
 11067 above with *low* set to *num* and *high* set to *num*.

11068 **-s** Suppress lines with no delimiter characters, when used with the **-f** option. Unless  
 11069 specified, lines with no delimiters shall be passed through untouched.

#### 11070 OPERANDS

11071 The following operand shall be supported:

11072 **file** A path name of an input file. If no **file** operands are specified, or if a **file** operand is  
 11073 **'-'**, the standard input shall be used.

#### 11074 STDIN

11075 The standard input shall be used only if no **file** operands are specified, or if a **file** operand is **'-'**.  
 11076 See the INPUT FILES section.

#### 11077 INPUT FILES

11078 The input files shall be text files, except that line lengths shall be unlimited.

#### 11079 ENVIRONMENT VARIABLES

11080 The following environment variables shall affect the execution of *cut*:

11081 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 11082 If **LANG** is unset or null, the corresponding value from the implementation-  
 11083 defined default locale shall be used. If any of the internationalization variables  
 11084 contains an invalid setting, the utility shall behave as if none of the variables had  
 11085 been defined.

11086 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 11087 internationalization variables.

11088 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 11089 characters (for example, single-byte as opposed to multi-byte characters in  
 11090 arguments and input files).

#### 11091 LC\_MESSAGES

11092 Determine the locale that should be used to affect the format and contents of  
 11093 diagnostic messages written to standard error.

11094 **XSI** **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

#### 11095 ASYNCHRONOUS EVENTS

11096 Default.

#### 11097 STDOUT

11098 The *cut* utility output shall be a concatenation of the selected bytes, characters, or fields (one of  
 11099 the following):

11100 "%s\n", <concatenation of bytes>

11101 "%s\n", <concatenation of characters>

11102 "%s\n", <concatenation of fields and field delimiters>

#### 11103 STDERR

11104 Used only for diagnostic messages.

11105 **OUTPUT FILES**

11106 None.

11107 **EXTENDED DESCRIPTION**

11108 None.

11109 **EXIT STATUS**

11110 The following exit values shall be returned:

11111 0 All input files were output successfully.

11112 &gt;0 An error occurred.

11113 **CONSEQUENCES OF ERRORS**

11114 Default.

11115 **APPLICATION USAGE**

11116 Earlier versions of the *cut* utility worked in an environment where bytes and characters were  
 11117 considered equivalent (modulo <backspace> and <tab> character processing in some  
 11118 implementations). In the extended world of multi-byte characters, the new **-b** option has been  
 11119 added. The **-n** option (used with **-b**) allows it to be used to act on bytes rounded to character  
 11120 boundaries. The algorithm specified for **-n** guarantees that:

11121 `cut -b 1-500 -n file > file1`11122 `cut -b 501- -n file > file2`

11123 ends up with all the characters in **file** appearing exactly once in **file1** or **file2**. (There is,  
 11124 however, a <newline> character in both **file1** and **file2** for each <newline> character in **file**.)

11125 **EXAMPLES**

11126 Examples of the option qualifier list:

11127 1,4,7 Select the first, fourth, and seventh bytes, characters, or fields and field delimiters.

11128 1-3,8 Equivalent to 1,2,3,8.

11129 -5,10 Equivalent to 1,2,3,4,5,10.

11130 3- Equivalent to third to last, inclusive.

11131 The *low-high* forms are not always equivalent when used with **-b** and **-n** and multi-byte  
 11132 characters; see the description of **-n**.

11133 The following command:

11134 `cut -d : -f 1,6 /etc/passwd`

11135 reads the System V password file (user database) and produces lines of the form:

11136 `<user ID>:<home directory>`

11137 Most utilities in this volume of IEEE Std. 1003.1-200x work on text files. The *cut* utility can be  
 11138 used to turn files with arbitrary line lengths into a set of text files containing the same data. The  
 11139 *paste* utility can be used to create (or recreate) files with arbitrary line lengths. For example, if **file**  
 11140 contains long lines:

11141 `cut -b 1-500 -n file > file1`11142 `cut -b 501- -n file > file2`

11143 creates **file1** (a text file) with lines no longer than 500 bytes (plus the <newline> character) and  
 11144 **file2** that contains the remainder of the data from **file**. (Note that **file2** is not a text file if there  
 11145 are lines in **file** that are longer than 500 + {LINE\_MAX} bytes.) The original file can be recreated  
 11146 from **file1** and **file2** using the command:

11147       paste -d "\0" file1 file2 > file

#### 11148 RATIONALE

11149       Some historical implementations do not count <backspace> characters in determining character  
11150       counts with the `-c` option. This may be useful for using *cut* for processing *nroff* output. It was  
11151       deliberately decided not to have the `-c` option treat either <backspace> or <tab> characters in  
11152       any special fashion. The *fold* utility does treat these characters specially.

11153       Unlike other utilities, some historical implementations of *cut* exit after not finding an input file,  
11154       rather than continuing to process the remaining *file* operands. This behavior is prohibited by this  
11155       volume of IEEE Std. 1003.1-200x, where only the exit status is affected by this problem.

11156       The behavior of *cut* when provided with either mutually-exclusive options or options that do  
11157       not work logically together has been deliberately left unspecified in favor of global wording in  
11158       Section 1.11 (on page 2224).

11159       The OPTIONS section was changed in response to P1003.2-N149. The change represents  
11160       historical practice on all known systems. The original standard was ambiguous on the nature of  
11161       the output.

11162       The *list* option-arguments are historically used to select the portions of the line to be written, but  
11163       do not affect the order of the data. For example:

11164       echo abcdefghi | cut -c6,2,4-7,1

11165       yields "abdefg".

11166       A proposal to enhance *cut* with the following option:

11167       **-o** Preserve the selected field order. When this option is specified, each byte, character, or field  
11168       (or ranges of such) shall be written in the order specified by the *list* option-argument, even if  
11169       this requires multiple outputs of the same bytes, characters, or fields.

11170       was rejected because this type of enhancement is outside the scope of the IEEE P1003.2b draft  
11171       standard.

#### 11172 FUTURE DIRECTIONS

11173       None.

#### 11174 SEE ALSO

11175       *grep*, *paste*, Section 2.5 (on page 2241)

#### 11176 CHANGE HISTORY

11177       First released in Issue 2.

#### 11178 Issue 4

11179       Aligned with the ISO/IEC 9945-2:1993 standard.

#### 11180 Issue 6

11181       The OPTIONS section is changed to align with the IEEE P1003.2b draft standard.

11182       The normative text is reworded to avoid use of the term “must” for application requirements.

11183 **NAME**11184 cxref — generate a C-language program cross-reference table (**DEVELOPMENT**)11185 **SYNOPSIS**

```
11186 xsl cxref [-cs][-o file][-w num] [-D name[=def]]...[-I dir]...
11187 [-U name]... file ...
```

11188

11189 **DESCRIPTION**

11190 The *cxref* utility shall analyze a collection of C-language *files* and attempt to build a cross-  
 11191 reference table. Information from **#define** lines is included in the symbol table. A sorted listing  
 11192 shall be written to standard output of all symbols (auto, static, and global) in each *file* separately,  
 11193 or with the *-c* option, in combination. Each symbol contains an asterisk before the declaring  
 11194 reference.

11195 **OPTIONS**

11196 The *cxref* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 11197 12.2, Utility Syntax Guidelines, except that the order of the *-D*, *-I*, and *-U* options (which are  
 11198 identical to their interpretation by *c99*) is significant. The following options shall be supported:

- 11199 *-c* Write a combined cross-reference of all input files.
- 11200 *-w num* Format output no wider than *num* (decimal) columns. This option defaults to 80 if  
 11201 *num* is not specified or is less than 51.
- 11202 *-o file* Direct output to named *file*.
- 11203 *-s* Operate silently; do not print input file names.

11204 **OPERANDS**

11205 The following operand shall be supported:

- 11206 *file* A path name of a C-language source file.

11207 **STDIN**

11208 Not used.

11209 **INPUT FILES**

11210 The input files are C-language source files.

11211 **ENVIRONMENT VARIABLES**11212 The following environment variables shall affect the execution of *cxref*:

- 11213 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 11214 If *LANG* is unset or null, the corresponding value from the implementation-  
 11215 defined default locale shall be used. If any of the internationalization variables  
 11216 contains an invalid setting, the utility shall behave as if none of the variables had  
 11217 been defined.

- 11218 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 11219 internationalization variables.

11220 *LC\_COLLATE*

11221 Determine the locale for the ordering of the output.

- 11222 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 11223 characters (for example, single-byte as opposed to multi-byte characters in  
 11224 arguments and input files).

11225 *LC\_MESSAGES*

11226 Determine the locale that should be used to affect the format and contents of

- 11227 diagnostic messages written to standard error.
- 11228 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 11229 **ASYNCHRONOUS EVENTS**
- 11230 Default.
- 11231 **STDOUT**
- 11232 The standard output shall be used for the cross-reference listing, unless the **-o** option is used to  
11233 select a different output file.
- 11234 The format of standard output is unspecified, except that the following information shall be  
11235 included:
- 11236 • If the **-c** option is not specified, each portion of the listing starts with the name of the input  
11237 file on a separate line.
  - 11238 • The name line is followed by a sorted list of symbols, each with its associated location path  
11239 name, the name of the function in which it appears (if it is not a function name itself), and  
11240 line number references.
  - 11241 • Each line number may be preceded by an asterisk (**'\*'**) flag, meaning that this is the  
11242 declaring reference. Other single-character flags, with implementation-defined meanings,  
11243 may be included.
- 11244 **STDERR**
- 11245 Used only for diagnostic messages.
- 11246 **OUTPUT FILES**
- 11247 The output file named by the **-o** option shall be used instead of standard output.
- 11248 **EXTENDED DESCRIPTION**
- 11249 None.
- 11250 **EXIT STATUS**
- 11251 The following exit values shall be returned:
- 11252 0 Successful completion.
- 11253 >0 An error occurred.
- 11254 **CONSEQUENCES OF ERRORS**
- 11255 Default.
- 11256 **APPLICATION USAGE**
- 11257 None.
- 11258 **EXAMPLES**
- 11259 None.
- 11260 **RATIONALE**
- 11261 None.
- 11262 **FUTURE DIRECTIONS**
- 11263 None.
- 11264 **SEE ALSO**
- 11265 *c99*



11266 **CHANGE HISTORY**

11267 First released in Issue 2.

11268 **Issue 4**

11269 Format reorganized.

11270 Utility Syntax Guidelines support mandated.

11271 Internationalized environment variable support mandated.

11272 **Issue 5**11273 In the SYNOPSIS, [-U *dir*]ischangedto[-U *name*].11274 **Issue 6**

11275 The APPLICATION USAGE section is added.

11276 **NAME**

11277           date — write the date and time

11278 **SYNOPSIS**

11279           date [-u] [+format]

11280 xSI       date [-u] *mmddhhmm*[[*cc*]*yy*]

11281

11282 **DESCRIPTION**

11283 xSI       The *date* utility shall write the date and time to standard output or attempt to set the system date  
 11284       and time. By default, the current date and time shall be written. If an operand beginning with  
 11285       '+' is specified, the output format of *date* shall be controlled by the field descriptors and other  
 11286       text in the operand.

11287 **OPTIONS**

11288       The *date* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 11289       12.2, Utility Syntax Guidelines.

11290       The following option shall be supported:

11291       **-u**       Perform operations as if the *TZ* environment variable was set to the string "UTC0",  
 11292       or its equivalent historical value of "GMT0". Otherwise, *date* shall use the  
 11293       timezone indicated by the *TZ* environment variable or the system default if that  
 11294       variable is not set.

11295 **OPERANDS**

11296       The following operands shall be supported:

11297       **+format**   When the format is specified, each field descriptor shall be replaced in the  
 11298       standard output by its corresponding value. All other characters shall be copied to  
 11299       the output without change. The output always shall be terminated with a  
 11300       <newline> character.

11301           **Field Descriptors**

11302           **%a**       Locale's abbreviated weekday name.

11303           **%A**       Locale's full weekday name.

11304           **%b**       Locale's abbreviated month name.

11305           **%B**       Locale's full month name.

11306           **%c**       Locale's appropriate date and time representation.

11307           **%C**       Century (a year divided by 100 and truncated to an integer) as a decimal  
 11308           number [00-99].

11309           **%d**       Day of the month as a decimal number [01-31].

11310           **%D**       Date in the format *mm/dd/yy*.

11311           **%e**       Day of the month as a decimal number [1-31] in a two-digit field with  
 11312           leading space character fill.

11313           **%h**       A synonym for *%b*.

11314           **%H**       Hour (24-hour clock) as a decimal number [00-23].

11315           **%I**       Hour (12-hour clock) as a decimal number [01-12].

|       |    |                                                                                                                                                                                                                                                                                               |
|-------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11316 | %j | Day of the year as a decimal number [001-366].                                                                                                                                                                                                                                                |
| 11317 | %m | Month as a decimal number [01-12].                                                                                                                                                                                                                                                            |
| 11318 | %M | Minute as a decimal number [00-59].                                                                                                                                                                                                                                                           |
| 11319 | %n | A <newline> character.                                                                                                                                                                                                                                                                        |
| 11320 | %p | Locale's equivalent of either AM or PM.                                                                                                                                                                                                                                                       |
| 11321 | %r | 12-hour clock time [01-12] using the AM/PM notation; in the POSIX locale, this is equivalent to %I:%M:%S% p.                                                                                                                                                                                  |
| 11322 |    |                                                                                                                                                                                                                                                                                               |
| 11323 | %S | Seconds as a decimal number [00-61].                                                                                                                                                                                                                                                          |
| 11324 | %t | A <tab> character.                                                                                                                                                                                                                                                                            |
| 11325 | %T | 24-hour clock time [00-23] in the format <i>HH:MM:SS</i> .                                                                                                                                                                                                                                    |
| 11326 | %u | Weekday as a decimal number [1 (Monday)-7].                                                                                                                                                                                                                                                   |
| 11327 | %U | Week of the year (Sunday as the first day of the week) as a decimal number [00-53]. All days in a new year preceding the first Sunday shall be considered to be in week 0.                                                                                                                    |
| 11328 |    |                                                                                                                                                                                                                                                                                               |
| 11329 |    |                                                                                                                                                                                                                                                                                               |
| 11330 | %V | Week of the year (Monday as the first day of the week) as a decimal number [01-53]. If the week containing January 1 has four or more days in the new year, then it shall be considered week 1; otherwise, it shall be the last week of the previous year, and the next week shall be week 1. |
| 11331 |    |                                                                                                                                                                                                                                                                                               |
| 11332 |    |                                                                                                                                                                                                                                                                                               |
| 11333 |    |                                                                                                                                                                                                                                                                                               |
| 11334 | %w | Weekday as a decimal number [0 (Sunday)-6].                                                                                                                                                                                                                                                   |
| 11335 | %W | Week of the year (Monday as the first day of the week) as a decimal number [00-53]. All days in a new year preceding the first Monday shall be considered to be in week 0.                                                                                                                    |
| 11336 |    |                                                                                                                                                                                                                                                                                               |
| 11337 |    |                                                                                                                                                                                                                                                                                               |
| 11338 | %x | Locale's appropriate date representation.                                                                                                                                                                                                                                                     |
| 11339 | %X | Locale's appropriate time representation.                                                                                                                                                                                                                                                     |
| 11340 | %y | Year within century [00-99].                                                                                                                                                                                                                                                                  |
| 11341 | %Y | Year with century as a decimal number.                                                                                                                                                                                                                                                        |
| 11342 | %Z | Timezone name, or no characters if no timezone is determinable.                                                                                                                                                                                                                               |
| 11343 | %% | A percent sign character.                                                                                                                                                                                                                                                                     |
| 11344 |    | See the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3.5, <i>LC_TIME</i>                                                                                                                                                                                                       |
| 11345 |    | for the field descriptor values in the POSIX locale.                                                                                                                                                                                                                                          |

#### 11346 **Modified Field Descriptors**

|       |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11347 |     | Some field descriptors can be modified by the <i>E</i> and <i>O</i> modifier characters to indicate a different format or specification as specified in the <i>LC_TIME</i> locale description (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3.5, <i>LC_TIME</i> ). If the corresponding keyword (see <b>era</b> , <b>era_year</b> , <b>era_d_fmt</b> , and <b>alt_digits</b> in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3.5, <i>LC_TIME</i> ) is not specified or not supported for the current locale, the unmodified field descriptor value shall be used. |
| 11348 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11349 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11350 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11351 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11352 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11353 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11354 | %Ec | Locale's alternative appropriate date and time representation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|       |     |                                                                                                                                           |
|-------|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| 11355 | %EC | The name of the base year (period) in the locale's alternative representation.                                                            |
| 11356 |     |                                                                                                                                           |
| 11357 | %Ex | Locale's alternative date representation.                                                                                                 |
| 11358 | %EX | Locale's alternative time representation.                                                                                                 |
| 11359 | %Ey | Offset from %EC (year only) in the locale's alternative representation.                                                                   |
| 11360 | %EY | Full alternative year representation.                                                                                                     |
| 11361 | %Od | Day of month using the locale's alternative numeric symbols.                                                                              |
| 11362 | %Oe | Day of month using the locale's alternative numeric symbols.                                                                              |
| 11363 | %OH | Hour (24-hour clock) using the locale's alternative numeric symbols.                                                                      |
| 11364 | %OI | Hour (12-hour clock) using the locale's alternative numeric symbols.                                                                      |
| 11365 | %Om | Month using the locale's alternative numeric symbols.                                                                                     |
| 11366 | %OM | Minutes using the locale's alternative numeric symbols.                                                                                   |
| 11367 | %OS | Seconds using the locale's alternative numeric symbols.                                                                                   |
| 11368 | %Ou | Weekday as a number in the locale's alternative representation (Monday = 1).                                                              |
| 11369 |     |                                                                                                                                           |
| 11370 | %OU | Week number of the year (Sunday as the first day of the week) using the locale's alternative numeric symbols.                             |
| 11371 |     |                                                                                                                                           |
| 11372 | %OV | Week number of the year (Monday as the first day of the week, rules corresponding to %V), using the locale's alternative numeric symbols. |
| 11373 |     |                                                                                                                                           |
| 11374 | %Ow | Weekday as a number in the locale's alternative representation (Sunday = 0).                                                              |
| 11375 |     |                                                                                                                                           |
| 11376 | %OW | Week number of the year (Monday as the first day of the week) using the locale's alternative numeric symbols.                             |
| 11377 |     |                                                                                                                                           |
| 11378 | %Oy | Year (offset from %C) in alternative representation.                                                                                      |

|       |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11379 | XSI | <b>mmddhhmm[[cc]yy]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 11380 |     | Attempt to set the system date and time from the value given in the operand. This is only possible if the user has appropriate privileges and the system permits the setting of the system date and time. The first <i>mm</i> is the month (number); <i>dd</i> is the day (number); <i>hh</i> is the hour (number, 24-hour system); the second <i>mm</i> is the minute (number); <i>cc</i> is the century and is the first two digits of the year (this is optional); <i>yy</i> is the last two digits of the year and is optional. If century is not specified, then values in the range [69-99] shall refer to years 1969 to 1999 inclusive, and values in the range [00-68] shall refer to years 2000 to 2068 inclusive. |
| 11381 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11382 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11383 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11384 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11385 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11386 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 11387 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|       |              |           |
|-------|--------------|-----------|
| 11388 | <b>STDIN</b> |           |
| 11389 |              | Not used. |

|       |                    |       |
|-------|--------------------|-------|
| 11390 | <b>INPUT FILES</b> |       |
| 11391 |                    | None. |

#### 11392 ENVIRONMENT VARIABLES

|       |             |                                                                                        |
|-------|-------------|----------------------------------------------------------------------------------------|
| 11393 |             | The following environment variables shall affect the execution of <i>date</i> :        |
| 11394 | <i>LANG</i> | Provide a default value for the internationalization variables that are unset or null. |
| 11395 |             | If <i>LANG</i> is unset or null, the corresponding value from the implementation-      |

11396 defined default locale shall be used. If any of the internationalization variables  
 11397 contains an invalid setting, the utility shall behave as if none of the variables had  
 11398 been defined.

11399 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 11400 internationalization variables.

11401 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 11402 characters (for example, single-byte as opposed to multi-byte characters in  
 11403 arguments).

11404 **LC\_MESSAGES**  
 11405 Determine the locale that should be used to affect the format and contents of  
 11406 diagnostic messages written to standard error.

11407 **LC\_TIME** Determine the format and contents of date and time strings written by *date*.

11408 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

11409 **TZ** Determine the timezone in which the time and date are written, unless the **-u**  
 11410 option is specified. If the **TZ** variable is not set and the **-u** is not specified, an  
 11411 unspecified system default timezone is used.

11412 **ASYNCHRONOUS EVENTS**

11413 Default.

11414 **STDOUT**

11415 When no formatting operand is specified, the output in the POSIX locale shall be equivalent to  
 11416 specifying:  
 11417 `date "+%a %b %e %H:%M:%S %Z %Y"`

11418 **STDERR**

11419 Used only for diagnostic messages.

11420 **OUTPUT FILES**

11421 None.

11422 **EXTENDED DESCRIPTION**

11423 None.

11424 **EXIT STATUS**

11425 The following exit values shall be returned:

11426 0 The date was written successfully.

11427 >0 An error occurred.

11428 **CONSEQUENCES OF ERRORS**

11429 Default.

11430 **APPLICATION USAGE**

11431 Field descriptors are of unspecified format when not in the POSIX locale. Some of them can  
 11432 contain <newline> characters in some locales, so it may be difficult to use the format shown in  
 11433 standard output for parsing the output of *date* in those locales.

11434 The range of values for %S extends from 0 to 61 seconds to accommodate the occasional leap  
 11435 second or double leap second.

11436 Although certain of the field descriptors in the POSIX locale (such as the name of the month) are  
 11437 shown with initial capital letters, this need not be the case in other locales. Programs using these  
 11438 fields may need to adjust the capitalization if the output is going to be used at the beginning of a

11439 sentence.

11440 The date string formatting capabilities are intended for use in Gregorian-style calendars,  
11441 possibly with a different starting year (or years). The `%x` and `%c` field descriptors, however, are  
11442 intended for local representation; these may be based on a different, non-Gregorian calendar.

11443 The `%C` field descriptor was introduced to allow a fallback for the `%EC` (alternative year format  
11444 base year); it can be viewed as the base of the current subdivision in the Gregorian calendar. A  
11445 century is not calculated as an ordinal number; IEEE Std. 1003.1-200x was published in century  
11446 20, not the twenty-first. Both the `%Ey` and `%y` can then be viewed as the offset from `%EC` and  
11447 `%C`, respectively.

11448 The `E` and `O` modifiers modify the traditional field descriptors, so that they can always be used,  
11449 even if the implementation (or the current locale) does not support the modifier.

11450 The `E` modifier supports alternative date formats, such as the Japanese Emperor's Era, as long as  
11451 these are based on the Gregorian calendar system. Extending the `E` modifiers to other date  
11452 elements may provide an implementation-defined extension capable of supporting other  
11453 calendar systems, especially in combination with the `O` modifier.

11454 The `O` modifier supports time and date formats using the locale's alternative numerical symbols,  
11455 such as Kanji or Hindi digits or ordinal number representation.

11456 Non-European locales, whether they use Latin digits in computational items or not, often have  
11457 local forms of the digits for use in date formats. This is not totally unknown even in Europe; a  
11458 variant of dates uses Roman numerals for the months: the third day of September 1991 would be  
11459 written as 3.IX.1991. In Japan, Kanji digits are regularly used for dates; in Arabic-speaking  
11460 countries, Hindi digits are used. The `%d`, `%e`, `%H`, `%I`, `%m`, `%S`, `%U`, `%w`, `%W`, and `%y` field  
11461 descriptors always return the date and time field in Latin digits (that is, 0 to 9). The `%O` modifier  
11462 was introduced to support the use for display purposes of non-Latin digits. In the `LC_TIME`  
11463 category in `localedef`, the optional `alt_digits` keyword is intended for this purpose. As an  
11464 example, assume the following (partial) `localedef` source:

```
11465 alt_digits "";"I";"II";"III";"IV";"V";"VI";"VII";"VIII" \
11466 "IX";"X";"XI";"XII"
11467 d_fmt "%e.%Om.%Y"
```

11468 With the above date, the command:

```
11469 date "+%x"
```

11470 would yield 3.IX.1991. With the same `d_fmt`, but without the `alt_digits`, the command would  
11471 yield 3.9.1991.

#### 11472 EXAMPLES

11473 1. The following are input/output examples of `date` used at arbitrary times in the POSIX  
11474 locale:

```
11475 $ date
11476 Tue Jun 26 09:58:10 PDT 1990
```

```
11477 $ date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"
11478 DATE: 11/02/91
11479 TIME: 13:36:16
```

```
11480 $ date "+TIME: %r"
11481 TIME: 01:36:32 PM
```

```

11482 2. Examples for Denmark, where the default date and time format is %a %d %b %Y %T %Z:
11483 $ LANG=da_DK.iso_8859-1 date
11484 ons 02 okt 1991 15:03:32 CET

11485 $ LANG=da_DK.iso_8859-1 date "+DATO: %A den %e. %B %Y%nKLOKKEN: %H:%M:%S"
11486 DATO: onsdag den 2. oktober 1991
11487 KLOKKEN: 15:03:56

11488 3. Examples for Germany, where the default date and time format is %a %d.%h.%Y, %T %Z:
11489 $ LANG=De_DE.88591 date
11490 Mi 02.Okt.1991, 15:01:21 MEZ

11491 $ LANG=De_DE.88591 date "+DATUM: %A, %d. %B %Y%nZEIT: %H:%M:%S"
11492 DATUM: Mittwoch, 02. Oktober 1991
11493 ZEIT: 15:02:02

11494 4. Examples for France, where the default date and time format is %a %d %h %Y %Z %T:
11495 $ LANG=Fr_FR.88591 date
11496 Mer 02 oct 1991 MET 15:03:32

11497 $ LANG=Fr_FR.88591 date "+JOUR: %A %d %B %Y%nHEURE: %H:%M:%S"
11498 JOUR: Mercredi 02 octobre 1991
11499 HEURE: 15:03:56

```

## 11500 RATIONALE

```

11501 Some of the new options for formatting are from the ISO C standard. The -u option was
11502 introduced to allow portable access to Coordinated Universal Time (UTC). The string "GMT0" is
11503 allowed as an equivalent TZ value to be compatible with all of the systems using the BSD
11504 implementation, where this option originated.

11505 The %e format field descriptor (adopted from System V) was added because the ISO C standard
11506 descriptors did not provide any way to produce the historical default date output during the first
11507 nine days of any month.

11508 There are two varieties of day and week numbering supported (in addition to any others created
11509 with the locale-dependent %E and %O modifier characters):

11510 • The historical variety in which Sunday is the first day of the week and the weekdays
11511 preceding the first Sunday of the year are considered week 0. These are represented by %w
11512 and %U. A variant of this is %W, using Monday as the first day of the week, but still referring
11513 to week 0. This view of the calendar was retained because so many historical applications
11514 depend on it and the ISO C standard strftime() function, on which many date
11515 implementations are based, was defined in this way.

11516 • The international standard, based on the ISO 8601:1988 standard where Monday is the first
11517 weekday and the algorithm for the first week number is more complex: If the week (Monday
11518 to Sunday) containing January 1 has four or more days in the new year, then it is week 1;
11519 otherwise, it is week 53 of the previous year, and the next week is week 1. These are
11520 represented by the new field descriptors %u and %V, added as a result of international
11521 comments.

11522 The %C field descriptor was introduced to allow a fallback for the %EC (alternate year format
11523 base year); it can be viewed as the base of the current subdivision in the Gregorian calendar. A
11524 century is not calculated as an ordinal number. The original version of this volume of
11525 IEEE Std. 1003.1-200x was approved in century 19, not the twentieth. Both the %Ey and %y can
11526 then be viewed as the offset from %EC and %C, respectively.

```

11527 **FUTURE DIRECTIONS**

11528 None.

11529 **SEE ALSO**11530 The System Interfaces volume of IEEE Std. 1003.1-200x, *ctime()*, *printf()*11531 **CHANGE HISTORY**

11532 First released in Issue 2.

11533 **Issue 4**

11534 Aligned with the ISO/IEC 9945-2:1993 standard.

11535 **Issue 5**

11536 Changes are made for Year 2000 alignment.

11537 **Issue 6**11538 The following new requirements on POSIX implementations derive from alignment with the  
11539 Single UNIX Specification:

- 11540 • The setting of system date and time is described, including how to interpret two-digit year
- 11541 values if a century is not given.
- 11542 • The *%EX* modified field descriptor is added.

11543 The Open Group corrigenda item U048/2 has been applied, correcting the examples. |



## 11544 NAME

11545 dd — convert and copy a file

## 11546 SYNOPSIS

11547 dd [*operand* ...]

## 11548 DESCRIPTION

11549 The *dd* utility shall copy the specified input file to the specified output file with possible  
 11550 conversions using specific input and output block sizes. It shall read the input one block at a  
 11551 time, using the specified input block size; it shall then process the block of data actually  
 11552 returned, which could be smaller than the requested block size. It shall apply any conversions  
 11553 that have been specified and write the resulting data to the output in blocks of the specified  
 11554 output block size. If the **bs=expr** operand is specified and no conversions other than **sync**,  
 11555 **noerror**, or **notrunc** are requested, the data returned from each input block shall be written as a  
 11556 separate output block; if the read returns less than a full block and the **sync** conversion is not  
 11557 specified, the resulting output block shall be the same size as the input block. If the **bs=expr**  
 11558 operand is not specified, or a conversion other than **sync**, **noerror**, or **notrunc** is requested, the  
 11559 input shall be processed and collected into full-sized output blocks until the end of the input is  
 11560 reached.

11561 The processing order shall be as follows:

- 11562 1. An input block is read.
- 11563 2. If the input block is shorter than the specified input block size and the **sync** conversion is  
 11564 specified, null bytes shall be appended to the input data up to the specified size. (If either  
 11565 **block** or **unblock** is also specified, <space> characters shall be appended instead of null  
 11566 bytes.) The remaining conversions and output shall include the pad characters as if they  
 11567 had been read from the input.
- 11568 3. If the **bs=expr** operand is specified and no conversion other than **sync** or **noerror** is  
 11569 requested, the resulting data shall be written to the output as a single block, and the  
 11570 remaining steps are omitted.
- 11571 4. If the **swab** conversion is specified, each pair of input data bytes shall be swapped. If there  
 11572 is an odd number of bytes in the input block, the last byte in the input record shall not be  
 11573 swapped.
- 11574 5. Any remaining conversions (**block**, **unblock**, **lcase**, and **ucase**) shall be performed. These  
 11575 conversions shall operate on the input data independently of the input blocking; an input  
 11576 or output fixed-length record may span block boundaries.
- 11577 6. The data resulting from input or conversion or both shall be aggregated into output blocks  
 11578 of the specified size. After the end of input is reached, any remaining output shall be  
 11579 written as a block without padding if **conv=sync** is not specified; thus, the final output  
 11580 block may be shorter than the output block size.

## 11581 OPTIONS

11582 None.

## 11583 OPERANDS

11584 All of the operands shall be processed before any input is read. The following operands shall be  
 11585 supported:

- 11586 **if=file** Specify the input path name; the default is standard input.
- 11587 **of=file** Specify the output path name; the default is standard output. If the **seek=expr**  
 11588 conversion is not also specified, the output file shall be truncated before the copy  
 11589 begins, unless **conv=notrunc** is specified. If **seek=expr** is specified, but

|           |                               |                                                                                                                |
|-----------|-------------------------------|----------------------------------------------------------------------------------------------------------------|
| 11590     |                               | <b>conv=notrunc</b> is not, the effect of the copy shall be to preserve the blocks in the                      |
| 11591     |                               | output file over which <i>dd</i> seeks, but no other portion of the output file shall be                       |
| 11592     |                               | preserved. (If the size of the seek plus the size of the input file is less than the                           |
| 11593     |                               | previous size of the output file, the output file shall be shortened by the copy.)                             |
| 11594     | <b>ibs=expr</b>               | Specify the input block size, in bytes, by <i>expr</i> (default is 512).                                       |
| 11595     | <b>obs=expr</b>               | Specify the output block size, in bytes, by <i>expr</i> (default is 512).                                      |
| 11596     | <b>bs=expr</b>                | Set both input and output block sizes to <i>expr</i> bytes, superseding <b>ibs=</b> and <b>obs=</b> . If       |
| 11597     |                               | no conversion other than <b>sync</b> , <b>noerror</b> , and <b>notrunc</b> is specified, each input block      |
| 11598     |                               | shall be copied to the output as a single block without aggregating short blocks.                              |
| 11599     | <b>cbs=expr</b>               | Specify the conversion block size for <b>block</b> and <b>unblock</b> in bytes by <i>expr</i> (default is      |
| 11600     |                               | zero). If <b>cbs=</b> is omitted or given a value of zero, using <b>block</b> or <b>unblock</b> produces       |
| 11601     |                               | unspecified results.                                                                                           |
| 11602 XSI |                               | The application shall ensure that this operand is also specified if the <b>conv=</b>                           |
| 11603     |                               | operand is specified with a value of <b>ascii</b> , <b>ebcdic</b> , or <b>ibm</b> . For a <b>conv=</b> operand |
| 11604     |                               | with an <b>ascii</b> value, the input is handled as described for the <b>unblock</b> value, except             |
| 11605     |                               | that characters are converted to ASCII before any trailing <space> characters are                              |
| 11606     |                               | deleted. For <b>conv=</b> operands with <b>ebcdic</b> or <b>ibm</b> values, the input is handled as            |
| 11607     |                               | described for the <b>block</b> value except that the characters are converted to EBCDIC                        |
| 11608     |                               | or IBM EBCDIC, respectively, after any trailing <space> characters are added.                                  |
| 11609     | <b>skip=n</b>                 | Skip <i>n</i> input blocks (using the specified input block size) before starting to copy.                     |
| 11610     |                               | On seekable files, the implementation shall read the blocks or seek past them; on                              |
| 11611     |                               | non-seekable files, the blocks shall be read and the data shall be discarded.                                  |
| 11612     | <b>seek=n</b>                 | Skip <i>n</i> blocks (using the specified output block size) from beginning of the output                      |
| 11613     |                               | file before copying. On non-seekable files, existing blocks shall be read and space                            |
| 11614     |                               | from the current end-of-file to the specified offset, if any, filled with null bytes; on                       |
| 11615     |                               | seekable files, the implementation shall seek to the specified offset or read the                              |
| 11616     |                               | blocks as described for non-seekable files.                                                                    |
| 11617     | <b>count=n</b>                | Copy only <i>n</i> input blocks.                                                                               |
| 11618     | <b>conv=value[,value ...]</b> |                                                                                                                |
| 11619     |                               | Where <i>values</i> are comma-separated symbols from the following list:                                       |
| 11620 XSI | <b>ascii</b>                  | Convert EBCDIC to ASCII; see Table 4-6 (on page 2518).                                                         |
| 11621 XSI | <b>ebcdic</b>                 | Convert ASCII to EBCDIC; see Table 4-6 (on page 2518).                                                         |
| 11622 XSI | <b>ibm</b>                    | Convert ASCII to a different EBCDIC set; see Table 4-7 (on page                                                |
| 11623     |                               | 2518).                                                                                                         |
| 11624     |                               | The <b>ascii</b> , <b>ebcdic</b> , and <b>ibm</b> values are mutually-exclusive.                               |
| 11625     | <b>block</b>                  | Treat the input as a sequence of <newline> character-terminated or                                             |
| 11626     |                               | end-of-file-terminated variable-length records independent of the                                              |
| 11627     |                               | input block boundaries. Each record shall be converted to a record                                             |
| 11628     |                               | with a fixed length specified by the conversion block size. Any                                                |
| 11629     |                               | <newline> character shall be removed from the input line; <space>                                              |
| 11630     |                               | characters shall be appended to lines that are shorter than their                                              |
| 11631     |                               | conversion block size to fill the block. Lines that are longer than the                                        |
| 11632     |                               | conversion block size shall be truncated to the largest number of                                              |
| 11633     |                               | characters that fit into that size; the number of truncated lines shall                                        |
| 11634     |                               | be reported (see the STDERR section).                                                                          |

|       |                |                                                                                                                       |
|-------|----------------|-----------------------------------------------------------------------------------------------------------------------|
| 11635 |                | The <b>block</b> and <b>unblock</b> values are mutually-exclusive.                                                    |
| 11636 | <b>unblock</b> | Convert fixed-length records to variable length. Read a number of                                                     |
| 11637 |                | bytes equal to the conversion block size (or the number of bytes                                                      |
| 11638 |                | remaining in the input, if less than the conversion block size), delete                                               |
| 11639 |                | all trailing <space> characters, and append a <newline> character.                                                    |
| 11640 | <b>lcase</b>   | Map uppercase characters specified by the <i>LC_CTYPE</i> keyword                                                     |
| 11641 |                | <b>tolower</b> to the corresponding lowercase character. Characters for                                               |
| 11642 |                | which no mapping is specified shall not be modified by this                                                           |
| 11643 |                | conversion.                                                                                                           |
| 11644 |                | The <b>lcase</b> and <b>ucase</b> symbols are mutually-exclusive.                                                     |
| 11645 | <b>ucase</b>   | Map lowercase characters specified by the <i>LC_CTYPE</i> keyword                                                     |
| 11646 |                | <b>toupper</b> to the corresponding uppercase character. Characters for                                               |
| 11647 |                | which no mapping is specified shall not be modified by this                                                           |
| 11648 |                | conversion.                                                                                                           |
| 11649 | <b>swab</b>    | Swap every pair of input bytes.                                                                                       |
| 11650 | <b>noerror</b> | Do not stop processing on an input error. When an input error                                                         |
| 11651 |                | occurs, a diagnostic message shall be written on standard error,                                                      |
| 11652 |                | followed by the current input and output block counts in the same                                                     |
| 11653 |                | format as used at completion (see the <i>STDERR</i> section). If the <b>sync</b>                                      |
| 11654 |                | conversion is specified, the missing input shall be replaced with null                                                |
| 11655 |                | bytes and processed normally; otherwise, the input block shall be                                                     |
| 11656 |                | omitted from the output.                                                                                              |
| 11657 | <b>notrunc</b> | Do not truncate the output file. Preserve blocks in the output file not                                               |
| 11658 |                | explicitly written by this invocation of the <i>dd</i> utility. (See also the                                         |
| 11659 |                | preceding <b>of=file</b> operand.)                                                                                    |
| 11660 | <b>sync</b>    | Pad every input block to the size of the <b>ibs=</b> buffer, appending null                                           |
| 11661 |                | bytes. (If either <b>block</b> or <b>unblock</b> is also specified, append <space>                                    |
| 11662 |                | characters, rather than null bytes.)                                                                                  |
| 11663 |                | The behavior is unspecified if operands other than <b>conv=</b> are specified more than once.                         |
| 11664 |                | For the <b>bs=</b> , <b>cbs=</b> , <b>ibs=</b> , and <b>obs=</b> operands, the application shall supply an expression |
| 11665 |                | specifying a size in bytes. The expression, <i>expr</i> , can be:                                                     |
| 11666 |                | 1. A positive decimal number                                                                                          |
| 11667 |                | 2. A positive decimal number followed by <i>k</i> , specifying multiplication by 1 024                                |
| 11668 |                | 3. A positive decimal number followed by <i>b</i> , specifying multiplication by 512                                  |
| 11669 |                | 4. Two or more positive decimal numbers (with or without <i>k</i> or <i>b</i> ) separated by <i>x</i> , specifying    |
| 11670 |                | the product of the indicated values                                                                                   |
| 11671 |                | All of the operands are processed before any input is read.                                                           |
| 11672 | XSI            | The following two tables display the octal number character values used for the <b>ascii</b> and <b>ebcdic</b>        |
| 11673 |                | conversions (first table) and for the <b>ibm</b> conversion (second table). In both tables, the ASCII                 |
| 11674 |                | values are the row and column headers and the EBCDIC values are found at their intersections.                         |
| 11675 |                | For example, ASCII 0012 (LF) is the second row, third column, yielding 0045 in EBCDIC. The                            |
| 11676 |                | inverted tables (for EBCDIC to ASCII conversion) are not shown, but are in one-to-one                                 |
| 11677 |                | correspondence with these tables. The differences between the two tables are highlighted by                           |
| 11678 |                | small boxes drawn around five entries.                                                                                |

11679 **Notes to Reviewers**11680 *This section with side shading will not appear in the final copy. - Ed.*

11681 The following 2 tables are commented out of this draft to make document handling easier  
 11682 (ability to print 2-up). There are no changes to them. These diagrams are available from the  
 11683 Austin Group web site as a separate PDF file.

11684 **Table 4-6** ASCII to EBCDIC Conversion11685 **Table 4-7** ASCII to IBM EBCDIC Conversion11686 **STDIN**11687 If no **if=** operand is specified, the standard input shall be used. See the INPUT FILES section.11688 **INPUT FILES**

11689 The input file can be any file type.

11690 **ENVIRONMENT VARIABLES**11691 The following environment variables shall affect the execution of *dd*:

11692 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 11693 If *LANG* is unset or null, the corresponding value from the implementation-  
 11694 defined default locale shall be used. If any of the internationalization variables  
 11695 contains an invalid setting, the utility shall behave as if none of the variables had  
 11696 been defined.

11697 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 11698 internationalization variables.

11699 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 11700 characters (for example, single-byte as opposed to multi-byte characters in  
 11701 arguments and input files), the classification of characters as uppercase or  
 11702 lowercase, and the mapping of characters from one case to the other.

11703 **LC\_MESSAGES**

11704 Determine the locale that should be used to affect the format and contents of  
 11705 diagnostic messages written to standard error and informative messages written to  
 11706 standard output.

11707 **XSI** **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.11708 **ASYNCHRONOUS EVENTS**

11709 For SIGINT, the *dd* utility shall interrupt its current processing, write status information to  
 11710 standard error, and exit as though terminated by SIGINT. It shall take the standard action for all  
 11711 other signals; see the ASYNCHRONOUS EVENTS section in Section 1.11 (on page 2224).

11712 **STDOUT**

11713 If no **of=** operand is specified, the standard output shall be used. The nature of the output  
 11714 depends on the operands selected.

11715 **STDERR**

11716 On completion, *dd* shall write the number of input and output blocks to standard error. In the  
 11717 POSIX locale the following formats shall be used:

11718 "%u+%u records in\n", <number of whole input blocks>,  
 11719 <number of partial input blocks>

11720 "%u+%u records out\n", <number of whole output blocks>,  
 11721 <number of partial output blocks>

11722 A partial input block is one for which *read()* returned less than the input block size. A partial  
 11723 output block is one that was written with fewer bytes than specified by the output block size.

11724 In addition, when there is at least one truncated block, the number of truncated blocks shall be  
 11725 written to standard error. In the POSIX locale, the format shall be:

11726 "%u truncated %s\n", *<number of truncated blocks>*, "record" (if  
 11727 *<number of truncated blocks>* is one) "records" (otherwise)

11728 Diagnostic messages may also be written to standard error.

#### 11729 OUTPUT FILES

11730 If the **of=** operand is used, the output shall be the same as described in the STDOUT section.

#### 11731 EXTENDED DESCRIPTION

11732 None.

#### 11733 EXIT STATUS

11734 The following exit values shall be returned:

11735 0 The input file was copied successfully.

11736 >0 An error occurred.

#### 11737 CONSEQUENCES OF ERRORS

11738 If an input error is detected and the **noerror** conversion has not been specified, any partial  
 11739 output block shall be written to the output file, a diagnostic message shall be written, and the  
 11740 copy operation shall be discontinued. If some other error is detected, a diagnostic message shall  
 11741 be written and the copy operation shall be discontinued.

#### 11742 APPLICATION USAGE

11743 The input and output block size can be specified to take advantage of raw physical I/O.

11744 There are many different versions of the EBCDIC codesets. The ASCII and EBCDIC conversions  
 11745 specified for the *dd* utility perform conversions for the version specified by the tables.

#### 11746 EXAMPLES

11747 The following command:

```
11748 dd if=/dev/rmt0h of=/dev/rmt1h
```

11749 copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

11750 The following command:

```
11751 dd ibs=10 skip=1
```

11752 strips the first 10 bytes from standard input.

11753 This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the  
 11754 ASCII file **x**:

```
11755 dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

#### 11756 RATIONALE

11757 The OPTIONS section is listed as “None” because there are no options recognized by historical  
 11758 *dd* utilities. Certainly, many of the operands could have been designed to use the Utility Syntax  
 11759 Guidelines, which would have resulted in the classic hyphenated option letters. In this version  
 11760 of this volume of IEEE Std. 1003.1-200x, *dd* retains its curious JCL-like syntax due to the large  
 11761 number of applications that depend on the historical implementation.

11762 A suggested implementation technique for **conv=noerror, sync** is to zero (or <space>-fill, if  
 11763 **blocking** or **unblocking**) the input buffer before each read and to write the contents of the input

11764 buffer to the output even after an error. In this manner, any data transferred to the input buffer  
 11765 before the error was detected is preserved. Another point is that a failed read on a regular file or  
 11766 a disk generally does not increment the file offset, and *dd* must then seek past the block on which  
 11767 the error occurred; otherwise, the input error occurs repetitively. When the input is a magnetic  
 11768 tape, however, the tape normally has passed the block containing the error when the error is  
 11769 reported, and thus no seek is necessary.

11770 The default **ibs=** and **obs=** sizes are specified as 512 bytes because there are historical (largely  
 11771 portable) scripts that assume these values. If they were left unspecified, unusual results could  
 11772 occur if an implementation chose an odd block size.

11773 Historical implementations of *dd* used *creat()* when processing **of=file**. This makes the **seek=**  
 11774 operand unusable except on special files. The **conv=notrunc** feature was added because more  
 11775 recent BSD-based implementations use *open()* (without `O_TRUNC`) instead of *creat()*, but they  
 11776 fail to delete output file contents after the data copied.

11777 The *w* multiplier (historically meaning *word*), is used in System V to mean 2 and in 4.2 BSD to  
 11778 mean 4. Since *word* is inherently non-portable, its use is not supported by this volume of  
 11779 IEEE Std. 1003.1-200x.

11780 Standard EBCDIC does not have the characters '[' and ']'. The values used in the table are  
 11781 taken from a common print train that does contain them. Other than those characters, the print  
 11782 train values are not filled in, but appear to provide some of the motivation for the historical  
 11783 choice of translations reflected here.

11784 The Standard EBCDIC table provides a 1:1 translation for all 256 bytes.

11785 The IBM EBCDIC table does not provide such a translation. The marked cells in the tables differ  
 11786 in such a way that:

- 11787 1. EBCDIC 0112 ('ϕ') and 0152 (broken pipe) do not appear in the table.
- 11788 2. EBCDIC 0137 ('¬') translates to/from ASCII 0236 ('^'). In the standard table, EBCDIC  
 11789 0232 (no graphic) is used.
- 11790 3. EBCDIC 0241 ('~') translates to/from ASCII 0176 ('~'). In the standard table, EBCDIC  
 11791 0137 ('¬') is used.
- 11792 4. 0255 ('[') and 0275 (']') appear twice, once in the same place as for the standard table  
 11793 and once in place of 0112 ('ϕ') and 0241 ('~').

11794 In net result:

11795 EBCDIC 0275 (']') displaced EBCDIC 0241 ('~') in cell 0345.

11796 That displaced EBCDIC 0137 ('¬') in cell 0176.

11797 That displaced EBCDIC 0232 (no graphic) in cell 0136.

11798 That replaced EBCDIC 0152 (broken pipe) in cell 0313.

11799 EBCDIC 0255 ('[') replaced EBCDIC 0112 ('ϕ').

11800 This translation, however, reflects historical practice that (ASCII) '~' and '¬' were often  
 11801 mapped to each other, as were '[' and 'ϕ'; and ']' and (EBCDIC) '~'.

11802 The **chs** operand is required if any of the **ascii**, **ebcdic**, or **ibm** operands are specified. For the  
 11803 **ascii** operand, the input is handled as described for the **unblock** operand except that characters  
 11804 are converted to ASCII before the trailing <space>s are deleted. For the **ebcdic** and **ibm**  
 11805 operands, the input is handled as described for the **block** operand except that the characters are  
 11806 converted to EBCDIC or IBM EBCDIC after the trailing <space>s are added.

- 11807 The **block** and **unblock** keywords are from historical BSD practice.
- 11808 The consistent use of the word **record** in standard error messages matches most historical  
11809 practice. An earlier version of System V used **block**, but this has been updated in more recent  
11810 releases.
- 11811 Early proposals only allowed two numbers separated by **x** to be used in a product when  
11812 specifying **bs=**, **cbs=**, **ibs=**, and **obs=** sizes. This was changed to reflect the historical practice of  
11813 allowing multiple numbers in the product as provided by Version 7 and all releases of System V  
11814 and BSD.
- 11815 A change to the *swab* conversion is required to match historical practice and is the result of IEEE  
11816 PASC Interpretation 1003.2 #03 and #04, submitted for the ISO POSIX-2: 1993 standard.
- 11817 A change to the handling of SIGINT is required to match historical practice and is the result of  
11818 IEEE PASC Interpretation 1003.2 #06 submitted for the ISO POSIX-2: 1993 standard.
- 11819 **FUTURE DIRECTIONS**
- 11820 None.
- 11821 **SEE ALSO**
- 11822 *sed, tr*
- 11823 **CHANGE HISTORY**
- 11824 First released in Issue 2.
- 11825 **Issue 4**
- 11826 Aligned with the ISO/IEC 9945-2: 1993 standard.
- 11827 **Issue 5**
- 11828 The second paragraph of the **cbs=** description is reworded and marked EX.
- 11829 FUTURE DIRECTIONS section added.
- 11830 **Issue 6**
- 11831 Changes are made to *swab* conversion and SIGINT handling to align with the IEEE P1003.2b  
11832 draft standard.
- 11833 The normative text is reworded to avoid use of the term “must” for application requirements.

## 11834 NAME

11835 delta — make a delta (change) to an SCCS file (**DEVELOPMENT**)

## 11836 SYNOPSIS

```
11837 xSI delta [-nps][-g list][-m mrlist][-r SID][-y[comment]] file...
```

11838

## 11839 DESCRIPTION

11840 The *delta* utility shall be used to permanently introduce into the named SCCS files changes that  
11841 were made to the files retrieved by *get* (called the *g-files*, or generated files).

## 11842 OPTIONS

11843 The *delta* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
11844 12.2, Utility Syntax Guidelines, except that the *-y* option has an optional option-argument. This  
11845 optional option-argument cannot be presented as a separate argument.

11846 The following options shall be supported:

11847 *-r SID* Uniquely identify which delta is to be made to the SCCS file. The use of this option  
11848 is necessary only if two or more outstanding *get* commands for editing (*get -e*) on  
11849 the same SCCS file were done by the same person (login name). The SID value  
11850 specified with the *-r* option can be either the SID specified on the *get* command  
11851 line or the SID to be made as reported by the *get* utility; see *get* (on page 2685).

11852 *-s* Suppress the report to standard output of the activity associated with each *file*.  
11853 See the STDOUT section.

11854 *-n* Specify retention of the edited *g-file* (normally removed at completion of delta  
11855 processing).

11856 *-g list* Specify a *list*, (see *get* (on page 2685) for the definition of *list*) of deltas that shall be  
11857 ignored when the file is accessed at the change level (SID) created by this delta.

11858 *-m mrlist* Specify a modification request (MR) number that the application shall supply as  
11859 the reason for creating the new delta. This is used if the SCCS file has the *v* flag set;  
11860 see *admin* (on page 2340).

11861 If *-m* is not used and the standard input is a terminal, the prompt described in the  
11862 STDOUT section shall be written to standard output before the standard input is  
11863 read; if the standard input is not a terminal, no prompt shall be issued.

11864 MRs in a list shall be separated by <blank>*s*. An unescaped <newline> character  
11865 shall terminate the MR list.

11866 If the *v* flag has a value, it shall be taken to be the name of a program which  
11867 validates the correctness of the MR numbers. If a non-zero exit status is returned  
11868 from the MR number validation program, the *delta* utility shall terminate. (It is  
11869 assumed that the MR numbers were not all valid.)

11870 *-y[comment]* Describe the reason for making the delta. The *comment* shall be an arbitrary group  
11871 of lines that would meet the definition of a text file. Implementations shall support  
11872 *comments* from zero to 512 bytes and may support longer values. A null string  
11873 (specified as either *-y*, *-y* " ", or in response to a prompt for a comment) is  
11874 considered a valid *comment*.

11875 If *-y* is not specified and the standard input is a terminal, the prompt described in  
11876 the STDOUT section shall be written to standard output before the standard input  
11877 is read; if the standard input is not a terminal, no prompt shall be issued. An  
11878 unescaped <newline> character terminates the comment text.



- 11879                   The `-y` option shall be required if the *file* operand is specified as `'-'`.
- 11880           **-p**           Write (to standard output) the SCCS file differences before and after the delta is  
11881                   applied in *diff* format; see *diff* (on page 2529).
- 11882 **OPERANDS**
- 11883           The following operand shall be supported:
- 11884           **file**           A path name of an existing SCCS file or a directory. If *file* is a directory, the *delta*  
11885                   utility shall behave as though each file in the directory were specified as a named  
11886                   file, except that non-SCCS files (last component of the path name does not begin  
11887                   with *s.*) and unreadable files shall be silently ignored.
- 11888                   If a single instance *file* is specified as `'-'`, the standard input shall be read; each  
11889                   line of the standard input shall be taken to be the name of an SCCS file to be  
11890                   processed. Non-SCCS files and unreadable files shall be silently ignored.
- 11891 **STDIN**
- 11892           The standard input shall be a text file used only in the following cases:
- 11893
  - To read an *mrlist* or a *command* (see the `-m` and `-y` options).

11894           
  - A *file* operand is specified as `'-'`.

11895 **INPUT FILES**

11896           Input files shall be text files whose data is to be included in the SCCS files. If the first character of  
11897           any line of an input file is SOH (binary 001), the results are unspecified.

11898 **ENVIRONMENT VARIABLES**

11899           The following environment variables shall affect the execution of *delta*:

11900           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
11901                   If *LANG* is unset or null, the corresponding value from the implementation-  
11902                   defined default locale shall be used. If any of the internationalization variables  
11903                   contains an invalid setting, the utility shall behave as if none of the variables had  
11904                   been defined.

11905           **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
11906                   internationalization variables.

11907           **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
11908                   characters (for example, single-byte as opposed to multi-byte characters in  
11909                   arguments and input files).

11910           **LC\_MESSAGES**

11911                   Determine the locale that should be used to affect the format and contents of  
11912                   diagnostic messages written to standard error, and informative messages written  
11913                   to standard output.

11914           **NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

11915 **ASYNCHRONOUS EVENTS**

11916           Default.

11917 **STDOUT**

11918           The standard output shall be used only for the following messages in the POSIX locale:

11919           
  - Prompts (see the `-m` and `-y` options) in the following formats:

11920           "MRs? "

- 11921           "comments? "
- 11922           The MR prompt, if written, shall always precede the comments prompt.
- 11923           • A report of each *file*'s activities (unless the `-s` option is specified) in the following format:
- 11924           " %s\n%d inserted\n%d deleted\n%d unchanged\n", <New SID>,  
11925            <number of lines inserted>, <number of lines deleted>,  
11926            <number of lines unchanged>
- 11927 **STDERR**
- 11928           Used only for diagnostic messages.
- 11929 **OUTPUT FILES**
- 11930           Any SCCS files updated are files of an unspecified format.
- 11931 **EXTENDED DESCRIPTION**
- 11932           None.
- 11933 **EXIT STATUS**
- 11934           The following exit values shall be returned:
- 11935           0   Successful completion.
- 11936           >0  An error occurred.
- 11937 **CONSEQUENCES OF ERRORS**
- 11938           Default.
- 11939 **APPLICATION USAGE**
- 11940           None.
- 11941 **EXAMPLES**
- 11942           None.
- 11943 **RATIONALE**
- 11944           None.
- 11945 **FUTURE DIRECTIONS**
- 11946           None.
- 11947 **SEE ALSO**
- 11948           *admin, diff, get, prs, rmdel*
- 11949 **CHANGE HISTORY**
- 11950           First released in Issue 2.
- 11951 **Issue 4**
- 11952           Format reorganized.
- 11953           Exceptions to Utility Syntax Guidelines conformance noted.
- 11954           Internationalized environment variable support mandated.
- 11955 **Issue 5**
- 11956           The output format description in the `STDOUT` section is corrected.
- 11957 **Issue 6**
- 11958           The `APPLICATION USAGE` section is added.
- 11959           The normative text is reworded to avoid use of the term "must" for application requirements.
- 11960           The normative text is reworded to emphasise the term "shall" for implementation requirements.

11961 **NAME**

11962           df — report free disk space

11963 **SYNOPSIS**11964 UP XSI   df [-k][-P|-t][*file...*]

11965

11966 **DESCRIPTION**

11967 XSI       The *df* utility shall write the amount of available space and file slots for file systems on which the  
 11968           invoking user has appropriate read access. File systems shall be specified by the *file* operands;  
 11969           when none are specified, information shall be written for all file systems. The format of the  
 11970           default output from *df* is unspecified, but all space figures are reported in 512-byte units, unless  
 11971           the **-k** option is specified. This output shall contain at least the file system names, amount of  
 11972 XSI       available space on each of these file systems, and the number of free file slots, or *inodes*,  
 11973           available; when **-t** is specified, the output contains the total allocated space as well.

11974 **OPTIONS**

11975           The *df* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 11976           Utility Syntax Guidelines.

11977           The following options shall be supported:

11978           **-k**           Use 1024-byte units, instead of the default 512-byte units, when writing space  
 11979           figures.

11980           **-P**           Produce output in the format described in the STDOUT section.

11981 XSI       **-t**           Include total allocated-space figures in the output.

11982 **OPERANDS**

11983           The following operand shall be supported:

11984           *file*           A path name of a file within the hierarchy of the desired file system. If a file other  
 11985 XSI       than a FIFO, a regular file, a directory or a special file representing the device  
 11986           containing the file system (for example, **/dev/dsk/0s1**) is specified, the results are  
 11987           unspecified. Otherwise, *df* shall write the amount of free space in the file system  
 11988           containing the specified *file* operand.

11989 **STDIN**

11990           Not used.

11991 **INPUT FILES**

11992           None.

11993 **ENVIRONMENT VARIABLES**11994           The following environment variables shall affect the execution of *df*:

11995           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 11996           If **LANG** is unset or null, the corresponding value from the implementation-  
 11997           defined default locale shall be used. If any of the internationalization variables  
 11998           contains an invalid setting, the utility shall behave as if none of the variables had  
 11999           been defined.

12000           **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
 12001           internationalization variables.

12002           **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 12003           characters (for example, single-byte as opposed to multi-byte characters in  
 12004           arguments).

- 12005 *LC\_MESSAGES*
- 12006 Determine the locale that should be used to affect the format and contents of
- 12007 diagnostic messages written to standard error and informative messages written to
- 12008 standard output.
- 12009 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 12010 **ASYNCHRONOUS EVENTS**
- 12011 Default.
- 12012 **STDOUT**
- 12013 When both the **-k** and **-P** options are specified, the following header line shall be written (in the
- 12014 POSIX locale):
- 12015 "Filesystem 1024-blocks Used Available Capacity Mounted on\n"
- 12016 When the **-P** option is specified without the **-k** option, the following header line shall be written
- 12017 (in the POSIX locale):
- 12018 "Filesystem 512-blocks Used Available Capacity Mounted on\n"
- 12019 The implementation may adjust the spacing of the header line and the individual data lines so
- 12020 that the information is presented in orderly columns.
- 12021 The remaining output with **-P** shall consist of one line of information for each specified file
- 12022 system. These lines shall be formatted as follows:
- 12023 "%s %d %d %d %d%% %s\n", *<file system name>*, *<total space>*,
- 12024 *<space used>*, *<space free>*, *<percentage used>*,
- 12025 *<file system root>*
- 12026 In the following list, all quantities expressed in 512-byte units (1 024-byte when **-k** is specified)
- 12027 shall be rounded up to the next higher unit. The fields are:
- 12028 *<file system name>*
- 12029 The name of the file system, in an implementation-defined format.
- 12030 *<total space>* The total size of the file system in 512-byte units. The exact meaning of this figure
- 12031 is implementation-defined, but should include *<space used>*, *<space free>*, plus any
- 12032 space reserved by the system not normally available to a user.
- 12033 *<space used>* The total amount of space allocated to existing files in the file system, in 512-byte
- 12034 units.
- 12035 *<space free>* The total amount of space available within the file system for the creation of new
- 12036 files by unprivileged users, in 512-byte units. When this figure is less than or equal
- 12037 to zero, it shall not be possible to create any new files on the file system without
- 12038 first deleting others, unless the process has appropriate privileges. The figure
- 12039 written may be less than zero.
- 12040 *<percentage used>*
- 12041 The percentage of the normally available space that is currently allocated to all
- 12042 files on the file system. This shall be calculated using the fraction:
- 12043 
$$\frac{\textit{<space used>}}{\textit{<space used>} + \textit{<space free>}}$$
- 12044 expressed as a percentage. This percentage may be greater than 100 if *<space free>*
- 12045 is less than zero. The percentage value shall be expressed as a positive integer,
- 12046 with any fractional result causing it to be rounded to the next highest integer.

12047 <file system root>  
 12048 The directory below which the file system hierarchy appears.

12049 XSI The output format is unspecified when `-t` is used.

12050 **STDERR**  
 12051 Used only for diagnostic messages.

12052 **OUTPUT FILES**  
 12053 None.

12054 **EXTENDED DESCRIPTION**  
 12055 None.

12056 **EXIT STATUS**  
 12057 The following exit values shall be returned:  
 12058 0 Successful completion.  
 12059 >0 An error occurred.

12060 **CONSEQUENCES OF ERRORS**  
 12061 Default.

12062 **APPLICATION USAGE**  
 12063 On most systems, the “name of the file system, in an implementation-defined format” is the  
 12064 special file on which the file system is mounted.  
 12065 On large file systems, the calculation specified for percentage used can create huge rounding  
 12066 errors.

12067 **EXAMPLES**  
 12068 1. The following example writes portable information about the `/usr` file system:  
 12069 `df -P /usr`  
 12070 2. Assuming that `/usr/src` is part of the `/usr` file system, the following produces the same  
 12071 output as the previous example:  
 12072 `df -P /usr/src`

12073 **RATIONALE**  
 12074 The behavior of `df` with the `-P` option is the default action of the 4.2 BSD `df` utility. The uppercase  
 12075 `-P` was selected to avoid collision with a known industry extension using `-p`.  
 12076 Historical `df` implementations vary considerably in their default output. It was therefore  
 12077 necessary to describe the default output in a loose manner to accommodate all known historical  
 12078 implementations and to add a portable option (`-P`) to provide information in a portable format.  
 12079 The use of 512-byte units is historical practice and maintains compatibility with `ls` and other  
 12080 utilities in this volume of IEEE Std. 1003.1-200x. This does not mandate that the file system itself  
 12081 be based on 512-byte blocks. The `-k` option was added as a compromise measure. It was agreed  
 12082 by the standard developers that 512 bytes was the best default unit because of its complete  
 12083 historical consistency on System V (*versus* the mixed 512/1024-byte usage on BSD systems), and  
 12084 that a `-k` option to switch to 1024-byte units was a good compromise. Users who prefer the  
 12085 more logical 1024-byte quantity can easily alias `df` to `df -k` without breaking many historical  
 12086 scripts relying on the 512-byte units.  
 12087 It was suggested that `df` and the various related utilities be modified to access a `BLOCKSIZE`  
 12088 environment variable to achieve consistency and user acceptance. Since this is not historical  
 12089 practice on any system, it is left as a possible area for system extensions and will be re-evaluated

12090 in a future version if it is widely implemented.

12091 **FUTURE DIRECTIONS**

12092 None.

12093 **SEE ALSO**

12094 *find*

12095 **CHANGE HISTORY**

12096 First released in Issue 2.

12097 **Issue 4**

12098 Aligned with the ISO/IEC 9945-2:1993 standard.

12099 **Issue 6**

12100 This utility is now marked as part of the User Portability Utilities option.

12101 **NAME**

12102 diff — compare two files

12103 **SYNOPSIS**12104 diff [-c | -e | -f | -C n][-br] *file1 file2*12105 **DESCRIPTION**

12106 The *diff* utility shall compare the contents of *file1* and *file2* and write to standard output a list of  
 12107 changes necessary to convert *file1* into *file2*. This list should be minimal. No output shall be  
 12108 produced if the files are identical.

12109 **OPTIONS**

12110 The *diff* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 12111 12.2, Utility Syntax Guidelines.

12112 The following options shall be supported:

12113 **-b** Cause any amount of white space at the end of a line to be treated as a single  
 12114 <newline> character (that is, the white-space characters preceding the <newline>  
 12115 character are ignored) and other strings of white-space characters, not including  
 12116 <newline> characters, to compare equal.

12117 **-c** Produce output in a form that provides three lines of context.

12118 **-C n** Produce output in a form that provides *n* lines of context (where *n* shall be  
 12119 interpreted as a positive decimal integer).

12120 **-e** Produce output in a form suitable as input for the *ed* utility, which can then be  
 12121 used to convert *file1* into *file2*.

12122 **-f** Produce output in an alternative form, similar in format to **-e**, but not intended to  
 12123 be suitable as input for the *ed* utility, and in the opposite order.

12124 **-r** Apply *diff* recursively to files and directories of the same name when *file1* and *file2*  
 12125 are both directories.

12126 **OPERANDS**

12127 The following operands shall be supported:

12128 *file1, file2* A path name of a file to be compared. If either the *file1* or *file2* operand is '-', the  
 12129 standard input shall be used in its place.

12130 If both *file1* and *file2* are directories, *diff* shall not compare block special files, character special  
 12131 files, or FIFO special files to any files and shall not compare regular files to directories. The  
 12132 system documentation shall specify the behavior of *diff* on implementation-defined file types not  
 12133 specified by the System Interfaces volume of IEEE Std. 1003.1-200x when found in directories.  
 12134 Further details are as specified in **Diff Directory Comparison Format** (on page 2530).

12135 If only one of *file1* and *file2* is a directory, *diff* shall be applied to the non-directory file and the file  
 12136 contained in the directory file with a file name that is the same as the last component of the non-  
 12137 directory file.

12138 **STDIN**

12139 The standard input shall be used only if one of the *file1* or *file2* operands references standard  
 12140 input. See the INPUT FILES section.

12141 **INPUT FILES**

12142 The input files shall be text files.

12143 **Notes to Reviewers**12144 *This section with side shading will not appear in the final copy. - Ed.*

12145 D3, XCU, ERN 75 proposes adding the following text: "If a file which is not a text file is  
 12146 encountered, a binary comparison shall be performed, and if they are not identical, an  
 12147 unspecified message containing the two file names and the string "differ" shall be produced." The  
 12148 reviewers agreed in principle; however, this change needs further cleanup such as the locale and  
 12149 output formats specifying before it can be made.

12150 **ENVIRONMENT VARIABLES**12151 The following environment variables shall affect the execution of *diff*:

12152 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 12153 If *LANG* is unset or null, the corresponding value from the implementation-  
 12154 defined default locale shall be used. If any of the internationalization variables  
 12155 contains an invalid setting, the utility shall behave as if none of the variables had  
 12156 been defined.

12157 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 12158 internationalization variables.

12159 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 12160 characters (for example, single-byte as opposed to multi-byte characters in  
 12161 arguments and input files).

12162 **LC\_MESSAGES**

12163 Determine the locale that should be used to affect the format and contents of  
 12164 diagnostic messages written to standard error and informative messages written to  
 12165 standard output.

12166 **LC\_TIME** Determine the locale for affecting the format of file timestamps written with the  
 12167 **-C** and **-c** options.

12168 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

12169 **TZ** Determine the locale for affecting the timezone used for calculating file  
 12170 timestamps written with the **-C** and **-c** options.

12171 **ASYNCHRONOUS EVENTS**

12172 Default.

12173 **STDOUT**12174 **Diff Directory Comparison Format**12175 If both *file1* and *file2* are directories, the following output formats shall be used.

12176 In the POSIX locale, each file that is present in only one directory shall be reported using the  
 12177 following format:

12178 "Only in %s: %s\n", *<directory pathname>*, *<filename>*

12179 In the POSIX locale, subdirectories that are common to the two directories may be reported with  
 12180 the following format:

12181 "Common subdirectories: %s and %s\n", *<directory1 pathname>*,  
 12182 *<directory2 pathname>*

12183 For each file common to the two directories if the two files are not to be compared, the following  
 12184 format shall be used in the POSIX locale:



12185 "File %s is a %s while file %s is a %s\n", <directory1 pathname>,  
 12186 <file type of directory1 pathname>, <directory2 pathname>,  
 12187 <file type of directory2 pathname>

12188 For each file common to the two directories, if the files are compared and are identical, no output  
 12189 shall be written. If the two files differ, the following format is written:

12190 "diff %s %s %s\n", <diff\_options>, <filename1>, <filename2>

12191 where <diff\_options> are the options as specified on the command line. Depending on these  
 12192 options, one of the following output formats shall be used to write the differences.

12193 All directory path names listed in this section shall be relative to the original command line  
 12194 arguments. All other names of files listed in this section are file names (path name components).

### 12195 **Diff Default Output Format**

12196 The default (without **-e**, **-f**, **-c**, or **-C** options) *diff* utility output shall contain lines of these  
 12197 forms:

12198 "%da%d\n", <num1>, <num2>

12199 "%da%d,%d\n", <num1>, <num2>, <num3>

12200 "%dd%d\n", <num1>, <num2>

12201 "%d,%dd%d\n", <num1>, <num2>, <num3>

12202 "%dc%d\n", <num1>, <num2>

12203 "%d,%dc%d\n", <num1>, <num2>, <num3>

12204 "%dc%d,%d\n", <num1>, <num2>, <num3>

12205 "%d,%dc%d,%d\n", <num1>, <num2>, <num3>, <num4>

12206 These lines resemble *ed* subcommands to convert *file1* into *file2*. The line numbers before the  
 12207 action letters shall pertain to *file1*; those after shall pertain to *file2*. Thus, by exchanging *a* for *d*  
 12208 and reading the line in reverse order, one can also determine how to convert *file2* into *file1*. As in  
 12209 *ed*, identical pairs (where *num1*= *num2*) are abbreviated as a single number.

12210 Following each of these lines, *diff* shall write to standard output all lines affected in the first file  
 12211 using the format:

12212 "<Δ%s", <line>

12213 and all lines affected in the second file using the format:

12214 ">Δ%s", <line>

12215 If there are lines affected in both *file1* and *file2* (as with the *c* subcommand), the changes are  
 12216 separated with a line consisting of three hyphens:

12217 "——\n"

12218 **Diff -e Output Format**

12219 With the **-e** option, a script shall be produced that shall, when provided as input to *ed*, along  
 12220 with an appended **w** (write) command, convert *file1* into *file2*. Only the **a** (append), **c** (change), **d**  
 12221 (delete), **i** (insert), and **s** (substitute) commands of *ed* shall be used in this script. Text lines,  
 12222 except those consisting of the single character period ( `'.'` ), shall be output as they appear in the  
 12223 file.

12224 **Diff -f Output Format**

12225 With the **-f** option, an alternative format of script shall be produced. It is similar to that  
 12226 produced by **-e**, with the following differences:

- 12227 1. It is expressed in reverse sequence; the output of **-e** orders changes from the end of the file  
 12228 to the beginning; the **-f** from beginning to end.
- 12229 2. The command form `<lines> <command-letter>` used by **-e** is reversed. For example,  
 12230 `10c` with **-e** would be `c10` with **-f**.
- 12231 3. The form used for ranges of line numbers is `<space>` character-separated, rather than  
 12232 comma-separated.

12233 **Diff -c or -C Output Format**

12234 With the **-c** or **-C** option, the output format shall consist of affected lines along with  
 12235 surrounding lines of context. The affected lines shall show which ones need to be deleted or  
 12236 changed in *file1*, and those added from *file2*. With the **-c** option, three lines of context, if  
 12237 available, shall be written before and after the affected lines. With the **-C** option, the user can  
 12238 specify how many lines of context are written. The exact format follows.

12239 The name and last modification time of each file shall be output in the following format:

```
12240 "**** %s %s\n", file1, <file1 timestamp>
12241 "---- %s %s\n", file2, <file2 timestamp>
```

12242 Each `<file>` field shall be the path name of the corresponding file being compared. The path  
 12243 name written for standard input is unspecified.

12244 In the POSIX locale, each `<timestamp>` field shall be equivalent to the output from the following  
 12245 command:

```
12246 date "+%a %b %e %T %Y"
```

12247 without the trailing `<newline>` character, executed at the time of last modification of the  
 12248 corresponding file (or the current time, if the file is standard input).

12249 Then, the following output formats shall be applied for every set of changes.

12250 First, a line shall be written in the following format:

```
12251 "*****\n"
```

12252 Next, the range of lines in *file1* shall be written in the following format:

```
12253 "**** %d,%d ****\n", <beginning line number>, <ending line number>
```

12254 Next, the affected lines along with lines of context (unaffected lines) shall be written. Unaffected  
 12255 lines shall be written in the following format:

```
12256 "ΔΔ%s", <unaffected_line>
```

12257 Deleted lines shall be written as:

12258 `"-Δ%s", <deleted_line>`

12259 Changed lines shall be written as:

12260 `"!Δ%s", <changed_line>`

12261 Next, the range of lines in *file2* shall be written in the following format:

12262 `"—— %d,%d ——\n", <beginning line number>, <ending line number>`

12263 Then, lines of context and changed lines shall be written as described in the previous formats.

12264 Lines added from *file2* shall be written in the following format:

12265 `" +Δ%s", <added_line>`

12266 **STDERR**

12267 Used only for diagnostic messages.

12268 **OUTPUT FILES**

12269 None.

12270 **EXTENDED DESCRIPTION**

12271 None.

12272 **EXIT STATUS**

12273 The following exit values shall be returned:

12274 0 No differences were found.

12275 1 Differences were found.

12276 >1 An error occurred.

12277 **CONSEQUENCES OF ERRORS**

12278 Default.

12279 **APPLICATION USAGE**

12280 If lines at the end of a file are changed and other lines are added, *diff* output may show this as a delete and add, as a change, or as a change and add; *diff* is not expected to know which happened and users should not care about the difference in output as long as it clearly shows the differences between the files.

12284 **EXAMPLES**

12285 If **dir1** is a directory containing a directory named **x**, **dir2** is a directory containing a directory

12286 named **x**, **dir1/x** and **dir2/x** both contain files named **date.out**, and **dir2/x** contains a file named **y**,

12287 the command:

12288 `diff -r dir1 dir2`

12289 could produce output similar to:

12290 Common subdirectories: dir1/x and dir2/x

12291 Only in dir2/x: y

12292 `diff -r dir1/x/date.out dir2/x/date.out`

12293 `lc1`

12294 `< Mon Jul 2 13:12:16 PDT 1990`

12295 `——`

12296 `> Tue Jun 19 21:41:39 PDT 1990`

## 12297 RATIONALE

12298 The `-h` option was omitted because it was insufficiently specified and does not add to  
12299 applications portability.

12300 Historical implementations employ algorithms that do not always produce a minimum list of  
12301 differences; the current language about making every effort is the best this volume of  
12302 IEEE Std. 1003.1-200x can do, as there is no metric that could be employed to judge the quality of  
12303 implementations against any and all file contents. The statement “This list should be minimal”  
12304 clearly implies that implementations are not expected to provide the following output when  
12305 comparing two 100-line files that differ in only one character on a single line:

```
12306 1,100c1,100
12307 all 100 lines from file1 preceded with "< "
12308 _____
12309 all 100 lines from file2 preceded with "> "
```

12310 The “Only in” messages required when the `-r` option is specified are not used by most historical  
12311 implementations if the `-e` option is also specified. It is required here because it provides useful  
12312 information that must be provided to update a target directory hierarchy to match a source  
12313 hierarchy. The “Common subdirectories” messages are written by System V and 4.3 BSD when  
12314 the `-r` option is specified. They are allowed here but are not required because they are reporting  
12315 on something that is the same, not reporting a difference, and are not needed to update a target  
12316 hierarchy.

12317 The `-c` option, which writes output in a format using lines of context, has been included. The  
12318 format is useful for a variety of reasons, among them being much improved readability and the  
12319 ability to understand difference changes when the target file has line numbers that differ from  
12320 another similar, but slightly different, copy. The *patch* utility is most valuable when working  
12321 with difference listings using the context format. The BSD version of `-c` takes an optional  
12322 argument specifying the amount of context. Rather than overloading `-c` and breaking the Utility  
12323 Syntax Guidelines for *diff*, the standard developers decided to add a separate option for  
12324 specifying a context *diff* with a specified amount of context (`-C`). Also, the format for context  
12325 *diffs* was extended slightly in 4.3 BSD to allow multiple changes that are within context lines  
12326 from each other to be merged together. The output format contains an additional four asterisks  
12327 after the range of affected lines in the first file name. This was to provide a flag for old programs  
12328 (like old versions of *patch*) that only understand the old context format. The version of context  
12329 described here does not require that multiple changes within context lines be merged, but it does  
12330 not prohibit it either. The extension is upward-compatible, so any vendors that wish to retain the  
12331 old version of *diff* can do so by adding the extra four asterisks (that is, utilities that currently use  
12332 *diff* and understand the new merged format will also understand the old unmerged format, but  
12333 not *vice versa*).

12334 The substitute command was added as an additional format for the `-e` option. This was added to  
12335 provide implementations a way to fix the classic “dot alone on a line” bug present in many  
12336 versions of *diff*. Since many implementations have fixed this bug, the standard developers  
12337 decided not to standardize broken behavior, but rather to provide the necessary tool for fixing  
12338 the bug. One way to fix this bug is to output two periods whenever a lone period is needed, then  
12339 terminate the append command with a period, and then use the substitute command to convert  
12340 the two periods into one period.

12341 The BSD-derived `-r` option was added to provide a mechanism for using *diff* to compare two file  
12342 system trees. This behavior is useful, is standard practice on all BSD-derived systems, and is not  
12343 easily reproducible with the *find* utility.

12344 The requirement that *diff* not compare files in some circumstances, even though they have the  
12345 same name, is based on the actual output of historical implementations. The message specified

- 12346 here is already in use when a directory is being compared to a non-directory. It is extended here  
12347 to preclude the problems arising from running into FIFOs and other files that would cause *diff* to  
12348 hang waiting for input with no indication to the user that *diff* was hung. In most common usage,  
12349 *diff -r* should indicate differences in the file hierarchies, not the difference of contents of devices  
12350 pointed to by the hierarchies.
- 12351 Many early implementations of *diff* require seekable files. Since the System Interfaces volume of  
12352 IEEE Std. 1003.1-200x supports named pipes, the standard developers decided that such a  
12353 restriction was unreasonable. Note also that the allowed file name – almost always refers to a  
12354 pipe.
- 12355 No directory search order is specified for *diff*. The historical ordering is, in fact, not optimal, in  
12356 that it prints out all of the differences at the current level, including the statements about all  
12357 common subdirectories before recursing into those subdirectories.
- 12358 The message:
- 12359 `"diff %s %s %s\n", <diff_options>, <filename1>, <filename2>`
- 12360 does not vary by locale because it is the representation of a command, not an English sentence.
- 12361 **FUTURE DIRECTIONS**
- 12362 None.
- 12363 **SEE ALSO**
- 12364 *cmp, comm, ed*
- 12365 **CHANGE HISTORY**
- 12366 First released in Issue 2.
- 12367 **Issue 4**
- 12368 Aligned with the ISO/IEC 9945-2:1993 standard.
- 12369 **Issue 5**
- 12370 FUTURE DIRECTIONS section added.
- 12371 **Issue 6**
- 12372 The following new requirements on POSIX implementations derive from alignment with the  
12373 Single UNIX Specification:
- 12374
  - The `-f` option is added.
- 12375 The output format for `-c` or `-C` format is changed to align with changes to the IEEE P1003.2b  
12376 draft standard resulting from IEEE PASC Interpretation 1003.2 #71.
- 12377 The normative text is reworded to avoid use of the term “must” for application requirements.

12378 **NAME**

12379           dirname — return the directory portion of path name

12380 **SYNOPSIS**12381           dirname *string*12382 **DESCRIPTION**

12383           The *string* operand shall be treated as a path name, as defined in the Base Definitions volume of  
 12384           IEEE Std. 1003.1-200x, Section 3.268, Path Name. The string *string* shall be converted to the name  
 12385           of the directory containing the file name corresponding to the last path name component in  
 12386           *string*, performing actions equivalent to the following steps in order:

- 12387           1. If *string* is //, skip steps 2 to 5.
- 12388           2. If *string* consists entirely of slash characters, *string* shall be set to a single slash character. In  
 12389           this case, skip steps 3 to 8.
- 12390           3. If there are any trailing slash characters in *string*, they shall be removed.
- 12391           4. If there are no slash characters remaining in *string*, *string* shall be set to a single period  
 12392           character. In this case, skip steps 5 to 8.
- 12393           5. If there are any trailing non-slash characters in *string*, they shall be removed.
- 12394           6. If the remaining *string* is //, it is implementation-defined whether steps 7 and 8 are skipped  
 12395           or processed.
- 12396           7. If there are any trailing slash characters in *string*, they shall be removed.
- 12397           8. If the remaining *string* is empty, *string* shall be set to a single slash character.

12398           The resulting string shall be written to standard output.

12399 **OPTIONS**

12400           None.

12401 **OPERANDS**

12402           The following operand shall be supported:

12403           *string*        A string.12404 **STDIN**

12405           Not used.

12406 **INPUT FILES**

12407           None.

12408 **ENVIRONMENT VARIABLES**12409           The following environment variables shall affect the execution of *dirname*:

- |                                           |                 |                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12410<br>12411<br>12412<br>12413<br>12414 | <i>LANG</i>     | Provide a default value for the internationalization variables that are unset or null. If <i>LANG</i> is unset or null, the corresponding value from the implementation-defined default locale will be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined. |
| 12415<br>12416                            | <i>LC_ALL</i>   | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                      |
| 12417<br>12418<br>12419                   | <i>LC_CTYPE</i> | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                     |

12420 **LC\_MESSAGES**

12421 Determine the locale that should be used to affect the format and contents of  
 12422 diagnostic messages written to standard error.

12423 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

12424 **ASYNCHRONOUS EVENTS**

12425 Default.

12426 **STDOUT**

12427 The *dirname* utility shall write a line to the standard output in the following format:

12428 "%s\n", <resulting string>

12429 **STDERR**

12430 Used only for diagnostic messages.

12431 **OUTPUT FILES**

12432 None.

12433 **EXTENDED DESCRIPTION**

12434 None.

12435 **EXIT STATUS**

12436 The following exit values shall be returned:

12437 0 Successful completion.

12438 >0 An error occurred.

12439 **CONSEQUENCES OF ERRORS**

12440 Default.

12441 **APPLICATION USAGE**

12442 The definition of *pathname* specifies implementation-defined behavior for path names starting  
 12443 with two slash characters. Therefore, applications shall not arbitrarily add slashes to the  
 12444 beginning of a path name unless they can ensure that there are more or less than two or are  
 12445 prepared to deal with the implementation-defined consequences.

12446 **EXAMPLES**

|       | Command                 | Results     |
|-------|-------------------------|-------------|
| 12447 | <i>dirname</i> /        | /           |
| 12448 | <i>dirname</i> //       | / or //     |
| 12449 | <i>dirname</i> /a/b/    | /a          |
| 12450 | <i>dirname</i> //a//b// | //a         |
| 12451 | <i>dirname</i>          | Unspecified |
| 12452 | <i>dirname</i> a        | .( \$? = 0) |
| 12453 | <i>dirname</i> ""       | .( \$? = 0) |
| 12454 | <i>dirname</i> /a       | /           |
| 12455 | <i>dirname</i> /a/b     | /a          |
| 12456 | <i>dirname</i> a/b      | a           |
| 12457 |                         |             |

12458 **RATIONALE**

12459 The *dirname* utility originated in System III. It has evolved through the System V releases to a  
 12460 version that matches the requirements specified in this description in System V Release 3. 4.3  
 12461 BSD and earlier versions did not include *dirname*.

12462 The behaviors of *basename* and *dirname* in this volume of IEEE Std. 1003.1-200x have been  
 12463 coordinated so that when *string* is a valid path name:

12464           \$(basename "*string*")

12465           would be a valid file name for the file in the directory:

12466           \$(dirname "*string*")

12467           This would not work for the versions of these utilities in early proposals due to the way  
12468           processing of trailing slashes was specified. Consideration was given to leaving processing  
12469           unspecified if there were trailing slashes, but this cannot be done; the Base Definitions volume of  
12470           IEEE Std. 1003.1-200x, Section 3.268, Path Name allows trailing slashes. The *basename* and  
12471           *dirname* utilities have to specify consistent handling for all valid path names.

12472 **FUTURE DIRECTIONS**

12473           None.

12474 **SEE ALSO**

12475           *basename*, Section 2.5 (on page 2241)

12476 **CHANGE HISTORY**

12477           First released in Issue 2.

12478 **Issue 4**

12479           Aligned with the ISO/IEC 9945-2:1993 standard.



12480 **NAME**12481 `du` — estimate file space usage12482 **SYNOPSIS**12483 UP `du [-a | -s][-kx][-H | -L][file ...]`

12484

12485 **DESCRIPTION**

12486 By default, the *du* utility shall write to standard output the size of the file space allocated to, and  
 12487 the size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the  
 12488 specified files. By default, when a symbolic link is encountered on the command line or in the  
 12489 file hierarchy, *du* shall count the size of the symbolic link (rather than the file referenced by the  
 12490 link), and shall not follow the link to another portion of the file hierarchy. The size of the file  
 12491 space allocated to a file of type directory shall be defined as the sum total of space allocated to  
 12492 all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself.

12493 When *du* cannot *stat()* files or *stat()* or read directories, it shall report an error condition and the  
 12494 final exit status is affected. Files with multiple links shall be counted and written for only one  
 12495 entry. The directory entry that is selected in the report is unspecified. By default, file sizes shall  
 12496 be written in 512-byte units, rounded up to the next 512-byte unit.

12497 **OPTIONS**

12498 The *du* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 12499 12.2, Utility Syntax Guidelines.

12500 The following options shall be supported:

12501 **-a** In addition to the default output, report the size of each file not of type directory in  
 12502 the file hierarchy rooted in the specified file. Regardless of the presence of the **-a**  
 12503 option, non-directories given as *file* operands shall always be listed.

12504 **-H** If a symbolic link is specified on the command line, *du* shall count the size of the  
 12505 file or file hierarchy referenced by the link.

12506 **-k** Write the files sizes in units of 1024 bytes, rather than the default 512-byte units.

12507 **-L** If a symbolic link is specified on the command line or encountered during the  
 12508 traversal of a file hierarchy, *du* shall count the size of the file or file hierarchy  
 12509 referenced by the link.

12510 **-s** Instead of the default output, report only the total sum for each of the specified  
 12511 files.

12512 **-x** When evaluating file sizes, evaluate only those files that have the same device as  
 12513 the file specified by the *file* operand.

12514 Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered  
 12515 an error. The last option specified shall determine the behavior of the utility.

12516 **OPERANDS**

12517 The following operand shall be supported:

12518 *file* The path name of a file whose size is to be written. If no *file* is specified, the current  
 12519 directory shall be used.

12520 **STDIN**

12521 Not used.

12522 **INPUT FILES**

12523       None.

12524 **ENVIRONMENT VARIABLES**12525       The following environment variables shall affect the execution of *du*:

12526       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
12527                   If *LANG* is unset or null, the corresponding value from the implementation-  
12528                   defined default locale shall be used. If any of the internationalization variables  
12529                   contains an invalid setting, the utility shall behave as if none of the variables had  
12530                   been defined.

12531       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
12532                   internationalization variables.

12533       *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
12534                   characters (for example, single-byte as opposed to multi-byte characters in  
12535                   arguments).

12536       *LC\_MESSAGES*   Determine the locale that should be used to affect the format and contents of  
12537                   diagnostic messages written to standard error.  
12538

12539 XSI      *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

12540 **ASYNCHRONOUS EVENTS**

12541       Default.

12542 **STDOUT**

12543       The output from *du* shall consist of the amount of the space allocated to a file and the name of  
12544       the file, in the following format:

12545       "%d %s\n", <size>, <pathname>

12546 **STDERR**

12547       Used only for diagnostic messages.

12548 **OUTPUT FILES**

12549       None.

12550 **EXTENDED DESCRIPTION**

12551       None.

12552 **EXIT STATUS**

12553       The following exit values shall be returned:

12554       0   Successful completion.

12555       >0   An error occurred.

12556 **CONSEQUENCES OF ERRORS**

12557       Default.

12558 **APPLICATION USAGE**

12559 None.

12560 **EXAMPLES**

12561 None.

12562 **RATIONALE**

12563 The use of 512-byte units is historical practice and maintains compatibility with *ls* and other  
12564 utilities in this volume of IEEE Std. 1003.1-200x. This does not mandate that the file system itself  
12565 be based on 512-byte blocks. The **-k** option was added as a compromise measure. It was agreed  
12566 by the standard developers that 512 bytes was the best default unit because of its complete  
12567 historical consistency on System V (*versus* the mixed 512/1 024-byte usage on BSD systems), and  
12568 that a **-k** option to switch to 1 024-byte units was a good compromise. Users who prefer the  
12569 1 024-byte quantity can easily alias *du* to *du -k* without breaking the many historical scripts  
12570 relying on the 512-byte units.

12571 The **-b** option was added to an early proposal to provide a resolution to the situation where  
12572 System V and BSD systems give figures for file sizes in *blocks*, which is an implementation-  
12573 defined concept. (In common usage, the block size is 512 bytes for System V and 1 024 bytes for  
12574 BSD systems.) However, **-b** was later deleted, since the default was eventually decided as 512-  
12575 byte units.

12576 Historical file systems provided no way to obtain exact figures for the space allocation given to  
12577 files. There are two known areas of inaccuracies in historical file systems: cases of *indirect blocks*  
12578 being used by the file system or *sparse* files yielding incorrectly high values. An indirect block is  
12579 space used by the file system in the storage of the file, but that need not be counted in the space  
12580 allocated to the file. A *sparse* file is one in which an *lseek()* call has been made to a position  
12581 beyond the end of the file and data has subsequently been written at that point. A file system  
12582 need not allocate all the intervening zero-filled blocks to such a file. It is up to the  
12583 implementation to define exactly how accurate its methods are.

12584 The **-a** and **-s** options were mutually-exclusive in the original version of *du*. The POSIX Shell  
12585 and Utilities description is implied by the language in the SVID where **-s** is described as causing  
12586 “only the grand total” to be reported. Some systems may produce output for **-sa**, but a Strictly  
12587 Conforming POSIX Shell and Utilities Application cannot use that combination.

12588 The **-a** and **-s** options were adopted from the SVID except that the System V behavior of not  
12589 listing non-directories explicitly given as operands, unless the **-a** option is specified, was  
12590 considered a bug; the BSD-based behavior (report for all operands) is mandated. The default  
12591 behavior of *du* in the SVID with regard to reporting the failure to read files (it produces no  
12592 messages) was considered counter-intuitive, and thus it was specified that the POSIX Shell and  
12593 Utilities default behavior shall be to produce such messages. These messages can be turned off  
12594 with shell redirection to achieve the System V behavior.

12595 The **-x** option is historical practice on recent BSD systems. It has been adopted by this volume of  
12596 IEEE Std. 1003.1-200x because there was no other historical method of limiting the *du* search to a  
12597 single file hierarchy. This limitation of the search is necessary to make it possible to obtain file  
12598 space usage information about a file system on which other file systems are mounted, without  
12599 having to resort to a lengthy *find* and *awk* script.

12600 **FUTURE DIRECTIONS**

12601 None.

12602 **SEE ALSO**12603 *ls*12604 **CHANGE HISTORY**

12605 First released in Issue 2.

12606 **Issue 4**

12607 Aligned with the ISO/IEC 9945-2: 1993 standard.

12608 **Issue 6**

12609 This utility is now marked as part of the User Portability Utilities option.

12610 The APPLICATION USAGE section is added.

12611 This utility is reinstated, as the LEGACY marking was incorrect in Issue 5.

12612 The obsolescent `-r` option has been removed.12613 The Open Group corrigenda item U025/3 has been applied. The *du* utility had incorrectly been  
12614 marked LEGACY.12615 The `-H` and `-L` options for symbolic links are added as described in the IEEE P1003.2b draft  
12616 standard.

12617 **NAME**

12618 echo — write arguments to standard output

12619 **SYNOPSIS**12620 echo [*string* ...]12621 **DESCRIPTION**12622 The *echo* utility writes its arguments to standard output, followed by a <newline> character. If  
12623 there are no arguments, only the <newline> character is written.12624 **OPTIONS**12625 The *echo* utility shall not recognize the "—" argument in the manner specified by Guideline 10  
12626 of the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines;  
12627 "—" shall be recognized as a string operand.

12628 Implementations shall not support any options.

12629 **OPERANDS**

12630 The following operands shall be supported:

12631 *string* A string to be written to standard output. If any operand is **-n**, it shall be treated as  
12632 a string, not an option. The following character sequences shall be recognized  
12633 within any of the arguments:

12634 \a Write an &lt;alert&gt; character.

12635 \b Write a &lt;backspace&gt; character.

12636 \c Suppress the <newline> character that otherwise follows the final  
12637 argument in the output. All characters following the '\c' in the  
12638 arguments shall be ignored.

12639 \f Write a &lt;form-feed&gt; character.

12640 \n Write a &lt;newline&gt; character.

12641 \r Write a &lt;carriage-return&gt; character.

12642 \t Write a &lt;tab&gt; character.

12643 \v Write a &lt;vertical-tab&gt; character.

12644 \\ Write a backslash character.

12645 \0*num* Write an 8-bit value that is the zero, one, two, or three-digit octal number  
12646 *num*.12647 **STDIN**

12648 Not used.

12649 **INPUT FILES**

12650 None.

12651 **ENVIRONMENT VARIABLES**12652 The following environment variables shall affect the execution of *echo*:12653 *LANG* Provide a default value for the internationalization variables that are unset or null.  
12654 If *LANG* is unset or null, the corresponding value from the implementation-  
12655 defined default locale shall be used. If any of the internationalization variables  
12656 contains an invalid setting, the utility shall behave as if none of the variables had  
12657 been defined.

12658 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 12659 internationalization variables.

12660 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 12661 characters (for example, single-byte as opposed to multi-byte characters in  
 12662 arguments).

12663 *LC\_MESSAGES*  
 12664 Determine the locale that should be used to affect the format and contents of  
 12665 diagnostic messages written to standard error.

12666 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

12667 **ASYNCHRONOUS EVENTS**  
 12668 Default.

12669 **STDOUT**  
 12670 The *echo* utility arguments shall be separated by single <space> characters and a <newline>  
 12671 character follows the last argument. Output transformations shall occur based on the escape  
 12672 sequences in the input. See the OPERANDS section.

12673 **STDERR**  
 12674 Used only for diagnostic messages.

12675 **OUTPUT FILES**  
 12676 None.

12677 **EXTENDED DESCRIPTION**  
 12678 None.

12679 **EXIT STATUS**  
 12680 The following exit values shall be returned:  
 12681 0 Successful completion.  
 12682 >0 An error occurred.

12683 **CONSEQUENCES OF ERRORS**  
 12684 Default.

12685 **APPLICATION USAGE**  
 12686 In the ISO/IEC 9945-2:1993 standard, it was not possible to use *echo* portably across all systems  
 12687 that were not XSI-conformant unless both *-n* (as the first argument) and escape sequences were  
 12688 omitted.

12689 The *printf* utility can be used portably to emulate any of the traditional behaviors of the *echo*  
 12690 utility as follows:

12691 • The historic System V *echo* and the current requirements in this volume of  
 12692 IEEE Std. 1003.1-200x are equivalent to:

```
12693 printf "%b\n" "$*"
12694
12695 • The BSD echo is equivalent to:
12696 if ["X$1" = "X-n"]
12697 then
12698 shift
12699 printf "%s" "$*"
12700 else
12701 printf "%s\n" "$*"
12702
```

- 12701            *fi*
- 12702            New applications are encouraged to use *printf* instead of *echo*.
- 12703 **EXAMPLES**
- 12704            None.
- 12705 **RATIONALE**
- 12706            The *echo* utility has not been made obsolescent because of its extremely widespread use in historical applications. Portable applications that wish to do prompting without <newline>s or that could possibly be expecting to echo a *-n*, should use the new *printf* utility derived from the Ninth Edition system.
- 12707
- 12708
- 12709
- 12710            As specified, *echo* writes its arguments in the simplest of ways. The two different historical versions of *echo* vary in fatally incompatible ways.
- 12711
- 12712            The BSD *echo* checks the first argument for the string *-n* which causes it to suppress the <newline> character that would otherwise follow the final argument in the output.
- 12713
- 12714            The System V *echo* does not support any options, but allows escape sequences within its operands, as described in the OPERANDS section.
- 12715
- 12716            The *echo* utility does not support Utility Syntax Guideline 10 because historical applications depend on *echo* to echo *all* of its arguments, except for the *-n* option in the BSD version.
- 12717
- 12718 **FUTURE DIRECTIONS**
- 12719            None.
- 12720 **SEE ALSO**
- 12721            *printf*
- 12722 **CHANGE HISTORY**
- 12723            First released in Issue 2.
- 12724 **Issue 4**
- 12725            Aligned with the ISO/IEC 9945-2:1993 standard.
- 12726 **Issue 5**
- 12727            In the OPTIONS section, the last sentence is changed to indicate that implementations “do not” support any options; in the previous issue this said “need not”.
- 12728
- 12729 **Issue 6**
- 12730            The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:
- 12731
- 12732
  - A set of character sequences is defined as *string* operands.
- 12733
  - *LC\_CTYPE* is added to the list of environment variables affecting *echo*.
- 12734
  - In the OPTIONS section, implementations shall not support any options.

## 12735 NAME

12736 ed — edit text

## 12737 SYNOPSIS

12738 ed [-p *string*][-s][*file*]

## 12739 DESCRIPTION

12740 The *ed* utility is a line-oriented text editor that uses two modes: *command mode* and *input mode*.  
 12741 In command mode the input characters shall be interpreted as commands, and in input mode  
 12742 they shall be interpreted as text. See the EXTENDED DESCRIPTION section.

## 12743 OPTIONS

12744 The *ed* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 12745 Utility Syntax Guidelines.

12746 The following options shall be supported:

12747 **-p *string*** Use *string* as the prompt string when in command mode. By default, there shall be  
 12748 no prompt string.

12749 **-s** Suppress the writing of byte counts by **e**, **E**, **r**, and **w** commands and of the **'!'**  
 12750 prompt after a *!command*.

## 12751 OPERANDS

12752 The following operand shall be supported:

12753 ***file*** If the *file* argument is given, *ed* shall simulate an **e** command on the file named by  
 12754 the path name, *file*, before accepting commands from the standard input. If the *file*  
 12755 operand is **'-'**, the results are unspecified.

## 12756 STDIN

12757 The standard input shall be a text file consisting of commands, as described in the EXTENDED  
 12758 DESCRIPTION section.

## 12759 INPUT FILES

12760 The input files shall be text files.

## 12761 ENVIRONMENT VARIABLES

12762 The following environment variables shall affect the execution of *ed*:

12763 **HOME** Determine the path name of the user's home directory.

12764 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 12765 If **LANG** is unset or null, the corresponding value from the implementation-  
 12766 defined default locale shall be used. If any of the internationalization variables  
 12767 contains an invalid setting, the utility shall behave as if none of the variables had  
 12768 been defined.

12769 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 12770 internationalization variables.

12771 **LC\_COLLATE**

12772 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 12773 character collating elements within regular expressions.

12774 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 12775 characters (for example, single-byte as opposed to multi-byte characters in  
 12776 arguments and input files) and the behavior of character classes within regular  
 12777 expressions.



- 12778 **LC\_MESSAGES**  
 12779 Determine the locale that should be used to affect the format and contents of  
 12780 diagnostic messages written to standard error and informative messages written to  
 12781 standard output.
- 12782 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 12783 **ASYNCHRONOUS EVENTS**  
 12784 The *ed* utility shall take the standard action for all signals (see the ASYNCHRONOUS EVENTS  
 12785 section in Section 1.11 (on page 2224)) with the following exceptions:
- 12786 **SIGINT** The *ed* utility shall interrupt its current activity, write the string "?\n" to standard  
 12787 output, and return to command mode (see the EXTENDED DESCRIPTION  
 12788 section).
- 12789 **SIGHUP** If the buffer is not empty and has changed since the last write, the *ed* utility shall  
 12790 attempt to write a copy of the buffer in a file. First, the file named **ed.hup** in the  
 12791 current directory shall be used; if that fails, the file named **ed.hup** in the directory  
 12792 named by the *HOME* environment variable shall be used. In any case, the *ed* utility  
 12793 shall exit without returning to command mode.
- 12794 **SIGQUIT** The *ed* utility shall ignore this event.
- 12795 **STDOUT**  
 12796 Various editing commands and the prompting feature (see **-p**) write to standard output, as  
 12797 described in the EXTENDED DESCRIPTION section.
- 12798 **STDERR**  
 12799 Used only for diagnostic messages.
- 12800 **OUTPUT FILES**  
 12801 The output files shall be text files whose formats are dependent on the editing commands given.
- 12802 **EXTENDED DESCRIPTION**  
 12803 The *ed* utility shall operate on a copy of the file it is editing; changes made to the copy shall have  
 12804 no effect on the file until a **w** (write) command is given. The copy of the text is called the *buffer*.
- 12805 Commands to *ed* have a simple and regular structure: zero, one, or two *addresses* followed by a  
 12806 single-character *command*, possibly followed by parameters to that command. These addresses  
 12807 specify one or more lines in the buffer. Every command that requires addresses has default  
 12808 addresses, so that the addresses very often can be omitted. If the **-p** option is specified, the  
 12809 prompt string shall be written to standard output before each command is read.
- 12810 In general, only one command can appear on a line. Certain commands allow text to be input.  
 12811 This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is said to be  
 12812 in *input mode*. In this mode, no commands shall be recognized; all input is merely collected.  
 12813 Input mode is terminated by entering a line consisting of two characters: a period ('.')  
 12814 followed by a <newline> character. This line is not considered part of the input text.
- 12815 **Regular Expressions in ed**
- 12816 The *ed* utility shall support basic regular expressions, as described in the Base Definitions  
 12817 volume of IEEE Std. 1003.1-200x, Section 9.3, Basic Regular Expressions. Since regular  
 12818 expressions in *ed* are always matched against single lines, never against any larger section of  
 12819 text, there is no way for a regular expression to match a <newline> character. A null RE shall be  
 12820 equivalent to the last RE encountered.
- 12821 Regular expressions are used in addresses to specify lines, and in some commands (for example,  
 12822 the **s** substitute command) to specify portions of a line to be substituted.

12823

**Addresses in ed**

12824

Addressing in *ed* relates to the current line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. If the edit buffer is not empty, the initial value for the current line shall be the last line in the edit buffer; otherwise, zero.

12825

12826

12827

Addresses shall be constructed as follows:

12828

1. The period character ( `' . '` ) shall address the current line.

12829

2. The dollar sign character ( `' $ '` ) shall address the last line of the edit buffer.

12830

3. The positive decimal number *n* shall address the *n*th line of the edit buffer.

12831

4. The apostrophe-x character pair ( `" ' x "` ) shall address the line marked with the mark name character *x*, which shall be a lowercase letter from the portable character set. It shall be an error if the character has not been set to mark a line or if the line that was marked is not currently present in the edit buffer.

12832

12833

12834

12835

5. A BRE enclosed by slash characters ( `' / '` ) shall address the first line found by searching forwards from the line following the current line toward the end of the edit buffer and stopping at the first line containing a string matching the BRE. The BRE consisting of a null BRE delimited by a pair of slash characters shall address the next line containing the last BRE encountered. In addition, the second slash can be omitted at the end of a command line. Within the BRE, a backslash-slash pair ( `" \ / "` ) shall represent a literal slash instead of the BRE delimiter. If necessary, the search shall wrap around to the beginning of the buffer and continue up to and including the current line, so that the entire buffer is searched.

12836

12837

12838

12839

12840

12841

12842

12843

6. A BRE enclosed by question-mark characters ( `' ? '` ) shall address the first line found by searching backwards from the line preceding the current line toward the beginning of the edit buffer and stopping at the first line containing a string matching the BRE. The BRE consisting of a null BRE delimited by a pair of question-mark characters ( `" ?? "` ) shall address the previous line containing the last BRE encountered. In addition, the second question-mark can be omitted at the end of a command line. Within the BRE, a backslash-question-mark pair ( `" \ ? "` ) shall represent a literal question mark instead of the BRE delimiter. If necessary, the search shall wrap around to the end of the buffer and continue up to and including the current line, so that the entire buffer is searched.

12844

12845

12846

12847

12848

12849

12850

12851

12852

7. A plus-sign ( `' + '` ) or hyphen character ( `' - '` ) followed by a decimal number shall address the current line plus or minus the number. A plus-sign or hyphen character not followed by a decimal number shall address the current line plus or minus 1.

12853

12854

12855

Addresses can be followed by zero or more address offsets, optionally <blank>-separated.

12856

Address offsets are constructed as follows:

12857

- A plus-sign or hyphen character followed by a decimal number shall add or subtract, respectively, the indicated number of lines to or from the address. A plus-sign or hyphen character not followed by a decimal number shall add or subtract 1 to or from the address.

12858

12859

12860

- A decimal number shall add the indicated number of lines to the address.

12861

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer. It shall be an error if a search for a BRE fails to find a matching line.

12862

12863

12864

12865

Commands accept zero, one, or two addresses. If more than the required number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first

12866

12867

12868 shall be evaluated and then discarded until the maximum number of valid addresses remain, for  
12869 the specified command.

12870 Addresses shall be separated from each other by a comma (',') or semicolon character (';').  
12871 In the case of a semicolon separator, the current line ('.') shall be set to the first address, and  
12872 only then will the second address be calculated. This feature can be used to determine the  
12873 starting line for forwards and backwards searches; see rules 5. and 6.

12874 Addresses can be omitted on either side of the comma or semicolon separator, in which case the  
12875 resulting address pairs shall be as follows:

12876

| Specified | Resulting   |
|-----------|-------------|
| ,         | 1 , \$      |
| , addr    | 1 ,a ddr    |
| addr ,    | addr , addr |
| ;         | . ; \$      |
| ; addr    | . ; addr    |
| addr ;    | addr ; addr |

12877

12878

12879

12880

12881

12882

12883 Any <blank> characters included between addresses, address separators, or address offsets shall  
12884 be ignored.

12885

### Commands in ed

12886 In the following list of *ed* commands, the default addresses are shown in parentheses. The  
12887 number of addresses shown in the default shall be the number expected by the command. The  
12888 parentheses are not part of the address; they show that the given addresses are the default.

12889 It is generally invalid for more than one command to appear on a line. However, any command  
12890 (except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, and **!**) can be suffixed by the letter **l**, **n**, or **p**; in which case, except for  
12891 the **l**, **n**, and **p** commands, the command shall be executed and then the new current line shall be  
12892 written as described below under the **l**, **n**, and **p** commands. When an **l**, **n**, or **p** suffix is used  
12893 with an **l**, **n**, or **p** command, the command shall write to standard output as described below, but  
12894 it is unspecified whether the suffix writes the current line again in the requested format or  
12895 whether the suffix has no effect. For example, the **pl** command (base **p** command with an **l**  
12896 suffix) shall either write just the current line or write it twice—once as specified for **p** and once  
12897 as specified for **l**. Also, the **g**, **G**, **v**, and **V** commands shall take a command as a parameter.

12898 Each address component can be preceded by zero or more <blank> characters. The command  
12899 letter can be preceded by zero or more <blank> characters. If a suffix letter (**l**, **n**, or **p**) is given,  
12900 the application shall ensure that it immediately follows the command.

12901 The **e**, **E**, **f**, **r**, and **w** commands shall take an optional *file* parameter, separated from the  
12902 command letter by one or more <blank> characters.

12903 If changes have been made in the buffer since the last **w** command that wrote the entire buffer,  
12904 *ed* shall warn the user if an attempt is made to destroy the editor buffer via the **e** or **q** commands.  
12905 The *ed* utility shall write the string:

12906 "?\n"

12907 (followed by an explanatory message if *help mode* has been enabled via the **H** command) to  
12908 standard output and shall continue in command mode with the current line number unchanged.  
12909 If the **e** or **q** command is repeated with no intervening command, it shall take effect.

12910 If a terminal disconnect is detected:

12911 • If the buffer is not empty and has changed since the last write, the *ed* utility shall attempt to  
 12912 write a copy of the buffer to a file named **ed.hup** in the current directory. If this write fails, *ed*  
 12913 shall attempt to write a copy of the buffer to a file name **ed.hup** in the directory named by the  
 12914 *HOME* environment variable. If both these attempts fail, *ed* shall exit without saving the  
 12915 buffer.

12916 • The *ed* utility shall not write the file to the currently remembered path name or return to  
 12917 command mode, and shall terminate with a non-zero exit status.

12918 If an end-of-file is detected on standard input:

12919 • If the *ed* utility is in input mode, *ed* shall terminate input mode and return to command mode.  
 12920 It is unspecified if any partially entered lines (that is, input text without a terminating  
 12921 <newline> character) are discarded from the input text.

12922 • If the *ed* utility is in command mode, it shall act as if a **q** command had been entered.

12923 If the closing delimiter of an RE or of a replacement string (for example, ' / ') in a **g**, **G**, **s**, **v**, or **V**  
 12924 command would be the last character before a <newline> character, that delimiter can be  
 12925 omitted, in which case the addressed line shall be written. For example, the following pairs of  
 12926 commands are equivalent:

12927 *s/s1/s2*     *s/s1/s2/p*

12928 *g/s1*        *g/s1/p*

12929 *?s1*         *?s1?*

12930 If an invalid command is entered, *ed* shall write the string:

12931 "?\n"

12932 (followed by an explanatory message if *help mode* has been enabled via the **H** command) to  
 12933 standard output and shall continue in command mode with the current line number unchanged.

### 12934 **Append Command**

12935 *Synopsis:*     (*.*)*a*  
 12936                 <*text*>  
 12937                 .

12938 The **a** command shall read the given text and append it after the addressed line; the current line  
 12939 number shall become the address of the last inserted line or, if there were none, the addressed  
 12940 line. Address 0 shall be valid for this command; it shall cause the appended text to be placed at  
 12941 the beginning of the buffer.

### 12942 **Change Command**

12943 *Synopsis:*     (*.,.*)*c*  
 12944                 <*text*>  
 12945                 .

12946 The **c** command shall delete the addressed lines, then accept input text that replaces these lines;  
 12947 the current line shall be set to the address of the last line input; or, if there were none, at the line  
 12948 after the last line deleted; if the lines deleted were originally at the end of the buffer, the current  
 12949 line number shall be set to the address of the new last line; if no lines remain in the buffer, the  
 12950 current line number shall be set to zero. Address 0 shall be valid for this command; it shall be  
 12951 interpreted as if address 1 were specified.

12952 **Delete Command**12953 *Synopsis:* (.,.)d

12954 The **d** command shall delete the addressed lines from the buffer. The address of the line after the  
 12955 last line deleted shall become the current line number; if the lines deleted were originally at the  
 12956 end of the buffer, the current line number shall be set to the address of the new last line; if no  
 12957 lines remain in the buffer, the current line number shall be set to zero.

12958 **Edit Command**12959 *Synopsis:* e [*file*]

12960 The **e** command shall delete the entire contents of the buffer and then read in the file named by  
 12961 the path name *file*. The current line number shall be set to the address of the last line of the  
 12962 buffer. If no path name is given, the currently remembered path name, if any, shall be used (see  
 12963 the **f** command). The number of bytes read shall be written to standard output, unless the **-s**  
 12964 option was specified, in the following format:

12965 "%d\n", &lt;number of bytes read&gt;

12966 The name *file* shall be remembered for possible use as a default path name in subsequent **e**, **E**, **r**,  
 12967 and **w** commands. If *file* is replaced by **'!'**, the rest of the line shall be taken to be a shell  
 12968 command line whose output is to be read. Such a shell command line shall not be remembered  
 12969 as the current *file*. All marks shall be discarded upon the completion of a successful **e** command.  
 12970 If the buffer has changed since the last time the entire buffer was written, the user shall be  
 12971 warned, as described previously.

12972 **Edit Without Checking Command**12973 *Synopsis:* E [*file*]

12974 The **E** command shall possess all properties and restrictions of the **e** command except that the  
 12975 editor shall not check to see whether any changes have been made to the buffer since the last **w**  
 12976 command.

12977 **File Name Command**12978 *Synopsis:* f [*file*]

12979 If *file* is given, the **f** command shall change the currently remembered path name to *file*; whether  
 12980 the name is changed or not, it shall then write the (possibly new) currently remembered path  
 12981 name to the standard output in the following format:

12982 "%s\n", &lt;pathname&gt;

12983 The current line number shall be unchanged.

12984 **Global Command**12985 *Synopsis:* (1,\$)g/RE/command list

12986 In the **g** command, the first step shall be to mark every line that matches the given *RE*. Then,  
 12987 going sequentially from the beginning of the file to the end of the file, the given *command list*  
 12988 shall be executed for each marked line, with the current line number set to the address of that  
 12989 line. Any line modified by the *command list* shall be unmarked. When the **g** command completes,  
 12990 the current line number shall have the value assigned by the last command in the *command list*.  
 12991 If there were no matching lines, the current line number shall not be changed. A single command  
 12992 or the first of a list of commands shall appear on the same line as the global command. All lines

12993 of a multi-line list except the last line shall be ended with a backslash; the **a**, **i**, and **c** commands  
 12994 and associated input are permitted. The `'.'` terminating input mode can be omitted if it would  
 12995 be the last line of the *command list*. An empty *command list* shall be equivalent to the **p** command.  
 12996 The use of the **g**, **G**, **v**, **V**, and **!** commands in the *command list* produces undefined results. Any  
 12997 character other than `<space>` or `<newline>` can be used instead of a slash to delimit the *RE*.  
 12998 Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a  
 12999 backslash.

### 13000 **Interactive Global Command**

13001 *Synopsis:* `(1, $)G/RE/`

13002 In the **G** command, the first step shall be to mark every line that matches the given *RE*. Then,  
 13003 for every such line, that line shall be written, the current line number shall be set to the address  
 13004 of that line, and any one command (other than one of the **a**, **c**, **i**, **g**, **G**, **v**, and **V** commands) shall  
 13005 be read and executed. A `<newline>` character shall act as a null command (causing no action to  
 13006 be taken on the current line); an `'&'` shall cause the re-execution of the most recent non-null  
 13007 command executed within the current invocation of **G**. Note that the commands input as part  
 13008 of the execution of the **G** command can address and affect any lines in the buffer. The final value  
 13009 of the current line number shall be the value set by the last command successfully executed.  
 13010 (Note that the last command successfully executed shall be the **G** command itself if a command  
 13011 fails or the null command is specified.) If there were no matching lines, the current line number  
 13012 shall not be changed. The **G** command can be terminated by a SIGINT signal. Any character  
 13013 other than `<space>` or `<newline>` can be used instead of a slash to delimit the *RE* and the  
 13014 replacement. Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is  
 13015 preceded by a backslash.

### 13016 **Help Command**

13017 *Synopsis:* `h`

13018 The **h** command shall write a short message to standard output that explains the reason for the  
 13019 most recent `'?'` notification. The current line number shall be unchanged.

### 13020 **Help-Mode Command**

13021 *Synopsis:* `H`

13022 The **H** command shall cause *ed* to enter a mode in which help messages (see the **h** command)  
 13023 shall be written to standard output for all subsequent `'?'` notifications. The **H** command  
 13024 alternatively shall turn this mode on and off; it is initially off. If the help-mode is being turned  
 13025 on, the **H** command also explains the previous `'?'` notification, if there was one. The current  
 13026 line number shall be unchanged.

### 13027 **Insert Command**

13028 *Synopsis:* `(.)i`  
 13029 `<text>`  
 13030 `.`

13031 The **i** command shall insert the given text before the addressed line; the current line is set to the  
 13032 last inserted line or, if there was none, to the addressed line. This command differs from the **a**  
 13033 command only in the placement of the input text. Address 0 shall be valid for this command; it  
 13034 shall be interpreted as if address 1 were specified.

13035 **Join Command**13036 *Synopsis:* ( . , .+1 ) j

13037 The **j** command shall join contiguous lines by removing the appropriate <newline> characters. If  
 13038 exactly one address is given, this command shall do nothing. If lines are joined, the current line  
 13039 number shall be set to the address of the joined line; otherwise, the current line number shall be  
 13040 unchanged.

13041 **Mark Command**13042 *Synopsis:* ( . ) k x

13043 The **k** command shall mark the addressed line with name *x*, which the application shall ensure is  
 13044 a lowercase letter from the portable character set. The address " *x* " shall then refer to this line;  
 13045 the current line number shall be unchanged.

13046 **List Command**13047 *Synopsis:* ( . , . ) l

13048 The **l** command shall write to standard output the addressed lines in a visually unambiguous  
 13049 form. The characters listed in the Base Definitions volume of IEEE Std. 1003.1-200x, Table 5-1,  
 13050 Escape Sequences and Associated Actions ( '\ ', '\a', '\b', '\f', '\r', '\t', '\v' ) shall  
 13051 be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-  
 13052 printable characters not in the table shall be written as one three-digit octal number (with a  
 13053 preceding backslash character) for each byte in the character (most significant byte first). If the  
 13054 size of a byte on the system is greater than nine bits, the format used for non-printable characters  
 13055 is implementation-defined.

13056 Long lines shall be folded, with the point of folding indicated by writing backslash/<newline>  
 13057 character; the length at which folding occurs is unspecified, but should be appropriate for the  
 13058 output device. The end of each line shall be marked with a '\$', and '\$' characters within the  
 13059 text shall be written with a preceding backslash. An **l** command can be appended to any other  
 13060 command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. The current line number shall be set to the address of  
 13061 the last line written.

13062 **Move Command**13063 *Synopsis:* ( . , . ) *address*

13064 The **m** command shall reposition the addressed lines after the line addressed by *address*.  
 13065 Address 0 shall be valid for *address* and cause the addressed lines to be moved to the beginning  
 13066 of the buffer. It shall be an error if *address* falls within the range of moved lines. The  
 13067 current line number shall be set to the address of the last line moved.

13068 **Number Command**13069 *Synopsis:* ( . , . ) n

13070 The **n** command shall write to standard output the addressed lines, preceding each line by its  
 13071 line number and a <tab> character; the current line number shall be set to the address of the last  
 13072 line written. The **n** command can be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

13073 **Print Command**13074 *Synopsis:* (.,.)p

13075 The **p** command shall write to standard output the addressed lines; the current line number shall  
 13076 be set to the address of the last line written. The **p** command can be appended to any command  
 13077 other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

13078 **Prompt Command**13079 *Synopsis:* P

13080 The **P** command shall cause *ed* to prompt with an asterisk ( '\*' ) (or *string*, if **-p** is specified) for  
 13081 all subsequent commands. The **P** command alternatively shall turn this mode on and off; it shall  
 13082 be initially on if the **-p** option is specified; otherwise, off. The current line number shall be  
 13083 unchanged.

13084 **Quit Command**13085 *Synopsis:* q

13086 The **q** command shall cause *ed* to exit. If the buffer has changed since the last time the entire  
 13087 buffer was written, the user shall be warned, as described previously.

13088 **Quit Without Checking Command**13089 *Synopsis:* Q

13090 The **Q** command shall cause *ed* to exit without checking whether changes have been made in the  
 13091 buffer since the last **w** command.

13092 **Read Command**13093 *Synopsis:* (\$)r [*file*]

13094 The **r** command shall read in the file named by the path name *file* and append it after the  
 13095 addressed line. If no *file* argument is given, the currently remembered path name, if any, shall be  
 13096 used (see the **e** and **f** commands). The currently remembered path name shall not be changed  
 13097 unless there is no remembered path name. Address 0 shall be valid for **r** and shall cause the file  
 13098 to be read at the beginning of the buffer. If the read is successful, and **-s** was not specified, the  
 13099 number of bytes read shall be written to standard output in the following format:

13100 "%d\n", &lt;number of bytes read&gt;

13101 The current line number shall be set to the address of the last line read in. If *file* is replaced by  
 13102 '!', the rest of the line shall be taken to be a shell command line whose output is to be read.  
 13103 Such a shell command line shall not be remembered as the current path name.

13104 **Substitute Command**13105 *Synopsis:* (.,.)s/RE/replacement/flags

13106 The **s** command shall search each addressed line for an occurrence of the specified *RE* and  
 13107 replace either the first or all (non-overlapped) matched strings with the *replacement*; see the  
 13108 following description of the **g** suffix. It is an error if the substitution fails on every addressed  
 13109 line. Any character other than <space> or <newline> can be used instead of a slash to delimit the  
 13110 *RE* and the replacement. Within the *RE*, the *RE* delimiter itself can be used as a literal character if  
 13111 it is preceded by a backslash. The current line shall be set to the address of the last line on which  
 13112 a substitution occurred.



13113 An ampersand ('&') appearing in the *replacement* shall be replaced by the string matching the  
 13114 RE on the current line. The special meaning of '&' in this context can be suppressed by  
 13115 preceding it by backslash. As a more general feature, the characters '\n', where *n* is a digit,  
 13116 shall be replaced by the text matched by the corresponding back-reference expression. When the  
 13117 character '%' is the only character in the *replacement*, the *replacement* used in the most recent  
 13118 substitute command shall be used as the *replacement* in the current substitute command; if there  
 13119 was no previous substitute command, the use of '%' in this manner shall be an error. The '%'  
 13120 shall lose its special meaning when it is in a replacement string of more than one character or is  
 13121 preceded by a backslash. For each backslash ('\') encountered in scanning *replacement* from  
 13122 beginning to end, the following character shall lose its special meaning (if any). It is unspecified  
 13123 what special meaning is given to any character other than '&', '\', '%', or digits.

13124 A line can be split by substituting a <newline> character into it. The application shall ensure it  
 13125 escapes the <newline> character in the *replacement* by preceding it by backslash. Such  
 13126 substitution cannot be done as part of a **g** or **v** *command list*. The current line number shall be set  
 13127 to the address of the last line on which a substitution is performed. If no substitution is  
 13128 performed, the current line number shall be unchanged. If a line is split, a substitution shall be  
 13129 considered to have been performed on each of the new lines for the purpose of determining the  
 13130 new current line number. A substitution shall be considered to have been performed even if the  
 13131 replacement string is identical to the string that it replaces.

13132 The application shall ensure that the value of *flags* is zero or more of:

- 13133 *count* Substitute for the *count*th occurrence only of the *RE* found on each addressed line.
- 13134 **g** Globally substitute for all non-overlapping instances of the *RE* rather than just the first  
 13135 one. If both **g** and *count* are specified, the results are unspecified.
- 13136 **l** Write to standard output the final line in which a substitution was made. The line shall  
 13137 be written in the format specified for the **l** command.
- 13138 **n** Write to standard output the final line in which a substitution was made. The line shall  
 13139 be written in the format specified for the **n** command.
- 13140 **p** Write to standard output the final line in which a substitution was made. The line shall  
 13141 be written in the format specified for the **p** command.

## 13142 Copy Command

13143 *Synopsis:* (.,.)*taddress*

13144 The **t** command shall be equivalent to the **m** command, except that a copy of the addressed lines  
 13145 shall be placed after address *address* (which can be 0); the current line number shall be set to the  
 13146 address of the last line added.

## 13147 Undo Command

13148 *Synopsis:* u

13149 The **u** command shall nullify the effect of the most recent command that modified anything in  
 13150 the buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, **G**, or **V** command. All changes  
 13151 made to the buffer by a **g**, **G**, **v**, or **V** global command shall be undone as a single change; if no  
 13152 changes were made by the global command (such as with **g/RE/p**), the **u** command shall have  
 13153 no effect. The current line number shall be set to the value it had immediately before the  
 13154 command being undone started.

13155 **Global Non-Matched Command**13156 *Synopsis:* (1,\$)v/RE/command list13157 This command shall be equivalent to the global command **g** except that the lines that are marked  
13158 during the first step shall be those that do not match the *RE*.13159 **Interactive Global Not-Matched Command**13160 *Synopsis:* (1,\$)V/RE/13161 This command shall be equivalent to the interactive global command **G** except that the lines that  
13162 are marked during the first step shall be those that do not match the *RE*.13163 **Write Command**13164 *Synopsis:* (1,\$)w [file]13165 The **w** command shall write the addressed lines into the file named by the path name *file*. The  
13166 command shall create the file, if it does not exist, or shall replace the contents of the existing file.  
13167 The currently remembered path name shall not be changed unless there is no remembered path  
13168 name. If no path name is given, the currently remembered path name, if any, shall be used (see  
13169 the **e** and **f** commands); the current line number shall be unchanged. If the command is  
13170 successful, the number of bytes written shall be written to standard output, unless the **-s** option  
13171 was specified, in the following format:

13172 "%d\n", &lt;number of bytes written&gt;

13173 If *file* begins with **'!'**, the rest of the line shall be taken to be a shell command line whose  
13174 standard input shall be the addressed lines. Such a shell command line shall not be remembered  
13175 as the current path name. This usage of the write command with **'!'** shall not be considered as  
13176 a "last **w** command that wrote the entire buffer", as described previously; thus, this alone shall  
13177 not prevent the warning to the user if an attempt is made to destroy the editor buffer via the **e** or  
13178 **q** commands.13179 **Line Number Command**13180 *Synopsis:* (\$)=13181 The line number of the addressed line shall be written to standard output in the following  
13182 format:

13183 "%d\n", &lt;line number&gt;

13184 The current line number shall be unchanged by this command.

13185 **Shell Escape Command**13186 *Synopsis:* !command13187 The remainder of the line after the **'!'** shall be sent to the command interpreter to be  
13188 interpreted as a shell command line. Within the text of that shell command line, the unescaped  
13189 character **'%'** shall be replaced with the remembered path name; if a **'!'** appears as the first  
13190 character of the command, it shall be replaced with the text of the previous shell command  
13191 executed via **'!'**. Thus, **"!!"** shall repeat the previous *!command*. If any replacements of **'%'** or  
13192 **'!'** are performed, the modified line shall be written to the standard output before *command* is  
13193 executed. The **'!'** command shall write:

13194 "!\n"

13195 to standard output upon completion, unless the `-s` option is specified. The current line number  
13196 shall be unchanged.

### 13197 **Null Command**

13198 *Synopsis:* ( `+.1` )

13199 An address alone on a line shall cause the addressed line to be written. A `<newline>` character  
13200 alone shall be equivalent to `"+.1p"`. The current line number shall be set to the address of the  
13201 written line.

### 13202 **EXIT STATUS**

13203 The following exit values shall be returned:

13204 0 Successful completion without any file or command errors.

13205 >0 An error occurred.

### 13206 **CONSEQUENCES OF ERRORS**

13207 When an error in the input script is encountered, or when an error is detected that is a  
13208 consequence of the data (not) present in the file or due to an external condition such as a read or  
13209 write error:

13210 • If the standard input is a terminal device file, all input shall be flushed, and a new command  
13211 read.

13212 • If the standard input is a regular file, *ed* shall terminate with a non-zero exit status.

### 13213 **APPLICATION USAGE**

13214 Because of the extremely terse nature of the default error messages, the prudent script writer  
13215 begins the *ed* input commands with an **H** command, so that if any errors do occur at least some  
13216 clue as to the cause is made available.

13217 In previous versions, an obsolescent `-` option was described. This is no longer specified.  
13218 Applications should use the `-s` option. Using `-` as a *file* operand now produces unspecified  
13219 results. This allows implementations to continue to support the former required behavior.

### 13220 **EXAMPLES**

13221 None.

### 13222 **RATIONALE**

13223 The initial description of this utility was adapted from the SVID. It contains some features not  
13224 found in Version 7 or BSD-derived systems. Some of the differences between the POSIX and  
13225 BSD *ed* utilities include, but need not be limited to:

13226 • The BSD `-` option does not suppress the `'!'` prompt after a `!` command.

13227 • BSD does not support the special meanings of the `'%'` and `'!'` characters within a `!`  
13228 command.

13229 • BSD does not support the *addresses* `' ; '` and `' , '`.

13230 • BSD allows the command/suffix pairs **pp**, **ll**, and so on, which are unspecified in this volume  
13231 of IEEE Std. 1003.1-200x.

13232 • BSD does not support the `'!'` character part of the **e**, **r**, or **w** commands.

13233 • A failed **g** command in BSD sets the line number to the last line searched if there are no  
13234 matches.

13235 • BSD does not default the *command list* to the **p** command.

- 13236 • BSD does not support the **G**, **h**, **H**, **n**, or **V** commands.
  - 13237 • On BSD, if there is no inserted text, the insert command changes the current line to the
  - 13238 referenced line `-1`; that is, the line before the specified line.
  - 13239 • On BSD, the *join* command with only a single address changes the current line to that
  - 13240 address.
  - 13241 • BSD does not support the **P** command; moreover, in BSD it is synonymous with the **p**
  - 13242 command.
  - 13243 • BSD does not support the *undo* of the commands **j**, **m**, **r**, **s**, or **t**.
  - 13244 • The Version 7 *ed* command **W**, and the BSD *ed* commands **W**, **wq**, and **z** are not present in this
  - 13245 volume of IEEE Std. 1003.1-200x.
- 13246 The `-s` option was added to allow the functionality of the now withdrawn `-` option in a manner
- 13247 compatible with the Utility Syntax Guidelines.
- 13248 In early proposals there was a limit, `{ED_FILE_MAX}`, that described the historical limitations of
- 13249 some *ed* utilities in their handling of large files; some of these have had problems with files larger
- 13250 than 100 000 bytes. It was this limitation that prompted much of the desire to include a *split*
- 13251 command in this volume of IEEE Std. 1003.1-200x. Since this limit was removed, this volume of
- 13252 IEEE Std. 1003.1-200x requires that implementations document the file size limits imposed by *ed*
- 13253 in the conformance document. The limit `{ED_LINE_MAX}` was also removed; therefore, the
- 13254 global limit `{LINE_MAX}` is used for input and output lines.
- 13255 The manner in which the **l** command writes non-printable characters was changed to avoid the
- 13256 historical backspace-overstrike method. On video display terminals, the overstrike is ambiguous
- 13257 because most terminals simply replace overstruck characters, making the **l** format not useful for
- 13258 its intended purpose of unambiguously understanding the content of the line. The historical
- 13259 backslash escapes were also ambiguous. (The string `"a\0011"` could represent a line containing
- 13260 those six characters or a line containing the three characters `'a'`, a byte with a binary value of 1,
- 13261 and a 1.) In the format required here, a backslash appearing in the line is written as `"\\"` so that
- 13262 the output is truly unambiguous. The method of marking the ends of lines was adopted from the
- 13263 *ex* editor and is required for any line ending in `<space>s`; the `'$'` is placed on all lines so that a
- 13264 real `'$'` at the end of a line cannot be misinterpreted.
- 13265 Systems with bytes too large to fit into three octal digits must devise other means of displaying
- 13266 non-printable characters. Consideration was given to requiring that the number of octal digits be
- 13267 large enough to hold a byte, but this seemed to be too confusing for applications on the vast
- 13268 majority of systems where three digits are adequate. It would be theoretically possible for the
- 13269 application to use the *getconf* utility to find out the `CHAR_BIT` value and deal with such an
- 13270 algorithm; however, there is really no portable way that an application can use the octal values
- 13271 of the bytes across various coded character sets, so the additional specification was not
- 13272 worthwhile.
- 13273 The description of how a NUL is written was removed. The NUL character cannot be in text
- 13274 files, and this volume of IEEE Std. 1003.1-200x should not dictate behavior in the case of
- 13275 undefined, erroneous input.
- 13276 Unlike some of the other editing utilities, the file names accepted by the **E**, **e**, **R**, and **r** commands
- 13277 are not patterns.
- 13278 Early proposals stated that the `-p` option worked only when standard input was associated with
- 13279 a terminal device. This has been changed to conform to historical implementations, thereby
- 13280 allowing applications to interpose themselves between a user and the *ed* utility.

- 13281 The form of the substitute command that uses the **n** suffix was limited in some historical  
13282 documentation (where this was described incorrectly as “backreferencing”). This limit has been  
13283 omitted because there is no reason an editor processing lines of {LINE\_MAX} length should have  
13284 this restriction. The command **s/x/X/2 047** should be able to substitute the 2 047th occurrence of **x**  
13285 on a line.
- 13286 The use of printing commands with printing suffixes (such as **pn**, **lp**, and so on) was made  
13287 unspecified because BSD-based systems allow this, whereas System V does not.
- 13288 Some BSD-based systems exit immediately upon receipt of end-of-file if all of the lines in the file  
13289 have been deleted. Since this volume of IEEE Std. 1003.1-200x refers to the **q** command in this  
13290 instance, such behavior is not allowed.
- 13291 Some historical implementations returned exit status zero even if command errors had occurred;  
13292 this is not allowed by this volume of IEEE Std. 1003.1-200x.
- 13293 Some historical implementations contained a bug that allowed a single period to be entered in  
13294 input mode as <backslash> <period> <newline>. This is not allowed by the *ed* because there is  
13295 no description of escaping any of the characters in input mode; backslashes are entered into the  
13296 buffer exactly as typed. The typical method of entering a single period has been to precede it  
13297 with another character and then use the substitute command to delete that character.
- 13298 It is difficult under some modes of some versions of historical operating system terminal drivers  
13299 to distinguish between an end-of-file condition and terminal disconnect. The ISO POSIX-2  
13300 standard does not require implementations to distinguish between the two situations, which  
13301 permits historical implementations of the *ed* utility on historical platforms to conform.  
13302 Implementations are encouraged to distinguish between the two, if possible, and take  
13303 appropriate action on terminal disconnect.
- 13304 Historically, *ed* accepted a zero address for the **a** and **r** commands in order to insert text at the  
13305 start of the edit buffer. When the buffer was empty the command **.=** returned zero.  
13306 IEEE Std. 1003.1-200x requires conformance to historical practice.
- 13307 For consistency with the **a** and **r** commands and better user functionality, the **i** and **c** commands  
13308 must also accept an address of 0, in which case **0i** is treated as **1i** and likewise for the **c**  
13309 command.
- 13310 All of the following are valid addresses:
- 13311 **+++** Three lines after the current line.
- 13312 **/pattern/-** One line before the next occurrence of pattern.
- 13313 **-2** Two lines before the current line.
- 13314 **3 ——— 2** Line one (note the intermediate negative address).
- 13315 **1 2 3** Line six.
- 13316 Any number of addresses can be provided to commands taking addresses; for example,  
13317 "**1,2,3,4,5p**" prints lines 4 and 5, because two is the greatest valid number of addresses  
13318 accepted by the **print** command. This, in combination with the semicolon delimiter, permits  
13319 users to create commands based on ordered patterns in the file. For example, the command  
13320 "**3;/foo/;+2p**" will display the first line after line 3 that contains the pattern *foo*, plus the next  
13321 two lines. Note that the address "**3;**" must still be evaluated before being discarded, because  
13322 the search origin for the **/foo/** command depends on this.
- 13323 Historically, *ed* disallowed address chains, as discussed above, consisting solely of comma or  
13324 semicolon separators; for example, "**,, ,**" or "**;; ;**" were considered an error. For consistency of  
13325 address specification, this restriction is removed. The following table lists some of the address

13326 forms now possible:

|       | Address | Addr1 | Addr2 | Status     | Comment               |
|-------|---------|-------|-------|------------|-----------------------|
| 13327 | 7,      | 7     | 7     | Historical |                       |
| 13328 | 7,5,    | 5     | 5     | Historical |                       |
| 13329 | 7,5,9   | 5     | 9     | Historical |                       |
| 13330 | 7,9     | 7     | 9     | Historical |                       |
| 13331 | 7,+     | 7     | 8     | Historical |                       |
| 13332 | ,       | 1     | \$    | Historical |                       |
| 13333 | ,7      | 1     | 7     | Extension  |                       |
| 13334 | ,,      | \$    | \$    | Extension  |                       |
| 13335 | ,;      | \$    | \$    | Extension  |                       |
| 13336 | 7;      | 7     | 7     | Historical |                       |
| 13337 | 7;5;    | 5     | 5     | Historical |                       |
| 13338 | 7;5;9   | 5     | 9     | Historical |                       |
| 13339 | 7;5,9   | 5     | 9     | Historical |                       |
| 13340 | 7;\$;4  | \$    | 4     | Historical | Valid, but erroneous. |
| 13341 | 7;9     | 7     | 9     | Historical |                       |
| 13342 | 7;+     | 7     | 8     | Historical |                       |
| 13343 | ;       | .     | \$    | Historical |                       |
| 13344 | ;7      | .     | 7     | Extension  |                       |
| 13345 | ;;      | \$    | \$    | Extension  |                       |
| 13346 | ;,      | \$    | \$    | Extension  |                       |

13348 Historically, values could be added to addresses by including them after one or more <blank>  
 13349 characters; for example, "3 - 5p" wrote the seventh line of the file, and "/foo/ 5" was the  
 13350 same as "5 /foo/". However, only absolute values could be added; for example, "5 /foo/"  
 13351 was an error. IEEE Std. 1003.1-200x requires conformance to historical practice.

13352 Historically, *ed* accepted the '^' character as an address, in which case it was identical to the  
 13353 hyphen character. IEEE Std. 1003.1-200x does not require or prohibit this behavior.

#### 13354 FUTURE DIRECTIONS

13355 None.

#### 13356 SEE ALSO

13357 *ex, sed, sh, vi*

#### 13358 CHANGE HISTORY

13359 First released in Issue 2.

#### 13360 Issue 4

13361 Aligned with the ISO/IEC 9945-2:1993 standard.

#### 13362 Issue 5

13363 In the OPTIONS section, the meaning of -s and - is clarified.

13364 Second FUTURE DIRECTION added.

#### 13365 Issue 6

13366 The obsolescent single-minus form has been removed.

13367 A second APPLICATION USAGE note has been added.

13368 The Open Group corrigenda item U025/2 has been applied, correcting the description of the Edit  
 13369 section.

- 13370 The *ed* utility is updated to align with the IEEE P1003.2b draft standard. This includes addition of  
13371 the treatment of the SIGQUIT signal, changes to *ed* addressing, changes to processing when  
13372 end-of-file is detected and when terminal disconnect is detected.
- 13373 The normative text is reworded to avoid use of the term “must” for application requirements.

13374 **NAME**

13375 env — set the environment for command invocation

13376 **SYNOPSIS**

13377 env [-i][name=value]... [utility [argument...]]

13378 **DESCRIPTION**13379 The *env* utility shall obtain the current environment, modify it according to its arguments, then  
13380 invoke the utility named by the *utility* operand with the modified environment.13381 Optional arguments shall be passed to *utility*.13382 If no *utility* operand is specified, the resulting environment shall be written to the standard  
13383 output, with one *name=value* pair per line.13384 **OPTIONS**13385 The *env* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
13386 12.2, Utility Syntax Guidelines.

13387 The following options shall be supported:

13388 **-i** Invoke *utility* with exactly the environment specified by the arguments; the  
13389 inherited environment shall be ignored completely.13390 **OPERANDS**

13391 The following operands shall be supported:

13392 *name=value* Arguments of the form *name=value* shall modify the execution environment, and  
13393 shall be placed into the inherited environment before the *utility* is invoked.13394 *utility* The name of the utility to be invoked. If the *utility* operand names any of the  
13395 special built-in utilities in Section 2.15 (on page 2276), the results are undefined.13396 *argument* A string to pass as an argument for the invoked utility.13397 **STDIN**

13398 Not used.

13399 **INPUT FILES**

13400 None.

13401 **ENVIRONMENT VARIABLES**13402 The following environment variables shall affect the execution of *env*:13403 *LANG* Provide a default value for the internationalization variables that are unset or null.  
13404 If *LANG* is unset or null, the corresponding value from the implementation-  
13405 defined default locale shall be used. If any of the internationalization variables  
13406 contains an invalid setting, the utility shall behave as if none of the variables had  
13407 been defined.13408 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
13409 internationalization variables.13410 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
13411 characters (for example, single-byte as opposed to multi-byte characters in  
13412 arguments).13413 *LC\_MESSAGES*13414 Determine the locale that should be used to affect the format and contents of  
13415 diagnostic messages written to standard error.



- 13416 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 13417 **PATH** Determine the location of the *utility*, as described in the Base Definitions volume of  
 13418 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables. If *PATH* is specified as a  
 13419 *name=value* operand to *env*, the *value* given shall be used in the search for *utility*.
- 13420 **ASYNCHRONOUS EVENTS**
- 13421 Default.
- 13422 **STDOUT**
- 13423 If no *utility* operand is specified, each *name=value* pair in the resulting environment shall be  
 13424 written in the form:
- 13425 "%s=%s\n", <name>, <value>
- 13426 If the *utility* operand is specified, the *env* utility shall not write to standard output.
- 13427 **STDERR**
- 13428 Used only for diagnostic messages.
- 13429 **OUTPUT FILES**
- 13430 None.
- 13431 **EXTENDED DESCRIPTION**
- 13432 None.
- 13433 **EXIT STATUS**
- 13434 If the *utility* utility is invoked, the exit status of *env* shall be the exit status of *utility*; otherwise,  
 13435 the *env* utility shall exit with one of the following values:
- 13436 0 The *env* utility completed successfully.
- 13437 1–125 An error occurred in the *env* utility.
- 13438 126 The utility specified by *utility* was found but could not be invoked.
- 13439 127 The utility specified by *utility* could not be found.
- 13440 **CONSEQUENCES OF ERRORS**
- 13441 Default.
- 13442 **APPLICATION USAGE**
- 13443 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
 13444 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
 13445 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
 13446 used for other meanings; most utilities use small values for “normal error conditions” and the  
 13447 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
 13448 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
 13449 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
 13450 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
 13451 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
 13452 any other reason.
- 13453 Historical implementations of the *env* utility use the *execvp()* or *execlp()* functions defined in the  
 13454 System Interfaces volume of IEEE Std. 1003.1-200x to invoke the specified utility; this provides  
 13455 better performance and keeps users from having to escape characters with special meaning to  
 13456 the shell. Therefore, shell functions, special built-ins, and built-ins that are only provided by the  
 13457 shell are not found.

13458 **EXAMPLES**

13459           The following command:

```
13460 env -i PATH=/mybin mygrep xyz myfile
```

13461           invokes the command *mygrep* with a new *PATH* value as the only entry in its environment. In  
13462           this case, *PATH* is used to locate *mygrep*, which then must reside in **/mybin**.

13463 **RATIONALE**

13464           As with all other utilities that invoke other utilities, this volume of IEEE Std. 1003.1-200x only  
13465           specifies what *env* does with standard input, standard output, standard error, input files, and  
13466           output files. If a utility is executed, it is not constrained by the specification of input and output  
13467           by *env*.

13468           The **-i** option was added to allow the functionality of the withdrawn **-** option in a manner  
13469           compatible with the Utility Syntax Guidelines.

13470           Some have suggested that *env* is redundant since the same effect is achieved by:

```
13471 name=value ... utility [argument ...]
```

13472           The example is equivalent to *env* when an environment variable is being added to the  
13473           environment of the command, but not when the environment is being set to the given value.

13474           The *env* utility also writes out the current environment if invoked without arguments. There is  
13475           sufficient functionality beyond what the example provides to justify inclusion of *env*.

13476 **FUTURE DIRECTIONS**

13477           None.

13478 **SEE ALSO**

13479           Section 2.5 (on page 2241)

13480 **CHANGE HISTORY**

13481           First released in Issue 2.

13482 **Issue 4**

13483           Aligned with the ISO/IEC 9945-2:1993 standard.

## 13484 NAME

13485 ex — text editor

## 13486 SYNOPSIS

13487 UP `ex [-rR][-l][-s | -v][-c command]-t tagstring[-w size][file ...]`

13488

## 13489 DESCRIPTION

13490 The *ex* utility is a line-oriented text editor. There are two other modes of the editor—open and  
 13491 visual—in which screen-oriented editing is available. This is described more fully by the *ex open*  
 13492 and *visual* commands and in *vi*.

13493 This section uses the term *edit buffer* to describe the current working text. No specific  
 13494 implementation is implied by this term. All editing changes are performed on the edit buffer,  
 13495 and no changes to it shall affect any file until an editor command writes the file.

13496 Certain terminals do not have all the capabilities necessary to support the complete *ex* definition,  
 13497 such as the full-screen editing commands (*visual mode* or *open mode*). When these commands  
 13498 cannot be supported on such terminals, this condition shall not produce an error message such  
 13499 as “not an editor command” or report a syntax error. The implementation may either accept the  
 13500 commands and produce results on the screen that are the result of an unsuccessful attempt to  
 13501 meet the requirements of this volume of IEEE Std. 1003.1-200x or report an error describing the  
 13502 terminal-related deficiency.

## 13503 OPTIONS

13504 The *ex* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 13505 Utility Syntax Guidelines.

13506 The following options shall be supported:

13507 **-c *command*** Specify an initial command to be executed in the first edit buffer loaded from an  
 13508 existing file (see the EXTENDED DESCRIPTION section). Implementations may  
 13509 support more than a single **-c** option. In such implementations, the specified  
 13510 commands shall be executed in the order specified on the command line.

13511 **-l** (The letter ell.) Set lisp mode; indents appropriately for LISP code; the **O**, **{**, **[**, and  
 13512 **]** commands in visual mode are modified to have meaning for LISP.

13513 **-r** Recover the named files (see the EXTENDED DESCRIPTION section). Recovery  
 13514 information for a file shall be saved during an editor or system crash (for example,  
 13515 when the editor is terminated by a signal which the editor can catch), or after the  
 13516 use of an *ex preserve* command.

13517 A *crash* in this context is an unexpected failure of the system or utility that requires  
 13518 restarting the failed system or utility. A system crash implies that any utilities  
 13519 running at the time also crash. In the case of an editor or system crash, the number  
 13520 of changes to the edit buffer (since the most recent *preserve* command) that will be  
 13521 recovered is unspecified.

13522 If no *file* operands are given and the **-t** option is not specified, all other options, the  
 13523 *EXINIT* variable, and any *.exrc* files shall be ignored; a list of all recoverable files  
 13524 available to the invoking user shall be written, and the editor shall exit normally  
 13525 without further action.

13526 **-R** Set **readonly** edit option.

13527 **-s** Prepare *ex* for batch use by taking the following actions:

- 13528                   • Suppress writing prompts and informational (but not diagnostic) messages.
- 13529                   • Ignore the value of *TERM* and any implementation default terminal type and
- 13530                   assume the terminal is a type incapable of supporting open or visual modes;
- 13531                   see the **visual** command and the description of *vi*.
- 13532                   • Suppress the use of the *EXINIT* environment variable and the reading of any
- 13533                   *.exrc* file; see the EXTENDED DESCRIPTION section.
- 13534                   • Suppress autoindentation, ignoring the value of the **autoindent** edit option.
- 13535            **-t tagstring** Edit the file containing the specified *tagstring*; see *ctags*. The tags feature
- 13536                   represented by **-t tagstring** and the **tag** command is optional. It shall be provided
- 13537                   on any system that also provides a conforming implementation of *ctags*; otherwise,
- 13538                   the use of **-t** produces undefined results. On any system, it shall be an error to
- 13539                   specify more than a single **-t** option.
- 13540            **-v**            Begin in visual mode (see *vi*).
- 13541            **-w size**       Set the value of the *window* editor option to *size*.

#### 13542 OPERANDS

13543            The following operand shall be supported:

13544            *file*            A path name of a file to be edited.

#### 13545 STDIN

13546            The standard input consists of a series of commands and input text, as described in the

13547            EXTENDED DESCRIPTION section. The implementation may limit each line of standard input

13548            to a length of {LINE\_MAX}.

13549            If the standard input is not a terminal device, it shall be as if the **-s** option had been specified.

13550            If a read from the standard input returns an error, or if the editor detects an end-of-file condition

13551            from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

#### 13552 INPUT FILES

13553            Input files shall be text files or files that would be text files except for an incomplete last line that

13554            is not longer than {LINE\_MAX}-1 bytes in length and contains no NUL characters. By default,

13555            any incomplete last line shall be treated as if it had a trailing <newline> character. The editing of

13556            other forms of files may optionally be allowed by *ex* implementations.

13557            The *.exrc* files and source files shall be text files consisting of *ex* commands; see the EXTENDED

13558            DESCRIPTION section.

13559            By default, the editor shall read lines from the files to be edited without interpreting any of those

13560            lines as any form of editor command.

#### 13561 ENVIRONMENT VARIABLES

13562            The following environment variables shall affect the execution of *ex*:

13563            **COLUMNS**    Override the system-selected horizontal screen size. See the Base Definitions |

13564                            volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid |

13565                            values and results when it is unset or null.

13566            **EXINIT**       Determine a list of *ex* commands that are executed on editor start-up. See the

13567                            EXTENDED DESCRIPTION section for more details of the initialization phase.

13568            **HOME**         Determine a path name of a directory that shall be searched for an editor start-up

13569                            file named *.exrc*; see the EXTENDED DESCRIPTION section.

- 13570        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 13571        If *LANG* is unset or null, the corresponding value from the implementation-  
 13572        defined default locale shall be used. If any of the internationalization variables  
 13573        contains an invalid setting, the utility shall behave as if none of the variables had  
 13574        been defined.
- 13575        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 13576        internationalization variables.
- 13577        *LC\_COLLATE*    Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 13578        character collating elements within regular expressions.  
 13579
- 13580        *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
 13581        characters (for example, single-byte as opposed to multi-byte characters in  
 13582        arguments and input files), the behavior of character classes within regular  
 13583        expressions, the classification of characters as uppercase or lowercase letters, the  
 13584        case conversion of letters, and the detection of word boundaries.
- 13585        *LC\_MESSAGES*    Determine the locale that should be used to affect the format and contents of  
 13586        diagnostic messages written to standard error.  
 13587
- 13588        *LINES*        Override the system-selected vertical screen size, used as the number of lines in a  
 13589        screenful and the vertical screen size in visual mode. See the Base Definitions  
 13590        volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid  
 13591        values and results when it is unset or null.
- 13592 XSI        *NLSPATH*       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 13593        *PATH*        Determine the search path for the shell command specified in the *ex* editor  
 13594        commands **!**, **shell**, **read**, and **write**, and the open and visual mode command **!**; see  
 13595        the description of command search and execution in Section 2.9.1.1 (on page 2257).
- 13596        *SHELL*        Determine the preferred command line interpreter for use as the default value of  
 13597        the **shell** edit option.
- 13598        *TERM*        Determine the name of the terminal type. If this variable is unset or null, an  
 13599        unspecified default terminal type shall be used.
- 13600 **ASYNCHRONOUS EVENTS**
- 13601        The following term is used in this and following sections to specify command and asynchronous  
 13602        event actions:
- 13603        *complete write*
- 13604        A complete write is a write of the entire contents of the edit buffer to a file of a type  
 13605        other than a terminal device, or the saving of the edit buffer caused by the user  
 13606        executing the *ex* **preserve** command. Writing the contents of the edit buffer to a  
 13607        temporary file that will be removed when the editor exits shall not be considered a  
 13608        complete write.
- 13609        The following actions shall be taken upon receipt of signals:
- 13610        *SIGINT*        If the standard input is not a terminal device, *ex* shall not write the file or return to  
 13611        command or text input mode, and shall exit with a non-zero exit status.
- 13612        Otherwise, if executing an open or visual text input mode command, *ex* in receipt  
 13613        of *SIGINT* shall behave identically to its receipt of the <ESC> character.

- 13614                   Otherwise:
- 13615                   1. If executing an *ex* text input mode command, all input lines that have been  
13616                   completely entered shall be resolved into the edit buffer, and any partially  
13617                   entered line shall be discarded.
- 13618                   2. If there is a currently executing command, it shall be aborted and a message  
13619                   displayed. Unless otherwise specified by the *ex* or *vi* command descriptions,  
13620                   it is unspecified whether any lines modified by the executing command  
13621                   appear modified, or as they were before being modified by the executing  
13622                   command, in the buffer.
- 13623                   If the currently executing command was a motion command, its associated  
13624                   command shall be discarded.
- 13625                   3. If in open or visual command mode, the terminal shall be alerted.
- 13626                   4. The editor shall then return to command mode.
- 13627           SIGCONT   The screen shall be refreshed if in open or visual mode.
- 13628           SIGHUP   If the edit buffer has been modified since the last complete write, *ex* shall attempt  
13629           to save the edit buffer so that it can be recovered later using the **-r** option or the *ex*  
13630           **recover** command. The editor shall not write the file or return to command or text  
13631           input mode, and shall terminate with a non-zero exit status.
- 13632           SIGTERM   Refer to SIGHUP.
- 13633           The action taken for all other signals is unspecified.
- 13634   **STDOUT**
- 13635           The standard output shall be used only for writing prompts to the user, for informational  
13636           messages, and for writing lines from the file.
- 13637   **STDERR**
- 13638           Used only for diagnostic messages.
- 13639   **OUTPUT FILES**
- 13640           The output from *ex* shall be text files.
- 13641   **EXTENDED DESCRIPTION**
- 13642           Only the *ex* mode of the editor is described in this section. See *vi* for additional editing  
13643           capabilities available in *ex*.
- 13644           When an error occurs, *ex* shall write a message. If the terminal supports a standout mode (such  
13645           as inverse video), the message shall be written in standout mode. If the terminal does not  
13646           support a standout mode, and the edit option **errorbells** is set, an alert action shall precede the  
13647           error message.
- 13648           By default, *ex* shall start in command mode, which shall be indicated by a **:** prompt; see the  
13649           **prompt** command. Text input mode can be entered by the **append**, **insert**, or **change** commands;  
13650           it can be exited (and command mode re-entered) by typing a period ( **.** ) alone at the beginning  
13651           of a line.

13652        **Initialization in ex and vi**

13653        The following symbols are used in this and following sections to specify locations in the edit  
13654        buffer:

13655        *alternate and current path names*

13656        Two path names, named *current* and *alternate*, are maintained by the editor. Any *ex*  
13657        commands that take file names as arguments shall set them as follows:

- 13658        1. If a *file* argument is specified to the *ex edit*, *ex*, or *recover* commands, or if an *ex tag*  
13659        command replaces the contents of the edit buffer.
  - 13660            a. If the command replaces the contents of the edit buffer, the current path name  
13661            shall be set to the *file* argument or the file indicated by the tag, and the alternate  
13662            path name shall be set to the previous value of the current path name.
  - 13663            b. Otherwise, the alternate path name shall be set to the *file* argument.
- 13664        2. If a *file* argument is specified to the *ex next* command:
  - 13665            a. If the command replaces the contents of the edit buffer, the current path name  
13666            shall be set to the first *file* argument, and the alternate path name shall be set to  
13667            the previous value of the current path name.
- 13668        3. If a *file* argument is specified to the *ex file* command, the current path name shall be  
13669        set to the *file* argument, and the alternate path name shall be set to the previous value  
13670        of the current path name.
- 13671        4. If a *file* argument is specified to the *ex read* and *write* commands (that is, when  
13672        reading or writing a file, and not to the program named by the *shell* edit option), or a  
13673        *file* argument is specified to the *ex xit* command:
  - 13674            a. If the current path name has no value, the current path name shall be set to the  
13675            *file* argument.
  - 13676            b. Otherwise, the alternate path name shall be set to the *file* argument.

13677        If the alternate path name is set to the previous value of the current path name when the  
13678        current path name had no previous value, then the alternate path name shall have no value  
13679        as a result.

13680        *current line*

13681        The line of the edit buffer referenced by the cursor. Each command description specifies the  
13682        current line after the command has been executed, as the *current line value*. When the edit  
13683        buffer contains no lines, the current line shall be zero; see **Addressing in ex** (on page 2571).

13684        *current column*

13685        The current screen column occupied by the cursor. (The columns shall be numbered  
13686        beginning at 1.) Each command description specifies the current column after the command  
13687        has been executed, as the *current column value*. This column is an *ideal* column that is  
13688        remembered over the lifetime of the editor. The actual screen column upon which the cursor  
13689        rests may be different from the current column; see the cursor positioning discussion in  
13690        **Command Descriptions in vi** (on page 3201).

13691        *set to non-<blank>*

13692        A description for a current column value, meaning that the current column shall be set to  
13693        the last screen column on which is displayed any part of the first non-<blank> character of  
13694        the line. If the line has no non-<blank> characters, the current column shall be set to the last  
13695        screen column on which is displayed any part of the last character in the line. If the line is  
13696        empty, the current column shall be set to column position 1.

13697 The length of lines in the edit buffer may be limited to {LINE\_MAX} bytes. In open and visual  
 13698 mode, the length of lines in the edit buffer may be limited to the number of characters that will  
 13699 fit in the display. If either limit is exceeded during editing, an error message shall be written. If  
 13700 either limit is exceeded by a line read in from a file, an error message shall be written and the  
 13701 edit session may be terminated.

13702 If the editor stops running due to any reason other than a user command, and the edit buffer has  
 13703 been modified since the last complete write, it shall be equivalent to a SIGHUP asynchronous  
 13704 event. If the system crashes, it shall be equivalent to a SIGHUP asynchronous event.

13705 During initialization (before the first file is copied into the edit buffer or any user commands  
 13706 from the terminal are processed) the following shall occur:

13707 1. If the environment variable *EXINIT* is set, the editor shall execute the *ex* commands  
 13708 contained in that variable.

13709 2. If the *EXINIT* variable is not set, and all of the following are true:

13710 a. The *HOME* environment variable is not null and not empty.

13711 b. The file *.exrc* in the directory referred to by the *HOME* environment variable:

13712 1. Exists

13713 2. Is owned by the same user ID as the real user ID of the process or the process  
 13714 has appropriate privileges

13715 3. Is not writeable by anyone other than the owner

13716 the editor shall execute the *ex* commands contained in that file.

13717 3. If and only if all the following are true:

13718 a. The current directory is not referred to by the *HOME* environment variable.

13719 b. A command in the *EXINIT* environment variable or a command in the *.exrc* file in the  
 13720 directory referred to by the *HOME* environment variable sets the editor option *exrc*.

13721 c. The *.exrc* file in the current directory:

13722 1. Exists

13723 2. Is owned by the same user ID as the real user ID of the process, or by one of a  
 13724 set of implementation-defined user IDs

13725 3. Is not writeable by anyone other than the owner

13726 the editor shall attempt to execute the *ex* commands contained in that file.

13727 Lines in any *.exrc* file that contain no characters or only <blank> characters shall be ignored. If  
 13728 any *.exrc* file exists, but is not read for ownership or permission reasons, it shall be an error.

13729 After the *EXINIT* variable and any *.exrc* files are processed, the first file specified by the user  
 13730 shall be edited, as follows:

13731 1. If the user specified the *-t* option, the effect shall be as if the *ex tag* command was entered  
 13732 with the specified argument, with the exception that if tag processing does not result in a  
 13733 file to edit, the effect shall be as described in step 3. below.

13734 2. Otherwise, if the user specified any command line *file* arguments, the effect shall be as if  
 13735 the *ex edit* command was entered with the first of those arguments as its *file* argument.

13736 3. Otherwise, the effect shall be as if the *ex edit* command was entered with a nonexistent file  
 13737 name as its *file* argument. It is unspecified whether this action shall set the current path



13738 name. In an implementation where this action does not set the current path name, any  
 13739 editor command using the current path name shall fail until an editor command sets the  
 13740 current path name.

13741 If the `-r` option was specified the first time a file in the initial argument list or a file specified by  
 13742 the `-t` option is edited, if recovery information has previously been saved about it, that  
 13743 information shall be recovered and the editor shall behave as if the contents of the edit buffer  
 13744 have already been modified. If there are multiple instances of the file to be recovered, the one  
 13745 most recently saved shall be recovered, and an informational message that there are previous  
 13746 versions of the file that can be recovered shall be written. If no recovery information about a file  
 13747 is available, an informational message to this effect shall be written, and the edit shall proceed as  
 13748 usual.

13749 If the `-` option was specified the first time a file that already exists (including a file that might  
 13750 not exist but for which recovery information is available, when the `-r` option is specified)  
 13751 replaces or initializes the contents of the edit buffer, the current line shall be set to the last line of  
 13752 the edit buffer, the current column shall be set to non-`<blank>`, and the `ex` commands specified  
 13753 with the `-c` option shall be executed. In this case, the current line and current column shall not be  
 13754 set as described for the command associated with the replacement or initialization of the edit  
 13755 buffer contents. However, if the `-t` option or a `tag` command is associated with this action, the `-c`  
 13756 option commands shall be executed and then the movement to the tag shall be performed.

13757 The current argument list shall initially be set to the file names specified by the user on the  
 13758 command line. If no file names are specified by the user, the current argument list shall be  
 13759 empty. If the `-t` option was specified, it is unspecified whether any file name resulting from tag  
 13760 processing shall be prepended to the current argument list. In the case where the file name is  
 13761 added as a prefix to the current argument list, the current argument list reference shall be set to  
 13762 that file name. In the case where the file name is not added as a prefix to the current argument  
 13763 list, the current argument list reference shall logically be located before the first of the file names  
 13764 specified on the command line (for example, a subsequent `ex next` command shall edit the first  
 13765 file name from the command line). If the `-t` option was not specified, the current argument list  
 13766 reference shall be to the first of the file names on the command line.

### 13767 Addressing in `ex`

13768 Addressing in `ex` relates to the current line and the current column; the address of a line is its 1-  
 13769 based line number, the address of a column is its 1-based count from the beginning of the line.  
 13770 Generally, the current line is the last line affected by a command. The current line number is the  
 13771 address of the current line. In each command description, the effect of the command on the  
 13772 current line number and the current column is described.

13773 Addresses are constructed as follows:

- 13774 1. The character `'.'` (period) shall address the current line.
- 13775 2. The character `'$'` shall address the last line of the edit buffer.
- 13776 3. The positive decimal number `n` shall address the `n`th line of the edit buffer.
- 13777 4. The address `"'x"` refers to the line marked with the mark name character `'x'`, which shall  
 13778 be a lowercase letter from the portable character set or one of the characters `'\'` or `'\''`. It  
 13779 shall be an error if the line that was marked is not currently present in the edit buffer or the  
 13780 mark has not been set. Lines can be marked with the `ex mark` or `k` commands, or the `vi m`  
 13781 command.
- 13782 5. A regular expression (RE) enclosed by slashes (`'/'`) shall address the first line found by  
 13783 searching forwards from the line following the current line toward the end of the edit

13784 buffer and stopping at the first line containing a string matching the regular expression. As  
 13785 stated in **Regular Expressions in ex** (on page 2601), an address consisting of a null regular  
 13786 expression delimited by slashes "/" shall address the next line containing the last regular  
 13787 expression encountered. In addition, the second slash can be omitted at the end of a  
 13788 command line. If the **wrapsan** edit option is set, the search shall wrap around to the  
 13789 beginning of the edit buffer and continue up to and including the current line, so that the  
 13790 entire edit buffer is searched. Within the regular expression, the sequence "\/" shall  
 13791 represent a literal slash instead of the regular expression delimiter.

13792 6. A regular expression enclosed in question marks ('?') shall address the first line found by  
 13793 searching backwards from the line preceding the current line toward the beginning of the  
 13794 edit buffer and stopping at the first line containing a string matching the regular  
 13795 expression. The second question mark can be omitted at the end of a command line. If the  
 13796 **wrapsan** edit option is set, the search shall wrap around from the beginning of the edit  
 13797 buffer to the end of the edit buffer and continue up to and including the current line, so  
 13798 that the entire edit buffer is searched. Within the regular expression, the sequence "\?"  
 13799 shall represent a literal question mark instead of the RE delimiter.

13800 7. A plus sign ('+') or a minus sign ('-') followed by a decimal number shall address the  
 13801 current line plus or minus the number. A '+' or '-' not followed by a decimal number  
 13802 shall address the current line plus or minus 1.

13803 Addresses can be followed by zero or more address offsets, optionally <blank> character-  
 13804 separated. Address offsets are constructed as follows:

13805 1. A '+' or '-' immediately followed by a decimal number shall add (subtract) the  
 13806 indicated number of lines to (from) the address. A '+' or '-' not followed by a decimal  
 13807 number shall add (subtract) 1 to (from) the address.

13808 2. A decimal number shall add the indicated number of lines to the address.

13809 It shall not be an error for an intermediate address value to be less than zero or greater than the  
 13810 last line in the edit buffer. It shall be an error for the final address value to be less than zero or  
 13811 greater than the last line in the edit buffer.

13812 Commands take zero, one, or two addresses; see the descriptions of *1addr* and *2addr* in  
 13813 **Command Descriptions in ex** (on page 2578). If more than the required number of addresses  
 13814 are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more  
 13815 than the required number of addresses are provided to a command, the addresses specified first  
 13816 shall be evaluated and then discarded until the maximum number of valid addresses remain.

13817 Addresses shall be separated from each other by a comma (',') or a semicolon (';'). If no  
 13818 address is specified before or after a comma or semicolon separator, it shall be as if the address  
 13819 of the current line was specified before or after the separator. In the case of a semicolon  
 13820 separator, the current line ('.') shall be set to the first address, and only then will the next  
 13821 address be calculated. This feature can be used to determine the starting line for forwards and  
 13822 backwards searches (see rules 5. and 6.).

13823 A percent sign ('%') shall be equivalent to entering the two addresses "1, \$".

13824 Any delimiting <blank> characters between addresses, address separators, or address offsets  
 13825 shall be discarded.

13826 **Command Line Parsing in ex**

13827 The following symbol is used in this and following sections to describe parsing behavior:

13828 *escape* If a character is referred to as “backslash escaped” or “<control>-V escaped,” it  
 13829 shall mean that the character acquired or lost a special meaning by virtue of being  
 13830 preceded, respectively, by a backslash or <control>-V character. Unless otherwise  
 13831 specified, the escaping character shall be discarded at that time and shall not be  
 13832 further considered for any purpose.

13833 Command-line parsing shall be done in the following steps. For each step, characters already  
 13834 evaluated shall be ignored; that is, the phrase “leading character” refers to the next character  
 13835 that has not yet been evaluated.

- 13836 1. Leading colon characters shall be skipped.
  - 13837 2. Leading <blank> characters shall be skipped.
  - 13838 3. If the leading character is a double-quote character, the characters up to and including the  
 13839 next non-backslash-escaped <newline> character shall be discarded, and any subsequent  
 13840 characters shall be parsed as a separate command.
  - 13841 4. Leading characters that can be interpreted as addresses shall be evaluated; see **Addressing**  
 13842 **in ex** (on page 2571).
  - 13843 5. Leading <blank> characters shall be skipped.
  - 13844 6. If the next character is a vertical-line character or a <newline> character:
    - 13845 a. If the next character is a <newline> character:
      - 13846 1. If *ex* is in open or visual mode, the current line shall be set to the last address  
 13847 specified, if any.
      - 13848 2. Otherwise, if the last command was terminated by a vertical-line character, no  
 13849 action shall be taken; for example, the command "||<newline>" shall  
 13850 execute two implied commands, not three.
      - 13851 3. Otherwise, step 6.b. shall apply.
    - 13852 b. Otherwise, the implied command shall be the **print** command. The last #, **p**, and **l**  
 13853 flags specified to any *ex* command shall be remembered and shall apply to this  
 13854 implied command. Executing the *ex* **number**, **print**, or **list** command shall set the  
 13855 remembered flags to #, nothing, and **l**, respectively, plus any other flags specified for  
 13856 that execution of the **number**, **print**, or **list** command.  
 If *ex* is not currently performing a **global** or **v** command, and no address or count is  
 13857 specified, the current line shall be incremented by 1 before the command is executed.  
 13858 If incrementing the current line would result in an address past the last line in the  
 13859 edit buffer, the command shall fail, and the increment shall not happen.
  - 13861 c. The <newline> character or vertical-line character shall be discarded and any  
 13862 subsequent characters shall be parsed as a separate command.
- 13863 7. The command name shall be comprised of the next character (if the character is not  
 13864 alphabetic), or the next character and any subsequent alphabetic characters (if the  
 13865 character is alphabetic), with the following exceptions:
  - 13866 a. Commands that consist of any prefix of the characters in the command name **delete**,  
 13867 followed immediately by any of the characters **l**, **p**, **+**, **-**, or **#** shall be interpreted as a  
 13868 **delete** command, followed by a <blank> character, followed by the characters that

13869 were not part of the prefix of the **delete** command. The maximum number of  
 13870 characters shall be matched to the command name **delete**; for example, "de1" shall  
 13871 not be treated as "de" followed by the flag **l**.

13872 b. Commands that consist of the character **k**, followed by a character that can be used  
 13873 as the name of a mark, shall be equivalent to the mark command followed by a  
 13874 <blank> character, followed by the character that followed the **k**.

13875 c. Commands that consist of the character **s**, followed by characters that could be  
 13876 interpreted as valid options to the **s** command, shall be the equivalent of the **s**  
 13877 command, without any pattern or replacement values, followed by a <blank>  
 13878 character, followed by the characters after the **s**.

13879 8. The command name shall be matched against the possible command names, and a  
 13880 command name that contains a prefix matching the characters specified by the user shall  
 13881 be the executed command. In the case of commands where the characters specified by the  
 13882 user could be ambiguous, the executed command shall be as follows:

13883  
 13884  
 13885  
 13886  
 13887  
 13888

|           |               |           |              |           |              |
|-----------|---------------|-----------|--------------|-----------|--------------|
| <b>a</b>  | <b>append</b> | <b>n</b>  | <b>next</b>  | <b>t</b>  | <b>t</b>     |
| <b>c</b>  | <b>change</b> | <b>p</b>  | <b>print</b> | <b>u</b>  | <b>undo</b>  |
| <b>ch</b> | <b>change</b> | <b>pr</b> | <b>print</b> | <b>un</b> | <b>undo</b>  |
| <b>e</b>  | <b>edit</b>   | <b>r</b>  | <b>read</b>  | <b>v</b>  | <b>v</b>     |
| <b>m</b>  | <b>move</b>   | <b>re</b> | <b>read</b>  | <b>w</b>  | <b>write</b> |
| <b>ma</b> | <b>mark</b>   | <b>s</b>  | <b>s</b>     |           |              |

13889 Implementation extensions with names causing similar ambiguities shall not be checked  
 13890 for a match until all possible matches for commands specified by IEEE Std. 1003.1-200x  
 13891 have been checked.

13892 9. If the command is a **!** command, or if the command is a **read** command followed by zero  
 13893 or more <blank> characters and a **!**, or if the command is a **write** command followed by  
 13894 one or more <blank> characters and a **!**, the rest of the command shall include all  
 13895 characters up to a non-backslash-escaped <newline> character. The <newline> character  
 13896 shall be discarded and any subsequent characters shall be parsed as a separate *ex*  
 13897 command.

13898 10. Otherwise, if the command is an **edit**, **ex**, or **next** command, or a **visual** command while in  
 13899 open or visual mode, the next part of the command shall be parsed as follows:

13900 a. Any **' ! '** character immediately following the command shall be skipped and be part  
 13901 of the command.

13902 b. Any leading <blank> characters shall be skipped and be part of the command.

13903 c. If the next character is a **' + '**, characters up to the first non-backslash-escaped  
 13904 <newline> character or non-backslash-escaped <blank> character shall be skipped  
 13905 and be part of the command.

13906 d. The rest of the command shall be determined by the steps specified in paragraph 12.

13907 11. Otherwise, if the command is a **global**, **open**, **s**, or **v** command, the next part of the  
 13908 command shall be parsed as follows:

13909 a. Any leading <blank> characters shall be skipped and be part of the command.

13910 b. If the next character is not an alphanumeric, double-quote, <newline>, backslash, or  
 13911 vertical-line character:

13912 1. The next character shall be used as a command delimiter.

- 13913                   2. If the command is a **global**, **open**, or **v** command, characters up to the first  
13914 non-backslash-escaped <newline> character, or first non-backslash-escaped  
13915 delimiter character, shall be skipped and be part of the command.
- 13916                   3. If the command is an **s** command, characters up to the first non-backslash-  
13917 escaped <newline> character, or second non-backslash-escaped delimiter  
13918 character, shall be skipped and be part of the command.
- 13919                   c. If the command is a **global** or **v** command, characters up to the first non-backslash-  
13920 escaped <newline> character shall be skipped and be part of the command.
- 13921                   d. Otherwise, the rest of the command shall be determined by the steps specified in  
13922 paragraph 12.

13923           12. Otherwise:

- 13924                   a. If the command was a **map**, **unmap**, **abbreviate**, or **unabbreviate** command,  
13925 characters up to the first non-<control>-V-escaped <newline>, vertical-line, or  
13926 double-quote character shall be skipped and be part of the command.
- 13927                   b. Otherwise, characters up to the first non-backslash-escaped <newline>, vertical-line,  
13928 or double-quote character shall be skipped and be part of the command.
- 13929                   c. If the command was an **append**, **change**, or **insert** command, and the step 12.b.  
13930 ended at a vertical-line character, any subsequent characters, up to the next non-  
13931 backslash-escaped <newline> character shall be used as input text to the command.
- 13932                   d. If the command was ended by a double-quote character, all subsequent characters,  
13933 up to the next non-backslash-escaped <newline> character, shall be discarded.
- 13934                   e. The terminating <newline> or vertical-line character shall be discarded and any  
13935 subsequent characters shall be parsed as a separate *ex* command.

13936           Command arguments shall be parsed as described by the Synopsis and Description of each  
13937 individual *ex* command. This parsing shall not be <blank> character-sensitive, except for the **!**  
13938 argument, which must follow the command name without intervening <blank> characters, and  
13939 where it would otherwise be ambiguous. For example, *count* and *flag* arguments need not be  
13940 <blank> character separated because "d22p" is not ambiguous, but *file* arguments to the *ex next*  
13941 command must be separated by one or more <blank> characters. Any <blank> character in  
13942 command arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands can be  
13943 <control>-V-escaped, in which case the <blank> character shall not be used as an argument  
13944 delimiter. Any <blank> character in the command argument for any other command can be  
13945 backslash-escaped, in which case that <blank> character shall not be used as an argument  
13946 delimiter.

13947           Within command arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands,  
13948 any character can be <control>-V-escaped. All such escaped characters shall be treated literally  
13949 and shall have no special meaning. Within command arguments for all other *ex* commands that  
13950 are not regular expressions or replacement strings, any character that would otherwise have a  
13951 special meaning can be backslash-escaped. Escaped characters shall be treated literally, without  
13952 special meaning as shell expansion characters or '!', '%', and '#' expansion characters. See  
13953 **Regular Expressions in ex** (on page 2601) and **Replacement Strings in ex** (on page 2602) for  
13954 descriptions of command arguments that are regular expressions or replacement strings.

13955           Non-backslash-escaped '%' characters appearing in *file* arguments to any *ex* command shall be  
13956 replaced by the current path name; unescaped '#' characters shall be replaced by the alternate  
13957 path name. It shall be an error if '%' or '#' characters appear unescaped in an argument and  
13958 their corresponding values are not set.

13959 Non-backslash-escaped '!' characters in the arguments to either the **ex!** command or the open  
 13960 and visual mode **!** command, or in the arguments to the **ex read** command, where the first non-  
 13961 <blank> character after the command name is a '!' character, or in the arguments to the **ex**  
 13962 **write** command where the command name is followed by one or more <blank> characters and  
 13963 the first non-<blank> character after the command name is a '!' character, shall be replaced  
 13964 with the arguments to the last of those three commands as they appeared after all unescaped  
 13965 '%', '#', and '!' characters were replaced. It shall be an error if '!' characters appear  
 13966 unescaped in one of these commands and there has been no previous execution of one of these  
 13967 commands.

13968 If an error occurs during the parsing or execution of an **ex** command:

- 13969 • An informational message to this effect shall be written. Execution of the **ex** command shall  
 13970 stop, and the cursor (for example, the current line and column) shall not be further modified.
- 13971 • If the **ex** command resulted from a map expansion, all characters from that map expansion  
 13972 shall be discarded, except as otherwise specified by the **map** command.
- 13973 • Otherwise, if the **ex** command resulted from the processing of an *EXINIT* environment  
 13974 variable, a **.exrc** file, a **:source** command, a **-c** option, or a **+command** specified to an **ex edit**,  
 13975 **ex, next**, or **visual** command, no further commands from the source of the commands shall  
 13976 be executed.
- 13977 • Otherwise, if the **ex** command resulted from the execution of a buffer or a **global** or **v**  
 13978 command, no further commands caused by the execution of the buffer or the **global** or **v**  
 13979 command shall be executed.
- 13980 • Otherwise, if the **ex** command was not terminated by a <newline> character, all characters up  
 13981 to and including the next non-backslash-escaped <newline> character shall be discarded.

### 13982 **Input Editing in ex**

13983 The following symbols are used in this and following sections to specify command actions.

13984 *word* In the POSIX locale, a word consists of a maximal sequence of letters, digits, and  
 13985 underscores, delimited at both ends by characters other than letters, digits, or  
 13986 underscores, or by the beginning or end of a line or the edit buffer.

13987 When accepting input characters from the user, in either **ex** command mode or **ex** text input  
 13988 mode, **ex** shall enable canonical mode input processing, as defined in the System Interfaces  
 13989 volume of IEEE Std. 1003.1-200x.

13990 If in **ex** text input mode:

- 13991 1. If the **number** edit option is set, **ex** shall prompt for input using the line number that would  
 13992 be assigned to the line if it is entered, in the format specified for the **ex number** command.
- 13993 2. If the **autoindent** edit option is set, **ex** shall prompt for input using **autoindent** characters,  
 13994 as described by the **autoindent** edit option. **autoindent** characters shall follow the line  
 13995 number, if any.

13996 If in **ex** command mode:

- 13997 1. If the **prompt** edit option is set, input shall be prompted for using a single ':' character;  
 13998 otherwise, there shall be no prompt.

13999 The input characters in the following sections shall have the following effects on the input line.

- 14000       **Scroll**
- 14001       *Synopsis:*     eof
- 14002       See the description of the *stty eof* character in *stty*.
- 14003       If in *ex* command mode:
- 14004             If the *eof* character is the first character entered on the line, the line shall be evaluated as if it
- 14005             contained two characters: a <control>-D and a <newline> character.
- 14006             Otherwise, the *eof* character shall have no special meaning.
- 14007       If in *ex* text input mode:
- 14008             If the cursor follows an **autoindent** character, the **autoindent** characters in the line shall be
- 14009             modified so that a part of the next text input character will be displayed on the first column
- 14010             in the line after the previous **shiftwidth** edit option column boundary, and the user shall be
- 14011             prompted again for input for the same line.
- 14012             Otherwise, if the cursor follows a '0', which follows an **autoindent** character, and the '0'
- 14013             was the previous text input character, the '0' and all **autoindent** characters in the line shall
- 14014             be discarded, and the user shall be prompted again for input for the same line.
- 14015             Otherwise, if the cursor follows a '^', which follows an **autoindent** character, and the '^'
- 14016             was the previous text input character, the '^' and all **autoindent** characters in the line shall
- 14017             be discarded, and the user shall be prompted again for input for the same line. In addition,
- 14018             the **autoindent** level for the next input line shall be derived from the same line from which
- 14019             the **autoindent** level for the current input line was derived.
- 14020             Otherwise, if there are no **autoindent** or text input characters in the line, the *eof* character
- 14021             shall be discarded.
- 14022             Otherwise, the *eof* character shall have no special meaning.
- 14023       <newline>
- 14024       *Synopsis:*     <newline>
- 14025                     <control>-J
- 14026       If in *ex* command mode:
- 14027             Cause the command line to be parsed; <control>-J shall be mapped to the <newline>
- 14028             character for this purpose.
- 14029       If in *ex* text input mode:
- 14030             Terminate the current line. If there are no characters other than **autoindent** characters on the
- 14031             line, all characters on the line shall be discarded.
- 14032             Prompt for text input on a new line after the current line. If the **autoindent** edit option is set,
- 14033             an appropriate number of **autoindent** characters shall be added as a prefix to the line as
- 14034             described by the *ex autoindent* edit option.

- 14035        **<backslash>**
- 14036        *Synopsis:*     <backslash>
- 14037        Allow the entry of a subsequent <newline> or <control>-J as a literal character, removing any special meaning that it may have to the editor during text input mode. The backslash character shall be retained and evaluated when the command line is parsed, or retained and included when the input text becomes part of the edit buffer.
- 14038
- 14039
- 14040
- 14041        **<control>-V**
- 14042        *Synopsis:*     <control>-V
- 14043        Allow the entry of any subsequent character as a literal character, removing any special meaning that it may have to the editor during text input mode. The <control>-V character shall be discarded before the command line is parsed or the input text becomes part of the edit buffer.
- 14044
- 14045
- 14046        If the “literal next” functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>-V performs this function.
- 14047
- 14048        **<control>-W**
- 14049        *Synopsis:*     <control>-W
- 14050        Discard the <control>-W, and the word previous to it in the input line, including any <blank> characters following the word and preceding the <control>-W. If the “word erase” functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>-W performs this function.
- 14051
- 14052
- 14053
- 14054        **Command Descriptions in ex**
- 14055        The following symbols are used in this section to represent command modifiers. Some of these modifiers can be omitted, in which case the specified defaults shall be used.
- 14056
- 14057        *1addr*        A single line address, given in any of the forms described in **Addressing in ex** (on page 2571); the default shall be the current line ( ' . ' ), unless otherwise specified.
- 14058
- 14059        If the line address is zero, it shall be an error, unless otherwise specified in the following command descriptions.
- 14060
- 14061        If the edit buffer is empty, and the address is specified with a command other than =, **append**, **insert**, **open**, **put**, **read**, or **visual**, or the address is not zero, it shall be an error.
- 14062
- 14063
- 14064        *2addr*        Two addresses specifying an inclusive range of lines. If no addresses are specified, the default for *2addr* shall be the current line only ( " . . " ), unless otherwise specified in the following command descriptions. If one address is specified, *2addr* shall specify that line only, unless otherwise specified in the following command descriptions.
- 14065
- 14066
- 14067
- 14068
- 14069        It shall be an error if the first address is greater than the second address.
- 14070        If the edit buffer is empty, and the two addresses are specified with a command other than the !, **write**, **wq**, or **xit** commands, or either address is not zero, it shall be an error.
- 14071
- 14072
- 14073        *count*        A positive decimal number. If *count* is specified, it shall be equivalent to specifying an additional address to the command, unless otherwise specified by the following command descriptions. The additional address shall be equal to the last address specified to the command (either explicitly or by default) plus *count*-1.
- 14074
- 14075
- 14076



- 14077 If this would result in an address greater than the last line of the edit buffer, it shall  
14078 be corrected to equal the last line of the edit buffer.
- 14079 *flags* One or more of the characters '+', '-', '#', 'p', or 'l' (ell). The flag characters  
14080 can be <blank>-separated, and in any order or combination. The characters '#',  
14081 'p', and 'l' shall cause lines to be written in the format specified by the **print**  
14082 command with the specified *flags*.
- 14083 The lines to be written are as follows:
- 14084 1. All edit buffer lines written during the execution of the *ex* &, ~, **list**, **number**,  
14085 **open**, **print**, **s**, **visual**, and **z** commands shall be written as specified by *flags*.
  - 14086 2. After the completion of an *ex* command with a flag as an argument, the  
14087 current line shall be written as specified by *flags*, unless the current line was  
14088 the last line written by the command.
- 14089 The characters '+' and '-' cause the value of the current line after the execution  
14090 of the *ex* command to be adjusted by the offset address as described in **Addressing**  
14091 **in ex** (on page 2571). This adjustment shall occur before the current line is written  
14092 as described in 2. above.
- 14093 The default for *flags* shall be none.
- 14094 *buffer* One of a number of named areas for holding text. The named buffers are specified  
14095 by the alphanumeric characters of the POSIX locale. There shall also be one  
14096 "unnamed" buffer. When no buffer is specified for editor commands that use a  
14097 buffer, the unnamed buffer shall be used. Commands that store text into buffers  
14098 shall store the text as it was before the command took effect, and shall store text  
14099 occurring earlier in the file before text occurring later in the file, regardless of how  
14100 the text region was specified. Commands that store text into buffers shall store the  
14101 text into the unnamed buffer as well as any specified buffer.
- 14102 In *ex* commands, buffer names are specified as the name by itself. In open or visual  
14103 mode commands the name is preceded by a double quote ('"') character.
- 14104 If the specified buffer name is an uppercase character, and the buffer contents are  
14105 to be modified, the buffer shall be appended to rather than being overwritten. If  
14106 the buffer is not being modified, specifying the buffer name in lowercase and  
14107 uppercase shall have identical results.
- 14108 There shall also be buffers named by the numbers 1 through 9. In open and visual  
14109 mode, if a region of text including characters from more than a single line is being  
14110 modified by the *vi* **c** or **d** commands, the motion character associated with the **c** or  
14111 **d** commands specifies that the buffer text shall be in line mode, or the commands  
14112 %, /, ?, (, ), **N**, **n**, {, or } are used to define a region of text for the **c** or **d** commands,  
14113 the contents of buffers 1 through 8 shall be moved into the buffer named by the  
14114 next numerically greater value, the contents of buffer 9 shall be discarded, and the  
14115 region of text shall be copied into buffer 1. This shall be in addition to copying the  
14116 text into a user-specified buffer or unnamed buffer, or both. Numeric buffers can  
14117 be specified as a source buffer for open and visual mode commands; however,  
14118 specifying a numeric buffer as the write target of an open or visual mode  
14119 command shall have unspecified results.
- 14120 The text of each buffer shall have the characteristic of being in either line or  
14121 character mode. Appending text to a non-empty buffer shall set the mode to match  
14122 the characteristic of the text being appended. Appending text to a buffer shall  
14123 cause the creation of at least one additional line in the buffer. All text stored into

14124 buffers by *ex* commands shall be in line mode. The *ex* commands that use buffers  
 14125 as the source of text specify individually how buffers of different modes are  
 14126 handled. Each open or visual mode command that uses buffers for any purpose  
 14127 specifies individually the mode of the text stored into the buffer and how buffers  
 14128 of different modes are handled.

14129 *file* Command text used to derive a path name. The default shall be the current path  
 14130 name, as defined previously, in which case, if no current path name has yet been  
 14131 established it shall be an error, except where specifically noted in the individual  
 14132 command descriptions that follow. If the command text contains any of the  
 14133 characters '~', '{', '[', '\*', '?', '\$', '\', ' ', ' ', ' ', ' ', and '\', it shall be  
 14134 subjected to the process of "shell expansions", as described below; if more than a  
 14135 single path name results and the command expects only one, it shall be an error.

14136 The process of shell expansions in the editor shall be done as follows. The *ex* utility  
 14137 shall pass two arguments to the program named by the shell edit option; the first  
 14138 shall be `-c`, and the second shall be the string "echo" and the command text as a  
 14139 single argument. The standard output and standard error of that command shall  
 14140 replace the command text.

14141 **!** A character that can be appended to the command name to modify its operation,  
 14142 as detailed in the individual command descriptions. With the exception of the *ex*  
 14143 **read**, **write**, and **!** commands, the '!' character shall only act as a modifier if there  
 14144 are no <blank> characters between it and the command name.

14145 *remembered search direction*

14146 The *vi* commands **N** and **n** begin searching in a forwards or backwards direction in  
 14147 the edit buffer based on a remembered search direction, which is initially unset,  
 14148 and is set by the *ex* **global**, **v**, **s**, and **tag** commands, and the *vi* / and ? commands.

## 14149 Abbreviate

14150 *Synopsis:* `ab[breviate][lhs rhs]`

14151 If *lhs* and *rhs* are not specified, write the current list of abbreviations and do nothing more.

14152 Implementations may restrict the set of characters accepted in *lhs* or *rh*, except that printable  
 14153 characters and <blank> characters shall not be restricted. Additional restrictions shall be  
 14154 implementation-defined.

14155 In both *lhs* and *rhs*, any character may be escaped with a <control>-V, in which case the  
 14156 character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V shall be  
 14157 discarded.

14158 In open and visual text input mode, if a non-word or <ESC> character that is not escaped by a  
 14159 <control>-V character is entered after a word character, a check shall be made for a set of  
 14160 characters matching *lhs*, in the text input entered during this command. If it is found, the effect  
 14161 shall be as if *rhs* was entered instead of *lhs*.

14162 The set of characters that are checked is defined as follows:

- 14163 1. If there are no characters inserted before the word and non-word or <ESC> characters that  
 14164 triggered the check, the set of characters shall consist of the word character.
- 14165 2. If the character inserted before the word and non-word or <ESC> characters that triggered  
 14166 the check is a word character, the set of characters shall consist of the characters inserted  
 14167 immediately before the triggering characters that are word characters, plus the triggering  
 14168 word character.

14169 3. If the character inserted before the word and non-word or <ESC> characters that triggered  
 14170 the check is not a word character, the set of characters shall consist of the characters that  
 14171 were inserted before the triggering characters that are neither <blank> characters nor word  
 14172 characters, plus the triggering word character.

14173 It is unspecified whether the *lhs* argument entered for the *ex* **abbreviate** and **unabbreviate**  
 14174 commands is replaced in this fashion. Regardless of whether or not the replacement occurs, the  
 14175 effect of the command shall be as if the replacement had not occurred.

14176 *Current line*: Unchanged.

14177 *Current column*: Unchanged.

## 14178 **Append**

14179 *Synopsis*: [1*addr*] a[ppend][!]

14180 Enter text input mode; the input text shall be placed after the specified line. If line zero is  
 14181 specified, the text shall be placed at the beginning of the edit buffer.

14182 This command shall be affected by the **number** and **autoindent** edit options; following the  
 14183 command name with '!' shall cause the **autoindent** edit option setting to be toggled for the  
 14184 duration of this command only.

14185 *Current line*: Set to the last input line; if no lines were input, set to the specified line, or to the  
 14186 first line of the edit buffer if a line of zero was specified, or zero if the edit buffer is empty.

14187 *Current column*: Set to non-<blank>.

## 14188 **Arguments**

14189 *Synopsis*: ar[gs]

14190 Write the current argument list, with the current argument-list entry, if any, between '[' and  
 14191 ']' characters.

14192 *Current line*: Unchanged.

14193 *Current column*: Unchanged.

## 14194 **Change**

14195 *Synopsis*: [2*addr*] c[hange][!][*count*]

14196 Enter *ex* text input mode; the input text shall replace the specified lines. The specified lines shall  
 14197 be copied into the unnamed buffer, which shall become a line mode buffer.

14198 This command shall be affected by the **number** and **autoindent** edit options; following the  
 14199 command name with '!' shall cause the **autoindent** edit option setting to be toggled for the  
 14200 duration of this command only.

14201 *Current line*: Set to the last input line; if no lines were input, set to the line before the first  
 14202 address, or to the first line of the edit buffer if there are no lines preceding the first address, or to  
 14203 zero if the edit buffer is empty.

14204 *Current column*: Set to non-<blank>.

14205 **Change Directory**

14206 *Synopsis:* chd[ir][!][*directory*]  
 14207 cd[!][*directory*]

14208 Change the current working directory to *directory*.

14209 If no *directory* argument is specified, and the *HOME* environment variable is set to a non-null  
 14210 and non-empty value, *directory* shall default to the value named in the *HOME* environment  
 14211 variable. If the *HOME* environment variable is empty or is undefined, the default value of  
 14212 *directory* is implementation-defined.

14213 If no '!' is appended to the command name, and the edit buffer has been modified since the  
 14214 last complete write, and the current path name does not begin with a '/', it shall be an error.

14215 *Current line:* Unchanged.

14216 *Current column:* Unchanged.

14217 **Copy**

14218 *Synopsis:* [*laddr*] co[py] [*laddr*] [*flags*]  
 14219 [*laddr*] t [*laddr*] [*flags*]

14220 Copy the specified lines after the specified destination line; line zero specifies that the lines shall  
 14221 be placed at the beginning of the edit buffer.

14222 *Current line:* Set to the last line copied.

14223 *Current column:* Set to non-<blank>.

14224 **Delete**

14225 *Synopsis:* [*laddr*] d[elete][*buffer*][*count*][*flags*]

14226 Delete the specified lines into a buffer (defaulting to the unnamed buffer), which shall become a  
 14227 line-mode buffer.

14228 Flags can immediately follow the command name; see **Command Line Parsing in ex** (on page  
 14229 2573).

14230 *Current line:* Set to the line following the deleted lines, or to the last line in the edit buffer if that  
 14231 line is past the end of the edit buffer, or to zero if the edit buffer is empty.

14232 *Current column:* Set to non-<blank>.

14233 **Edit**

14234 *Synopsis:* e[dit][!][+*command*][*file*]  
 14235 ex[!][+*command*][*file*]

14236 If no '!' is appended to the command name, and the edit buffer has been modified since the  
 14237 last complete write, it shall be an error.

14238 If *file* is specified, replace the current contents of the edit buffer with the current contents of *file*,  
 14239 and set the current path name to *file*. If *file* is not specified, replace the current contents of the  
 14240 edit buffer with the current contents of the file named by the current path name. If for any  
 14241 reason the current contents of the file cannot be accessed, the edit buffer shall be empty.

14242 The +*command* option shall be <blank> character-delimited; <blank> characters within  
 14243 +*command* can be escaped by preceding them with a backslash character. The +*command* shall be  
 14244 interpreted as an *ex* command immediately after the contents of the edit buffer have been

- 14245 replaced and the current line and column have been set.
- 14246 If the edit buffer is empty:
- 14247 *Current line*: Set to 0.
- 14248 *Current column*: Set to 1.
- 14249 Otherwise, if executed while in *ex* command mode or if the *+command* argument is specified:
- 14250 *Current line*: Set to the last line of the edit buffer.
- 14251 *Current column*: Set to non-<blank>.
- 14252 Otherwise, if *file* is omitted or results in the current path name:
- 14253 *Current line*: Set to the first line of the edit buffer.
- 14254 *Current column*: Set to non-<blank>.
- 14255 Otherwise, if *file* is the same as the last file edited, the line and column shall be set as follows; if
- 14256 the file was previously edited, the line and column may be set as follows:
- 14257 *Current line*: Set to the last value held when that file was last edited. If this value is not a valid
- 14258 line in the new edit buffer, set to the first line of the edit buffer.
- 14259 *Current column*: If the current line was set to the last value held when the file was last edited, set
- 14260 to the last value held when the file was last edited. Otherwise, or if the last value is not a valid
- 14261 column in the new edit buffer, set to non-<blank>.
- 14262 Otherwise:
- 14263 *Current line*: Set to the first line of the edit buffer.
- 14264 *Current column*: Set to non-<blank>.
- 14265 **File**
- 14266 *Synopsis*: f[*file*][*file*]
- 14267 If a *file* argument is specified, the alternate path name shall be set to the current path name, and
- 14268 the current path name shall be set to *file*.
- 14269 Write an informational message. If the file has a current path name, it shall be included in this
- 14270 message; otherwise, the message shall indicate that there is no current path name. If the edit
- 14271 buffer contains lines, the current line number and the number of lines in the edit buffer shall be
- 14272 included in this message; otherwise, the message shall indicate that the edit buffer is empty. If
- 14273 the edit buffer has been modified since the last complete write, this fact shall be included in this
- 14274 message. If the **readonly** edit option is set, this fact shall be included in this message. The
- 14275 message may contain other unspecified information.
- 14276 *Current line*: Unchanged.
- 14277 *Current column*: Unchanged.

14278 **Global**

14279 *Synopsis:* [2addr] g[lobal] /pattern/ [commands]  
 14280 [2addr] v /pattern/ [commands]

14281 The optional '!' character after the **global** command shall be the same as executing the **v**  
 14282 command.

14283 If *pattern* is empty (for example, "//") or not specified, the last regular expression used in the  
 14284 editor command shall be used as the *pattern*. The *pattern* can be delimited by slashes (shown in  
 14285 the Synopsis), as well as any non-alphanumeric or non-<blank> character other than backslash,  
 14286 vertical line, double quote, or <newline>.

14287 If no lines are specified, the lines shall default to the entire file.

14288 The **global** and **v** commands are logically two-pass operations. First, mark the lines within the  
 14289 specified lines that match (**global**) or do not match (**v** or **global!**) the specified pattern. Second,  
 14290 execute the *ex* commands given by *commands*, with the current line ('.') set to each marked  
 14291 line. If an error occurs during this process, or the contents of the edit buffer are replaced (for  
 14292 example, by the *ex* **:edit** command) an error message shall be written and no more commands  
 14293 resulting from the execution of this command shall be processed.

14294 Multiple *ex* commands can be specified by entering multiple commands on a single line using a  
 14295 vertical line to delimit them, or one per line, by escaping each <newline> with a backslash.

14296 If no commands are specified:

- 14297 1. If in *ex* command mode, it shall be as if the **print** command were specified.
- 14298 2. Otherwise, no command shall be executed.

14299 For the **append**, **change**, and **insert** commands, the input text shall be included as part of the  
 14300 command, and the terminating period can be omitted if the command ends the list of  
 14301 commands. The **open** and **visual** commands can be specified as one of the commands, in which  
 14302 case each marked line shall cause the editor to enter open or visual mode. If open or visual mode  
 14303 is exited using the *vi* **Q** command, the current line shall be set to the next marked line, and open  
 14304 or visual mode reentered, until the list of marked lines is exhausted.

14305 The **global**, **v**, and **undo** commands cannot be used in *commands*. Marked lines may be deleted  
 14306 by commands executed for lines occurring earlier in the file than the marked lines. In this case,  
 14307 no commands shall be executed for the deleted lines.

14308 If the remembered search direction is not set, the **global** and **v** commands shall set it to forward.

14309 The **autoprint** and **autoindent** edit options shall be inhibited for the duration of the **g** or **v**  
 14310 command.

14311 *Current line:* If no commands executed, set to the last marked line. Otherwise, as specified for  
 14312 the executed *ex* commands.

14313 *Current column:* If no commands are executed, set to non-<blank>; otherwise, as specified for the  
 14314 individual *ex* commands.

14315 **Insert**14316 *Synopsis:* [1addr] i[nsert][!]14317 Enter *ex* text input mode; the input text shall be placed before the specified line. If the line is zero  
14318 or 1, the text shall be placed at the beginning of the edit buffer.14319 This command shall be affected by the **number** and **autoindent** edit options; following the  
14320 command name with '!' shall cause the **autoindent** edit option setting to be toggled for the  
14321 duration of this command only.14322 *Current line:* Set to the last input line; if no lines were input, set to the line before the specified  
14323 line, or to the first line of the edit buffer if there are no lines preceding the specified line, or zero  
14324 if the edit buffer is empty.14325 *Current column:* Set to non-<blank>.14326 **Join**14327 *Synopsis:* [2addr] j[oin][!][count][flags]14328 If *count* is specified:14329 If no address was specified, the **join** command shall behave as if *2addr* were the current line  
14330 and the current line plus *count* (.,. + *count*).14331 If one address was specified, the **join** command shall behave as if *2addr* were the specified  
14332 address and the specified address plus *count* (*addr*,*addr* + *count*).14333 If two addresses were specified, the **join** command shall behave as if an additional address,  
14334 equal to the last address plus *count* - 1 (*addr1*,*addr2*,*addr2* + *count* - 1), was specified.14335 If this would result in a second address greater than the last line of the edit buffer, it shall be  
14336 corrected to be equal to the last line of the edit buffer.14337 If no *count* is specified:14338 If no address was specified, the **join** command shall behave as if *2addr* were the current line  
14339 and the next line (.,. + 1).14340 If one address was specified, the **join** command shall behave as if *2addr* were the specified  
14341 address and the next line (*addr*,*addr* + 1).14342 Join the text from the specified lines together into a single line, which shall replace the specified  
14343 lines.14344 If a '!' character is appended to the command name, the **join** shall be without modification of  
14345 any line, independent of the current locale.14346 Otherwise, in the POSIX locale, set the current line to the first of the specified lines, and then, for  
14347 each subsequent line, proceed as follows:

- 14348 1. Discard leading <space>s from the line to be joined.
- 14349 2. If the line to be joined is now empty, delete it, and skip steps 3 through 5.
- 14350 3. If the current line ends in a <blank> character, or the first character of the line to be joined  
14351 is a ' ) ' character, join the lines without further modification.
- 14352 4. If the last character of the current line is a ' . ' , join the lines with two <space> characters  
14353 between them.

14354 5. Otherwise, join the lines with a single <space> character between them.

14355 *Current line*: Set to the first line specified.

14356 *Current column*: Set to non-<blank>.

### 14357 **List**

14358 *Synopsis*: [2*addr*] l[*ist*][*count*][*flags*]

14359 This command shall be equivalent to the *ex* command:

14360 [2*addr*] p[*rint*][*count*] l[*flags*]

14361 See **Print** (on page 2590).

### 14362 **Map**

14363 *Synopsis*: map[!][*lhs rhs*]

14364 If *lhs* and *rhs* are not specified:

- 14365 1. If '!' is specified, write the current list of text input mode maps.
- 14366 2. Otherwise, write the current list of command mode maps.
- 14367 3. Do nothing more.

14368 Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except that printable  
 14369 characters and <blank> characters shall not be restricted. Additional restrictions shall be  
 14370 implementation-defined. In both *lhs* and *rhs*, any character can be escaped with a <control>-V, in  
 14371 which case the character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V  
 14372 shall be discarded.

14373 If the character '!' is appended to the **map** command name, the mapping shall be effective  
 14374 during open or visual text input mode rather than **open** or **visual** command mode. This allows  
 14375 *lhs* to have two different **map** definitions at the same time: one for command mode and one for  
 14376 text input mode.

14377 For command mode mappings:

14378 When the *lhs* is entered as any part of a *vi* command in open or visual mode (but not as part  
 14379 of the arguments to the command), the action shall be as if the corresponding *rhs* had been  
 14380 entered.

14381 If any character in the command, other than the first, is escaped using a <control>-V  
 14382 character, that character shall not be part of a match to an *lhs*.

14383 It is unspecified whether implementations shall support **map** commands where the *lhs* is  
 14384 more than a single character in length, where the first character of the *lhs* is printable.

14385 If *lhs* contains more than one character and the first character is '#', followed by a sequence  
 14386 of digits corresponding to a numbered function key, then when this function key is typed it  
 14387 shall be mapped to *rhs*. Characters other than digits following a '#' character also  
 14388 represent the function key named by the characters in the *lhs* following the '#' and may be  
 14389 mapped to *rhs*. It is unspecified how function keys are named or what function keys are  
 14390 supported.

14391 For text input mode mappings:



- 14392 When the *lhs* is entered as any part of text entered in open or visual text input modes, the  
14393 action shall be as if the corresponding *rhs* had been entered.
- 14394 If any character in the input text is escaped using a <control>-V character, that character shall  
14395 not be part of a match to an *lhs*.
- 14396 It is unspecified whether the *lhs* argument entered for the **map** or **unmap** commands is  
14397 replaced in this fashion. Regardless of whether or not the replacement occurs, the effect of  
14398 the command shall be as if the replacement had not occurred.
- 14399 If only part of the *lhs* is entered, it is unspecified how long the editor will wait for additional,  
14400 possibly matching characters before treating the already entered characters as not matching the  
14401 *lhs*.
- 14402 The *rhs* characters shall themselves be subject to remapping, unless otherwise specified by the  
14403 **remap** edit option, except that if the characters in *lhs* occur as prefix characters in *rhs*, those  
14404 characters shall not be remapped.
- 14405 On block-mode terminals, the mapping need not occur immediately (for example, it may occur  
14406 after the terminal transmits a group of characters to the system), but it shall achieve the same  
14407 results as if it occurred immediately.
- 14408 *Current line*: Unchanged.
- 14409 *Current column*: Unchanged.
- 14410 **Mark**
- 14411 *Synopsis:* [laddr] ma[*rk*] *character*  
14412 [laddr] k *character*
- 14413 Implementations shall support *character* values of a single lowercase letter of the POSIX locale  
14414 and the characters ' ' and ' ' '; support of other characters is implementation-defined.
- 14415 If executing the **vi m** command, set the specified mark to the current line and 1-based numbered  
14416 character referenced by the current column, if any; otherwise, column position 1.
- 14417 Otherwise, set the specified mark to the specified line and 1-based numbered first non-<blank>  
14418 character in the line, if any; otherwise, the last character in the line, if any; otherwise, column  
14419 position 1.
- 14420 The mark shall remain associated with the line until the mark is reset or the line is deleted. If a  
14421 deleted line is restored by a subsequent **undo** command, any marks previously associated with  
14422 the line, which have not been reset, shall be restored as well. Any use of a mark not associated  
14423 with a current line in the edit buffer shall be an error.
- 14424 The marks ' and ' shall be set as described previously, immediately before the following events  
14425 occur in the editor:
- 14426 1. The use of ' \$ ' as an *ex* address
  - 14427 2. The use of a positive decimal number as an *ex* address
  - 14428 3. The use of a search command as an *ex* address
  - 14429 4. The use of a mark reference as an *ex* address
  - 14430 5. The use of the following open and visual mode commands: <control>-[, %, (, ), [, ], {, }.
  - 14431 6. The use of the following open and visual mode commands: ', **G**, **H**, **L**, **M**, **z** if the current  
14432 line will change as a result of the command

14433 7. The use of the open and visual mode commands: /, ?, N, ', n if the current line or column  
14434 will change as a result of the command

14435 8. The use of the ex mode commands: z, undo, global, v

14436 For rules 1., 2., 3., and 4., the ' and ' marks shall not be set if the ex command is parsed as  
14437 specified by rule 6.a. in **Command Line Parsing in ex** (on page 2573).

14438 For rules 5., 6., and 7., the ' and ' marks shall not be set if the commands are used as motion  
14439 commands in open and visual mode.

14440 For rules 1., 2., 3., 4., 5., 6., 7., and 8., the ' and ' marks shall not be set if the command fails.

14441 The ' and ' marks shall be set as described previously, each time the contents of the edit buffer  
14442 are replaced (including the editing of the initial buffer), if in open or visual mode, or if in ex  
14443 mode and the edit buffer is not empty, before any commands or movements (including  
14444 commands or movements specified by the -c or -t options or the +command argument) are  
14445 executed on the edit buffer. If in open or visual mode, the marks shall be set as if executing the vi  
14446 m command; otherwise, as if executing the ex mark command.

14447 When changing from ex mode to open or visual mode, if the ' and ' marks are not already set,  
14448 the ' and ' marks shall be set as described previously.

14449 *Current line:* Unchanged.

14450 *Current column:* Unchanged.

#### 14451 **Move**

14452 *Synopsis:* [2addr] m[ove] laddr [flags]

14453 Move the specified lines after the specified destination line. A destination of line zero specifies  
14454 that the lines shall be placed at the beginning of the edit buffer. It shall be an error if the  
14455 destination line is within the range of lines to be moved.

14456 *Current line:* Set to the last of the moved lines.

14457 *Current column:* Set to non-<blank>.

#### 14458 **Next**

14459 *Synopsis:* n[ext][!][+command][file ...]

14460 If no '!' is appended to the command name, and the edit buffer has been modified since the  
14461 last complete write, it shall be an error, unless the file is successfully written as specified by the  
14462 **autowrite** option.

14463 If one or more files is specified:

- 14464 1. Set the argument list to the specified file names.
- 14465 2. Set the current argument list reference to be the first entry in the argument list.
- 14466 3. Set the current path name to the first file name specified.

14467 Otherwise:

- 14468 1. It shall be an error if there are no more file names in the argument list after the file name  
14469 currently referenced.
- 14470 2. Set the current path name and the current argument list reference to the file name after the  
14471 file name currently referenced in the argument list.

14472 Replace the contents of the edit buffer with the contents of the file named by the current path  
14473 name. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

14474 This command shall be affected by the **autowrite** and **writeany** edit options.

14475 The *+command* option shall be <blank> character-delimited; <blank> characters can be escaped  
14476 by preceding them with a backslash character. The *+command* shall be interpreted as an *ex*  
14477 command immediately after the contents of the edit buffer have been replaced and the current  
14478 line and column have been set.

14479 *Current line*: Set as described for the **edit** command.

14480 *Current column*: Set as described for the **edit** command.

## 14481 **Number**

14482 *Synopsis*: [2addr] nu[mber][count][flags]  
14483 [2addr] #[count][flags]

14484 These commands shall be equivalent to the *ex* command:

14485 [2addr] p[rint][count] #[flags]

14486 See **Print** (on page 2590).

## 14487 **Open**

14488 *Synopsis*: [1addr] o[pen] /pattern/ [flags]

14489 This command need not be supported on block-mode terminals or terminals with insufficient  
14490 capabilities. If standard input, standard output, or standard error are not terminal devices, the  
14491 results are unspecified.

14492 Enter open mode.

14493 The trailing delimiter can be omitted from pattern at the end of the command line. If pattern is  
14494 empty (for example, "//") or not specified, the last regular expression used in the editor shall be  
14495 used as the pattern. The pattern can be delimited by slashes (shown in the Synopsis), as well as  
14496 any alphanumeric, or non-<blank> character other than backslash, vertical line, double quote, or  
14497 <newline> character.

14498 *Current line*: Set to the specified line.

14499 *Current column*: Set to non-<blank>.

## 14500 **Preserve**

14501 *Synopsis*: pre[serve]

14502 Save the edit buffer in a form that can later be recovered by using the **-r** option or by using the *ex*  
14503 **recover** command. After the file has been preserved, a mail message shall be sent to the user.  
14504 This message shall be readable by invoking the *mailx* utility. The message shall contain the name  
14505 of the file, the time of preservation, and an *ex* command that could be used to recover the file.  
14506 Additional information may be included in the mail message.

14507 *Current line*: Unchanged.

14508 *Current column*: Unchanged.

14509 **Print**14510 *Synopsis:* [2addr] p[rint][count][flags]14511 Write the addressed lines. The behavior is unspecified if the number of columns on the display is  
14512 less than the number of columns required to write any single character in the lines being written.14513 Non-printable characters, except for the <tab> character, shall be written as implementation-  
14514 defined multi-character sequences.14515 If the # flag is specified or the **number** edit option is set, each line shall be preceded by its line  
14516 number in the following format:

14517 "%6dΔΔ", &lt;line number&gt;

14518 If the **l** flag is specified or the **list** edit option is set:14519 1. The characters listed in the Base Definitions volume of IEEE Std. 1003.1-200x, Table 5-1,  
14520 Escape Sequences and Associated Actions shall be written as the corresponding escape  
14521 sequence.14522 2. Non-printable characters not in the Base Definitions volume of IEEE Std. 1003.1-200x,  
14523 Table 5-1, Escape Sequences and Associated Actions shall be written as one three-digit  
14524 octal number (with a preceding backslash) for each byte in the character (most significant  
14525 byte first). If the size of a byte on the system is greater than 9 bits, the format used for non-  
14526 printable characters is implementation-defined.14527 3. The end of each line shall be marked with a '\$', and literal '\$' characters within the line  
14528 shall be written with a preceding backslash.14529 Long lines shall be folded; the length at which folding occurs is unspecified, but should be  
14530 appropriate for the output terminal, considering the number of columns of the terminal.14531 If a line is folded, and the **l** flag is not specified and the **list** edit option is not set, it is unspecified  
14532 whether a multi-column character at the folding position is separated; it shall not be discarded.14533 *Current line:* Set to the last written line.14534 *Current column:* Unchanged if the current line is unchanged; otherwise, set to non-<blank>.14535 **Put**14536 *Synopsis:* [laddr] pu[t][buffer]14537 Append text from the specified buffer (by default, the unnamed buffer) to the specified line; line  
14538 zero specifies that the text shall be placed at the beginning of the edit buffer. Each portion of a  
14539 line in the buffer shall become a new line in the edit buffer, regardless of the mode of the buffer.14540 *Current line:* Set to the last line entered into the edit buffer.14541 *Current column:* Set to non-<blank>.14542 **Quit**14543 *Synopsis:* q[uit][!]

14544 If no '!' is appended to the command name:

14545 1. If the edit buffer has been modified since the last complete write, it shall be an error.

14546 2. If there are file names in the argument list after the file name currently referenced, and the  
14547 last command was not a **quit**, **wq**, **xit**, or **ZZ** (see **Exit** (on page 3235)) command, it shall be  
14548 an error.

14549 Otherwise, terminate the editing session.

## 14550 **Read**

14551 *Synopsis:* `[laddr] r[ead][!][file]`

14552 If '!' is not the first non-<blank> character to follow the command name, a copy of the  
14553 specified file shall be appended into the edit buffer after the specified line; line zero specifies that  
14554 the copy shall be placed at the beginning of the edit buffer. The number of lines and bytes read  
14555 shall be written. If no *file* is named, the current path name shall be the default. If there is no  
14556 current path name, then *file* shall become the current path name. If there is no current path name  
14557 or *file* operand, it shall be an error. Specifying a *file* that is not of type regular shall have  
14558 unspecified results.

14559 Otherwise, if *file* is preceded by '!', the rest of the line after the '!' shall have '%', '#', and  
14560 '!' characters expanded as described in **Command Line Parsing in ex** (on page 2573).

14561 The *ex* utility shall then pass two arguments to the program named by the *shell* edit option; the  
14562 first shall be `-c` and the second shall be the expanded arguments to the **read** command as a  
14563 single argument. The standard input of the program shall be set to the standard input of the *ex*  
14564 program when it was invoked. The standard error and standard output of the program shall be  
14565 appended into the edit buffer after the specified line.

14566 Each line in the copied file or program output (as delimited by <newline> characters or the end  
14567 of the file or output if it is not immediately preceded by a <newline> character), shall be a  
14568 separate line in the edit buffer. Any occurrences of <carriage-return> and <newline> character  
14569 pairs in the output shall be treated as single <newline> characters.

14570 The special meaning of the '!' following the **read** command can be overridden by escaping it  
14571 with a backslash character.

14572 *Current line:* If no lines are added to the edit buffer, unchanged. Otherwise, if in open or visual  
14573 mode, set to the first line entered into the edit buffer. Otherwise, set to the last line entered into  
14574 the edit buffer.

14575 *Current column:* Set to non-<blank>.

## 14576 **Recover**

14577 *Synopsis:* `rec[over][!] file`

14578 If no '!' is appended to the command name, and the edit buffer has been modified since the  
14579 last complete write, it shall be an error.

14580 If no *file* operand is specified, then the current path name shall be used. If there is no current  
14581 path name or *file* operand, it shall be an error.

14582 If no recovery information has previously been saved about *file*, the **recover** command shall  
14583 behave identically to the **edit** command, and an informational message to this effect shall be  
14584 written.

14585 Otherwise, set the current path name to *file*, and replace the current contents of the edit buffer  
14586 with the recovered contents of *file*. If there are multiple instances of the file to be recovered, the  
14587 one most recently saved shall be recovered, and an informational message that there are  
14588 previous versions of the file that can be recovered shall be written. The editor shall behave as if  
14589 the contents of the edit buffer have already been modified.

14590 *Current file:* Set as described for the **edit** command.

14591 *Current column*: Set as described for the **edit** command.

### 14592 **Rewind**

14593 *Synopsis*: `rew[ind][!]`

14594 If no `'!'` is appended to the command name, and the edit buffer has been modified since the  
14595 last complete write, it shall be an error, unless the file is successfully written as specified by the  
14596 **autowrite** option.

14597 If the argument list is empty, it shall be an error.

14598 The current argument list reference and the current path name shall be set to the first file name  
14599 in the argument list.

14600 Replace the contents of the edit buffer with the contents of the file named by the current path  
14601 name. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

14602 This command shall be affected by the **autowrite** and **writeany** edit options.

14603 *Current line*: Set as described for the **edit** command.

14604 *Current column*: Set as described for the **edit** command.

### 14605 **Set**

14606 *Synopsis*: `se[t][option=[value]] ...][nooption ...][option? ...][all]`

14607 When no arguments are specified, write the value of the **term** edit option and those options  
14608 whose values have been changed from the default settings; when the argument *all* is specified,  
14609 write all of the option values.

14610 Giving an option name followed by the character `'?'` shall cause the current value of that  
14611 option to be written. The `'?'` can be separated from the option name by zero or more `<blank>`  
14612 characters. The `'?'` shall be necessary only for Boolean valued options. Boolean options can be  
14613 given values by the form **set option** to turn them on or **set nooption** to turn them off; string and  
14614 numeric options can be assigned by the form **set option=value**. Any `<blank>` characters in strings  
14615 can be included as is by preceding each `<blank>` with an escaping backslash. More than one  
14616 option can be set or listed by a single set command by specifying multiple arguments, each  
14617 separated from the next by one or more `<blank>` characters.

14618 See **Edit Options in ex** (on page 2602) for details about specific options.

14619 *Current line*: Unchanged.

14620 *Current column*: Unchanged.

### 14621 **Shell**

14622 *Synopsis*: `sh[ell]`

14623 Invoke the program named in the **shell** edit option with the single argument `-i` (interactive  
14624 mode). Editing shall be resumed when the program exits.

14625 *Current line*: Unchanged.

14626 *Current column*: Unchanged.

14627 **Source**14628 *Synopsis:* `so[urce] file`14629 Read and execute *ex* commands from *file*. Lines in the file that contain no characters or only  
14630 <blank> characters shall be ignored.14631 *Current line:* As specified for the individual *ex* commands.14632 *Current column:* As specified for the individual *ex* commands.14633 **Substitute**14634 *Synopsis:* `[2addr] s[substitute][/pattern/repl][options][count][flags]`14635 `[2addr] &[options][count][flags]`14636 `[2addr] ~[options][count][flags]`14637 Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See  
14638 **Regular Expressions in ex** (on page 2601) and **Replacement Strings in ex** (on page 2602).) Any  
14639 non-alphabetic, non-<blank> delimiter other than '\\', '|', double quote, or <newline>  
14640 character can be used instead of '/'. Backslash characters can be used to escape delimiters,  
14641 backslash characters, and other special characters.14642 The trailing delimiter can be omitted from *pattern* or from *repl* at the end of the command line. If  
14643 both *pattern* and *repl* are not specified or are empty (for example, "//"), the last **s** command  
14644 shall be repeated. If only *pattern* is not specified or is empty, the last regular expression used in  
14645 the editor shall be used as the pattern. If only *repl* is not specified or is empty, the pattern shall be  
14646 replaced by nothing. If the entire replacement pattern is '%', the last replacement pattern to an  
14647 **s** command shall be used.14648 Entering a <carriage-return> in *repl* (which requires an escaping backslash in *ex* mode and an  
14649 escaping <control>-V in open or *vi* mode) shall split the line at that point, creating a new line in  
14650 the edit buffer. The <carriage-return> shall be discarded.14651 If options include the letter 'g' (**global**), all non-overlapping instances of the pattern in the line  
14652 shall be replaced.14653 If options includes the letter 'c' (**confirm**), then before each substitution the line shall be  
14654 written; the written line shall reflect all previous substitutions. On the following line, <space>  
14655 characters shall be written beneath the characters from the line that are before the *pattern* to be  
14656 replaced, and '^' characters written beneath the characters included in the *pattern* to be  
14657 replaced. The *ex* utility shall then wait for a response from the user. An affirmative response  
14658 shall cause the substitution to be done, while any other input shall not make the substitution. An  
14659 affirmative response shall consist of a line with the affirmative response (as defined by the  
14660 current locale) at the beginning of the line. This line shall be subject to editing in the same way as  
14661 the *ex* command line.14662 If interrupted (see the ASYNCHRONOUS EVENTS section), any modifications confirmed by the  
14663 user shall be preserved in the edit buffer after the interrupt.14664 If the remembered search direction is not set, the **s** command shall set it to forward.14665 In the second Synopsis, the **&** command shall repeat the previous substitution, as if the **&**  
14666 command were replaced by:14667 `s/pattern/repl/`14668 where *pattern* and *repl* are as specified in the previous **s**, **&**, or **~** command.

14669 In the third Synopsis, the ~ command shall repeat the previous substitution, as if the '~' were  
14670 replaced by:

14671 *s/pattern/repl/*

14672 where *pattern* shall be the last regular expression specified to the editor, and *repl* shall be from  
14673 the previous substitution (including & and ~) command.

14674 These commands shall be affected by the *LC\_MESSAGES* environment variable.

14675 *Current line*: Set to the last line in which a substitution occurred, or, unchanged if no  
14676 substitution occurred.

14677 *Current column*: Set to non-<blank>.

## 14678 Suspend

14679 *Synopsis:*    su[spend][!]  
14680               st[op][!]

14681 Allow control to return to the invoking process; *ex* shall suspend itself as if it had received the  
14682 SIGTSTP signal. The suspension shall occur only if job control is enabled in the invoking shell  
14683 (see the description of *set -m*).

14684 These commands shall be affected by the **autowrite** and **writeany** edit options.

14685 The current **susp** character (see *stty*) shall have the same affect as the **suspend** command.

## 14686 Tag

14687 *Synopsis:*    ta[g][!] *tagstring*

14688 The results are unspecified if the format of a tags file is not as specified by the *ctags* utility (see  
14689 *ctags*) description.

14690 The **tag** command shall search for *tagstring* in the tag files referred to by the **tag** edit option, in  
14691 the order they are specified, until a reference to *tagstring* is found. Files shall be searched from  
14692 beginning to end. If no reference is found, it shall be an error and an error message to this effect  
14693 shall be written. If the reference is not found, or if an error occurs while processing a file referred  
14694 to in the **tag** edit option, it shall be an error, and an error message shall be written at the first  
14695 occurrence of such an error.

14696 Otherwise, if the tags file contained a pattern, the pattern shall be treated as a regular expression  
14697 used in the editor; for example, for the purposes of the **s** command.

14698 If the *tagstring* is in a file with a different name than the current path name, set the current path  
14699 name to the name of that file, and replace the contents of the edit buffer with the contents of that  
14700 file. In this case, if no '!' is appended to the command name, and the edit buffer has been  
14701 modified since the last complete write, it shall be an error, unless the file is successfully written  
14702 as specified by the **autowrite** option.

14703 This command shall be affected by the **autowrite**, **tag**, **taglength**, and **writeany** edit options.

14704 *Current line*: If the tags file contained a line number, set to that line number. If the line number is  
14705 larger than the last line in the edit buffer, an error message shall be written and the current line  
14706 shall be set as specified for the **edit** command.

14707 If the tags file contained a pattern, set to the first occurrence of the pattern in the file. If no  
14708 matching pattern is found, an error message shall be written and the current line shall be set as  
14709 specified for the **edit** command.



14710 *Current column:* If the tags file contained a line-number reference and that line-number was not  
 14711 larger than the last line in the edit buffer, or if the tags file contained a pattern and that pattern  
 14712 was found, set to non-<blank>. Otherwise, set as specified for the **edit** command.

### 14713 **Unabbreviate**

14714 *Synopsis:*    una[bbrev] lhs

14715 If *lhs* is not an entry in the current list of abbreviations (see **Abbreviate** (on page 2580)), it shall  
 14716 be an error. Otherwise, delete *lhs* from the list of abbreviations.

14717 *Current line:* Unchanged.

14718 *Current column:* Unchanged.

### 14719 **Undo**

14720 *Synopsis:*    u[ndo]

14721 Reverse the changes made by the last command that modified the contents of the edit buffer,  
 14722 including **undo**. For this purpose, the **global**, **v**, **open**, and **visual** commands, and commands  
 14723 resulting from buffer executions and mapped character expansions, are considered single  
 14724 commands.

14725 If no action that can be undone preceded the **undo** command, it shall be an error.

14726 If the **undo** command restores lines that were marked, the mark shall also be restored unless it  
 14727 was reset subsequent to the deletion of the lines.

14728 *Current line:*

- 14729     1. If lines are added or changed in the file, set to the first line added or changed.
- 14730     2. Set to the line before the first line deleted, if it exists.
- 14731     3. Set to 1 if the edit buffer is not empty.
- 14732     4. Set to zero.

14733 *Current column:* Set to non-<blank>.

### 14734 **Unmap**

14735 *Synopsis:*    unm[ap][!] lhs

14736 If '!' is appended to the command name, and if *lhs* is not an entry in the list of text input mode  
 14737 map definitions, it shall be an error. Otherwise, delete *lhs* from the list of text input mode map  
 14738 definitions.

14739 If no '!' is appended to the command name, and if *lhs* is not an entry in the list of command  
 14740 mode map definitions, it shall be an error. Otherwise, delete *lhs* from the list of command mode  
 14741 map definitions.

14742 *Current line:* Unchanged.

14743 *Current column:* Unchanged.

- 14744       **Version**
- 14745       *Synopsis:*    ve[rsion]
- 14746       Write a message containing version information for the editor. The format of the message is  
14747       unspecified.
- 14748       *Current line:* Unchanged.
- 14749       *Current column:* Unchanged.
- 14750       **Visual**
- 14751       *Synopsis:*    [*laddr*] vi[sual][*type*][*count*][*flags*]
- 14752       If *ex* is currently in open or visual mode, the Synopsis and behavior of the visual command shall  
14753       be the same as the **edit** command, as specified by **Edit** (on page 2582).
- 14754       Otherwise, this command need not be supported on block-mode terminals or terminals with  
14755       insufficient capabilities. If standard input, standard output, or standard error are not terminal  
14756       devices, the results are unspecified.
- 14757       If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in  
14758       **window** (on page 2609)). If the '^' type character was also specified, the **window** edit option  
14759       shall be set before being used by the type character.
- 14760       Enter visual mode. If *type* is not specified, it shall be as if a *type* of '+' was specified. The *type*  
14761       shall cause the following effects:
- 14762       +   Place the beginning of the specified line at the top of the display.
- 14763       -   Place the end of the specified line at the bottom of the display.
- 14764       .   Place the beginning of the specified line in the middle of the display.
- 14765       ^   If the specified line is less than or equal to the value of the **window** edit option, set the line  
14766       to 1; otherwise, decrement the line by the value of the **window** edit option minus 1. Place  
14767       the beginning of this line as close to the bottom of the displayed lines as possible, while still  
14768       displaying the value of the **window** edit option number of lines.
- 14769       *Current line:* Set to the specified line.
- 14770       *Current column:* Set to non-<blank>.
- 14771       **Write**
- 14772       *Synopsis:*    [*2addr*] w[rite][!][>>][*file*]  
14773                   [*2addr*] w[rite][!][*file*]  
14774                   [*2addr*] wq[!][>>][*file*]
- 14775       If no lines are specified, the lines shall default to the entire file.
- 14776       The command **wq** shall be equivalent to a **write** command followed by a **quit** command; **wq!**  
14777       shall be equivalent to **write!** followed by **quit**. In both cases, if the **write** command fails, the  
14778       **quit** shall not be attempted.
- 14779       If the command name is not followed by one or more <blank> characters, or *file* is not preceded  
14780       by a '!' character, the **write** shall be to a file.
- 14781       1. If the >> argument is specified, and the file already exists, the lines shall be appended to  
14782       the file instead of replacing its contents. If the >> argument is specified, and the file does  
14783       not already exist, it is unspecified whether the write shall proceed as if the >> argument

- 14784 had not been specified or if the write shall fail.
- 14785 2. If the **readonly** edit option is set (see **readonly** (on page 2606)), the **write** shall fail.
- 14786 3. If *file* is specified, and is not the current path name, and the file exists, the **write** shall fail.
- 14787 4. If *file* is not specified, the current path name shall be used. If there is no current path name,  
14788 the **write** command shall fail.
- 14789 5. If the current path name is used, and the current path name has been changed by the **file**  
14790 or **read** commands, and the file exists, the **write** shall fail. If the **write** is successful,  
14791 subsequent **writes** shall not fail for this reason (unless the current path name is changed  
14792 again).
- 14793 6. If the whole edit buffer is not being written, and the file to be written exists, the **write** shall  
14794 fail.
- 14795 For rules 1., 2., 4., and 5., the **write** can be forced by appending the character '!' to the  
14796 command name.
- 14797 For rules 2., 4., and 5., the **write** can be forced by setting the **writeany** edit option.
- 14798 Additional, implementation-defined tests may cause the **write** to fail.
- 14799 If the edit buffer is empty, a file without any contents shall be written.
- 14800 An informational message shall be written noting the number of lines and bytes written.
- 14801 Otherwise, if the command is followed by one or more <blank> characters, and file is preceded  
14802 by '!', the rest of the line after the '!' shall have '%', '#', and '!' characters expanded as  
14803 described in **Command Line Parsing in ex** (on page 2573).
- 14804 The *ex* utility shall then pass two arguments to the program named by the **shell** edit option; the  
14805 first shall be **-c** and the second shall be the expanded arguments to the **write** command as a  
14806 single argument. The specified lines shall be written to the standard input of the command. The  
14807 standard error and standard output of the program, if any, shall be written as described for the  
14808 **print** command. If the last character in that output is not a <newline> character, a <newline>  
14809 shall be written at the end of the output.
- 14810 The special meaning of the '!' following the **write** command can be overridden by escaping it  
14811 with a backslash character.
- 14812 *Current line:* Unchanged.
- 14813 *Current column:* Unchanged.
- 14814 **Write and Exit**
- 14815 *Synopsis:* [2*addr*] x[*it*][!][*file*]
- 14816 If the edit buffer has not been modified since the last complete **write**, **xit** shall be equivalent to  
14817 the **quit** command, or if a '!' is appended to the command name, to **quit!**.
- 14818 Otherwise, **xit** shall be equivalent to the **wq** command, or if a '!' is appended to the command  
14819 name, to **wq!**.
- 14820 *Current line:* Unchanged.
- 14821 *Current line:* Unchanged.

14822 **Yank**14823 *Synopsis:* `[laddr] ya[nk][buffer][count]`14824 Copy the specified lines to the specified buffer (by default, the unnamed buffer), which shall  
14825 become a line-mode buffer.14826 *Current line:* Unchanged.14827 *Current line:* Unchanged.14828 **Adjust Window**14829 *Synopsis:* `[laddr] z[!][type ...][count][flags]`14830 If no line is specified, the current line shall be the default; if *type* is omitted as well, the current  
14831 line value shall first be incremented by 1. If incrementing the current line would cause it to be  
14832 greater than the last line in the edit buffer, it shall be an error.14833 If there are <blank> characters between the *type* argument and the preceding *z* command name  
14834 or optional '!' character, it shall be an error.14835 If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in  
14836 **window** (on page 2609)). If *count* is omitted, it shall default to 2 times the value of the **scroll** edit  
14837 option, or if ! was specified, the number of lines in the display minus 1.14838 If *type* is omitted, then *count* lines starting with the specified line shall be written. Otherwise,  
14839 *count* lines starting with the line specified by the *type* argument shall be written.14840 The *type* argument shall change the lines to be written. The possible values of *type* are as follows:

14841 – The specified line shall be decremented by the following value:

14842  $((\text{number of ``-'' characters}) \times \text{count}) - 1$ 14843 If the calculation would result in a number less than 1, it shall be an error. Write lines from  
14844 the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit  
14845 buffer has been written.

14846 + The specified line shall be incremented by the following value:

14847  $((\text{number of ``+'' characters}) - 1) \times \text{count} + 1$ 14848 If the calculation would result in a number greater than the last line in the edit buffer, it  
14849 shall be an error. Write lines from the edit buffer, starting at the new value of line, until  
14850 *count* lines or the last line in the edit buffer has been written.14851 =, . If more than a single '.' or '=' is specified, it shall be an error. The following steps shall be  
14852 taken:14853 1. If *count* is zero, nothing shall be written.14854 2. Write as many of the *N* lines before the current line in the edit buffer as exist. If *count*  
14855 or '!' was specified, *N* shall be:14856  $(\text{count} - 1) / 2$ 14857 Otherwise, *N* shall be:14858  $(\text{count} - 3) / 2$ 14859 If *N* is a number less than 3, no lines shall be written.

- 14860 3. If '=' was specified as the type character, write a line consisting of the smaller of the  
14861 number of columns in the display divided by two, or 40 '-' characters.
- 14862 4. Write the current line.
- 14863 5. Repeat step 3.
- 14864 6. Write as many of the *N* lines after the current line in the edit buffer as exist. *N* shall be  
14865 defined as in step 2. If *N* is a number less than 3, no lines shall be written. current line  
14866 in the edit buffer as exist. If count is less than 3, no lines shall be written.
- 14867 ^ The specified line shall be decremented by the following value:  
14868  $((\text{number of ``^`` characters}) + 1) \times \text{count} - 1$
- 14869 If the calculation would result in a number less than 1, it shall be an error. Write lines from  
14870 the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit  
14871 buffer has been written.
- 14872 *Current line*: Set to the last line written, unless the type is =, in which case, set to the specified  
14873 line.
- 14874 *Current column*: Set to non-<blank>.
- 14875 **Escape**
- 14876 *Synopsis*: ! *command*  
14877 [ *addr* ] ! *command*
- 14878 The contents of the line after the '!' shall have '%', '#', and '!' characters expanded as  
14879 described in **Command Line Parsing in ex** (on page 2573). If the expansion causes the text of the  
14880 line to change, it shall be redisplayed, preceded by a single '!' character.
- 14881 The *ex* utility shall execute the program named by the **shell** edit option. It shall pass two  
14882 arguments to the program; the first shall be **-c**, and the second shall be the expanded arguments  
14883 to the ! command as a single argument.
- 14884 If no lines are specified, the standard input, standard output, and standard error of the program  
14885 shall be set to the standard input, standard output, and standard error of the *ex* program when it  
14886 was invoked. In addition, a warning message shall be written if the edit buffer has been  
14887 modified since the last complete write, and the **warn** edit option is set.
- 14888 If lines are specified, they shall be passed to the program as standard input, and the standard  
14889 output and standard error of the program shall replace those lines in the edit buffer. Each line in  
14890 the program output (as delimited by <newline> characters or the end of the output if it is not  
14891 immediately preceded by a <newline> character), shall be a separate line in the edit buffer. Any  
14892 occurrences of <carriage-return> and <newline> character pairs in the output shall be treated as  
14893 single <newline> characters. The specified lines shall be copied into the unnamed buffer before  
14894 they are replaced, and the unnamed buffer shall become a line-mode buffer.
- 14895 If in *ex* mode, a single '!' character shall be written when the program completes.
- 14896 This command shall be affected by the **shell** and **warn** edit options. If no lines are specified, this  
14897 command shall be affected by the **autowrite** and **writeany** edit options. If lines are specified, this  
14898 command shall be affected by the **autoprint** edit option.
- 14899 *Current line*:
- 14900 1. If no lines are specified, unchanged.

- 14901           2. Otherwise, set to the last line read in, if any lines are read in.
- 14902           3. Otherwise, set to the line before the first line of the lines specified, if that line exists.
- 14903           4. Otherwise, set to the first line of the edit buffer if the edit buffer is not empty.
- 14904           5. Otherwise, set to zero.
- 14905           *Current column*: If no lines are specified, unchanged. Otherwise, set to non-<blank>.
- 14906           **Shift Left**
- 14907           *Synopsis*:     [*2addr*] <[< ...][*count*][*flags*]
- 14908           Shift the specified lines to the start of the line; the number of column positions to be shifted shall be the number of command characters times the value of the **shiftwidth** edit option. Only leading <blank> characters shall be deleted or changed into other <blank> characters in shifting; other characters shall not be affected.
- 14909
- 14910
- 14911
- 14912           Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.
- 14913
- 14914           This command shall be affected by the **autoprint** edit option.
- 14915           *Current line*: Set to the last line in the lines specified.
- 14916           *Current column*: Set to non-<blank>.
- 14917           **Shift Right**
- 14918           *Synopsis*:     [*2addr*] >[> ...][*count*][*flags*]
- 14919           Shift the specified lines away from the start of the line; the number of column positions to be shifted shall be the number of command characters times the value of the **shiftwidth** edit option. The shift shall be accomplished by adding <blank> characters as a prefix to the line or changing leading <blank> characters into other <blank> characters. Empty lines shall not be changed.
- 14920
- 14921
- 14922
- 14923           Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.
- 14924
- 14925           This command shall be affected by the **autoprint** edit option.
- 14926           *Current line*: Set to the last line in the lines specified.
- 14927           *Current column*: Set to non-<blank>.
- 14928           <control>-D
- 14929           *Synopsis*:     <control>-D
- 14930           Write the next *n* lines, where *n* is the minimum of the values of the **scroll** edit option and the number of lines after the current line in the edit buffer. If the current line is the last line of the edit buffer it shall be an error.
- 14931
- 14932
- 14933           *Current line*: Set to the last line written.
- 14934           *Current column*: Set to non-<blank>.

14935 **Write Line Number**14936 *Synopsis:* [1addr] = [flags]14937 If *line* is not specified, it shall default to the last line in the edit buffer. Write the line number of  
14938 the specified line.14939 *Current line:* Unchanged.14940 *Current column:* Unchanged.14941 **Execute**14942 *Synopsis:* [2addr] @ *buffer*14943 [2addr] \* *buffer*14944 If no buffer is specified or is specified as '@' or '\* ', the last buffer executed shall be used. If no  
14945 previous buffer has been executed, it shall be an error.14946 For each line specified by the addresses, set the current line ('.') to the specified line, and  
14947 execute the contents of the named *buffer* (as they were at the time the @ command was executed)  
14948 as *ex* commands. For each line of a line-mode buffer, and all but the last line of a character-mode  
14949 buffer, the *ex* command parser shall behave as if the line was terminated by a <newline>  
14950 character.14951 If an error occurs during this process, or a line specified by the addresses does not exist when the  
14952 current line would be set to it, or more than a single line was specified by the addresses, and the  
14953 contents of the edit buffer are replaced (for example, by the *ex:edit* command) an error message  
14954 shall be written, and no more commands resulting from the execution of this command shall be  
14955 processed.14956 *Current line:* As specified for the individual *ex* commands.14957 *Current column:* As specified for the individual *ex* commands.14958 **Regular Expressions in ex**14959 The *ex* utility shall support regular expressions that are a superset of the basic regular  
14960 expressions described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3, Basic  
14961 Regular Expressions. A null regular expression ("//") shall be equivalent to the last regular  
14962 expression encountered.14963 Regular expressions can be used in addresses to specify lines and, in some commands (for  
14964 example, the **substitute** command), to specify portions of a line to be substituted.

14965 The following constructs can be used to enhance the basic regular expressions:

14966 \< < Match the beginning of a *word*. (See the definition of *word* at the beginning of **Command**  
14967 **Descriptions in ex** (on page 2578).)14968 \< > Match the end of a *word*.14969 ~ Match the replacement part of the last **substitute** command. The tilde ('~') character can  
14970 be escaped in a regular expression to become a normal character with no special meaning.  
14971 The backslash shall be discarded.14972 When the editor option **magic** is not set, the only characters with special meanings shall be '^'  
14973 at the beginning of a pattern, '\$' at the end of a pattern, and '\\'. The characters '.', '\*',  
14974 '[', and '~' shall be treated as ordinary characters unless preceded by a '\\'; when preceded  
14975 by a '\\' they shall regain their special meaning, or in the case of backslash, be handled as a  
14976 single backslash. Backslashes used to escape other characters shall be discarded.

14977 **Replacement Strings in ex**

14978 The character '&' ('\&' if the editor option **magic** is not set) in the replacement string shall  
 14979 stand for the text matched by the pattern to be replaced. The character '~' ('\~' if **magic** is not  
 14980 set) shall be replaced by the replacement part of the previous **substitute** command. The  
 14981 sequence '\n', where *n* is an integer, shall be replaced by the text matched by the pattern  
 14982 enclosed in the *n*th set of parentheses '\(' and '\)'.  
 14983

14984 The strings '\l', '\u', '\L', and '\U' can be used to modify the case of elements in the  
 14985 replacement string (using the '\&' or "\"digit) notation. The string '\l' ('\u') shall cause  
 14986 the character that follows to be converted to lowercase (uppercase). The string '\L' ('\U')  
 14987 shall cause all characters subsequent to it to be converted to lowercase (uppercase) as they are  
 14988 inserted by the substitution until the string '\e' or '\E', or the end of the replacement string,  
 is encountered.

14989 Otherwise, any character following a backslash shall be treated as that literal character, and the  
 14990 escaping backslash shall be discarded.

14991 An example of case conversion with the **s** command is as follows:

```
14992 :p
14993 The cat sat on the mat.
14994 :s/\<.at\>/\u&/gp
14995 The Cat Sat on the Mat.
14996 :s/S\(.*\)M/S\U\1\eM/p
14997 The Cat SAT ON THE Mat.
```

14998 **Edit Options in ex**

14999 The **ex** utility has a number of options that modify its behavior. These options have default  
 15000 settings, which can be changed using the **set** command.

15001 Options are Boolean unless otherwise specified.

15002 **autoindent, ai**

15003 [Default *unset*]

15004 If **autoindent** is set, each line in input mode shall be indented (using first as many <tab>  
 15005 characters as possible, as determined by the editor option **tabstop**, and then using <space>  
 15006 characters) to align with another line, as follows:

- 15007 1. If in open or visual mode and the text input is part of a line-oriented command (see the  
 15008 EXTENDED DESCRIPTION in *vi*), align to the first column. Otherwise, if in open or  
 15009 visual mode, indentation for each line shall be set as follows:
  - 15010 a. If a line was previously inserted as part of this command, it shall be set to the  
 15011 indentation of the last inserted line by default, or as otherwise specified for the  
 15012 <control>-D character in **Input Mode Commands in vi** (on page 3235).
  - 15013 b. Otherwise, it shall be set to the indentation of the previous current line, if any;  
 15014 otherwise, to the first column.
- 15015 2. For the **ex a, i, and c** commands, indentation for each line shall be set as follows:
  - 15016 a. If a line was previously inserted as part of this command, it shall be set to the  
 15017 indentation of the last inserted line by default, or as otherwise specified for the *eof*  
 15018 character in **Scroll** (on page 2577).



- 15019           b. Otherwise, if the command is the **ex a** command, it shall be set to the line appended  
15020           after, if any; otherwise to the first column.
- 15021           c. Otherwise, if the command is the **ex i** command, it shall be set to the line inserted  
15022           before, if any; otherwise to the first column.
- 15023           d. Otherwise, if the command is the **ex c** command, it shall be set to the indentation of  
15024           the line replaced.

### 15025           **autoprint, ap**

15026           [Default *set*]

15027           If **autoprint** is set, the current line shall be written after each **ex** command that modifies the  
15028           contents of the current edit buffer, and after each **tag** command for which the tag search pattern  
15029           was found or tag line number was valid, unless:

- 15030           1. The command was executed while in open or visual mode.
- 15031           2. The command was executed as part of a **global** or **v** command or @ buffer execution.
- 15032           3. The command was the form of the **read** command that reads a file into the edit buffer.
- 15033           4. The command was the **append**, **change**, or **insert** command.
- 15034           5. The command was not terminated by a <newline> character.
- 15035           6. The current line shall be written by a flag specified to the command; for example, **delete #**  
15036           shall write the current line as specified for the flag modifier to the **delete** command, and  
15037           not as specified by the **autoprint** edit option.

### 15038           **autowrite, aw**

15039           [Default *unset*]

15040           If **autowrite** is set, and the edit buffer has been modified since it was last completely written to  
15041           any file, the contents of the edit buffer shall be written as if the **ex write** command had been  
15042           specified without arguments, before each command affected by the **autowrite** edit option is  
15043           executed. Appending the character '!' to the command name of any of the **ex** commands  
15044           except '!' shall prevent the write. If the write fails, it shall be an error and the command shall  
15045           not be executed.

### 15046           **beautify, bf**

15047 XSI       [Default *unset*]

15048           If **beautify** is set, all non-printable characters, other than <tab>, <newline>, and <form-feed>  
15049           characters, shall be discarded from text read in from files.

### 15050           **directory, dir**

15051           [Default *implementation-defined*]

15052           The value of this option specifies the directory in which the editor buffer is to be placed. If this  
15053           directory is not writable by the user, the editor shall quit.

- 15054        **edcompatible, ed**
- 15055        [Default *unset*]
- 15056        Causes the presence of **g** and **c** suffixes on substitute commands to be remembered, and toggled  
15057        by repeating the suffixes.
- 15058        **errorbells, eb**
- 15059        [Default *unset*]
- 15060        If the editor is in *ex* mode, and the terminal does not support a standout mode (such as inverse  
15061        video), and **errorbells** is set, error messages shall be preceded by alerting the terminal.
- 15062        **exrc**
- 15063        [Default *unset*]
- 15064        If **exrc** is set, *ex* shall access any **.exrc** file in the current directory, as described in **Initialization in**  
15065        **ex and vi** (on page 2569). If **exrc** is not set, *ex* shall ignore any **.exrc** file in the current directory  
15066        during initialization, unless the current directory is that named by the *HOME* environment  
15067        variable.
- 15068        **ignorecase, ic**
- 15069        [Default *unset*]
- 15070        If **ignorecase** is set, characters that have uppercase and lowercase representations shall have  
15071        those representations considered as equivalent for purposes of regular expression comparison.
- 15072        The **ignorecase** edit option shall affect all remembered regular expressions; for example,  
15073        unsetting the **ignorecase** edit option shall cause a subsequent *vi n* command to search for the  
15074        last basic regular expression in a case-sensitive fashion.
- 15075        **lisp**
- 15076        [Default *unset*]
- 15077        **autoindent** mode and the ( ), { }, [[, and ]] commands in visual mode are suitably modified for  
15078        LISP code.
- 15079        **list**
- 15080        [Default *unset*]
- 15081        If **list** is set, edit buffer lines written while in *ex* command mode shall be written as specified for  
15082        the **print** command with the **l** flag specified. In open or visual mode, each edit buffer line shall  
15083        be displayed as specified for the *ex print* command with the **l** flag specified. In open or visual  
15084        text input mode, when the cursor does not rest on any character in the line, it shall rest on the  
15085        ' \$ ' marking the end of the line.

- 15086           **magic**  
 15087           [Default *set*]
- 15088           If **magic** is set, modify the interpretation of characters in regular expressions and substitution  
 15089           replacement strings (see **Regular Expressions in ex** (on page 2601) and **Replacement Strings in**  
 15090           **ex** (on page 2602)).
- 15091           **mesg**  
 15092           [Default *set*]
- 15093           If **mesg** is set, the permission for others to use the **write** or **talk** commands to write to the  
 15094           terminal shall be turned on while in open or visual mode. The shell-level command *mesg n* shall  
 15095           take precedence over any setting of the *ex mesg* option; that is, if *mesg y* was issued before the  
 15096           editor started (or in a shell escape), such as:
- 15097           :!*mesg y*
- 15098           the **mesg** option in *ex* shall suppress incoming messages, but the **mesg** option shall not enable  
 15099           incoming messages if *mesg n* was issued.
- 15100           **number, nu**  
 15101           [Default *unset*]
- 15102           If **number** is set, edit buffer lines written while in *ex* command mode shall be written with line  
 15103           numbers, in the format specified by the **print** command with the # flag specified. In *ex* text input  
 15104           mode, each line shall be preceded by the line number it will have in the file.
- 15105           In open or visual mode, each edit buffer line shall be displayed with a preceding line number, in  
 15106           the format specified by the *ex print* command with the # flag specified. This line number shall  
 15107           not be considered part of the line for the purposes of evaluating the current column; that is,  
 15108           column position 1 shall be the first column position after the format specified by the **print**  
 15109           command.
- 15110           **paragraphs, para**  
 15111           [Default in the POSIX locale *IPLPPPQPP LIpplpipbp*]
- 15112           The **paragraphs** edit option shall define additional paragraph boundaries for the open and visual  
 15113           mode commands. The **paragraphs** edit option can be set to a character string consisting of zero  
 15114           or more character pairs. It shall be an error to set it to an odd number of characters.
- 15115           **prompt**  
 15116           [Default *set*]
- 15117           If **prompt** is set, *ex* command mode input shall be prompted for with a colon (':'); when unset,  
 15118           no prompt shall be written.

15119 **readonly**15120 [Default *see text*]

15121 If **readonly** edit option is set, read-only mode shall be enabled (see **Write** (on page 2596)). The  
 15122 **readonly** edit option shall be initialized to set if either of the following conditions are true:

- 15123 • The command-line option **-R** was specified.
- 15124 • Performing actions equivalent to the *access()* function called with the following arguments  
 15125 indicates that the file lacks write permission:
  - 15126 1. The current path name is used as the *path* argument.
  - 15127 2. The constant **W\_OK** is used as the *amode* argument.

15128 The **readonly** edit option may be initialized to set for other, implementation-defined reasons. |  
 15129 The **readonly** edit option shall not be initialized to unset based on any special privileges of the |  
 15130 user or process. The **readonly** edit option shall be reinitialized each time that the contents of the |  
 15131 edit buffer are replaced (for example, by an **edit** or **next** command) unless the user has explicitly |  
 15132 set it, in which case it shall remain set until the user explicitly unsets it. Once unset, it shall again |  
 15133 be reinitialized each time that the contents of the edit buffer are replaced. |

15134 **redraw**15135 [Default *unset*]

15136 The editor simulates an intelligent terminal on a dumb terminal. (Since this is likely to require a  
 15137 large amount of output to the terminal, it is useful only at high transmission speeds.) |

15138 **remap**15139 [Default *set*]

15140 If **remap** is set, map translation shall allow for maps defined in terms of other maps; translation  
 15141 shall continue until a final product is obtained. If unset, only a one-step translation shall be done.

15142 **report**

15143 [Default 5]

15144 The value of this **report** edit option specifies what number of lines being added, copied, deleted,  
 15145 or modified in the edit buffer will cause an informational message to be written to the user. The  
 15146 following conditions shall cause an informational message. The message shall contain the  
 15147 number of lines added, copied, deleted, or modified, but is otherwise unspecified.

- 15148 • An *ex* or *vi* editor command, other than **open**, **undo**, or **visual**, that modifies at least the value  
 15149 of the **report** edit option number of lines, and which is not part of an *ex* **global** or *v*  
 15150 command, or *ex* or *vi* buffer execution, shall cause an informational message to be written.
- 15151 • An *ex* **yank** or *vi* **y** or **Y** command, that copies at least the value of the **report** edit option plus  
 15152 1 number of lines, and which is not part of an *ex* **global** or *v* command, or *ex* or *vi* buffer  
 15153 execution, shall cause an informational message to be written.
- 15154 • An *ex* **global**, *v*, **open**, **undo**, or **visual** command or *ex* or *vi* buffer execution, that adds or  
 15155 deletes a total of at least the value of the **report** edit option number of lines, and which is not  
 15156 part of an *ex* **global** or *v* command, or *ex* or *vi* buffer execution, shall cause an informational  
 15157 message to be written. (For example, if 3 lines were added and 8 lines deleted during an *ex*  
 15158 **visual** command, 5 would be the number compared against the **report** edit option after the  
 15159 command completed.

15160 **scroll, scr**

15161 [Default (number of lines in the display -1)/2]

15162 The value of the **scroll** edit option shall determine the number of lines scrolled by by the *ex*  
15163 <control>-D and **z** commands. For the *vi* <control>-D and <control>-U commands, it shall be the  
15164 initial number of lines to scroll when no previous <control>-D or <control>-U command has  
15165 been executed.

15166 **sections**15167 [Default in the POSIX locale `NHSHH HUnhsh`]

15168 The **sections** edit option shall define additional section boundaries for the open and visual mode  
15169 commands. The **sections** edit option can be set to a character string consisting of zero or more  
15170 character pairs; it shall be an error to set it to an odd number of characters.

15171 **shell, sh**15172 [Default from the environment variable *SHELL*]

15173 The value of this option shall be a string. The default shall be taken from the *SHELL*  
15174 environment variable. If the *SHELL* environment variable is null or empty, the *sh* (see *sh*) utility  
15175 shall be the default.

15176 **shiftwidth, sw**

15177 [Default 8]

15178 The value of this option shall give the width in columns of an indentation level used during  
15179 autoindentation and by the shift commands (< and >).

15180 **showmatch, sm**15181 [Default *unset*]

15182 The functionality described for the **showmatch** edit option need not be supported on block-  
15183 mode terminals or terminals with insufficient capabilities.

15184 If **showmatch** is set, in open or visual mode, when a ' ) ' or ' } ' is typed, if the matching ' ( ' or  
15185 ' { ' is currently visible on the display, the matching ' ( ' or ' { ' shall be flagged moving the  
15186 cursor to its location for an unspecified amount of time.

15187 **showmode**15188 [Default *unset*]

15189 If **showmode** is set, in open or visual mode, the current mode that the editor is in shall be  
15190 displayed on the last line of the display. Command mode and text input mode shall be  
15191 differentiated; other unspecified modes and implementation-defined information may be  
15192 displayed.

- 15193       **slowopen**  
15194       [Default *unset*]  
15195       If **slowopen** is set during open and visual text input modes, the editor shall not update portions  
15196       of the display other than those screen columns that display the characters entered by the user  
15197       (see **Input Mode Commands in vi** (on page 3235)).
- 15198       **tabstop, ts**  
15199       [Default 8]  
15200       The value of this edit option shall specify the column boundary used by a <tab> character in the  
15201       display (see **autoprint, ap** (on page 2603) and **Input Mode Commands in vi** (on page 3235)).
- 15202       **taglength, tl**  
15203       [Default zero]  
15204       The value of this edit option shall specify the maximum number of characters that are  
15205       considered significant in the user-specified tag name and in the tag name from the tags file. If the  
15206       value is zero, all characters in both tag names shall be significant.
- 15207       **tags**  
15208       [Default *see text*]  
15209       The value of this edit option shall be a string of <blank> character-delimited path names of files  
15210       used by the **tag** command. The default value is unspecified.
- 15211       **term**  
15212       [Default from the environment variable *TERM*]  
15213       The value of this edit option shall be a string. The default shall be taken from the *TERM* variable  
15214       in the environment. If the *TERM* environment variable is empty or null, the default is  
15215       unspecified. The editor shall use the value of this edit option to determine the type of the display  
15216       device.  
15217       The results are unspecified if the user changes the value of the term edit option after editor  
15218       initialization.
- 15219       **terse**  
15220       [Default *unset*]  
15221       If **terse** is set, error messages may be less verbose. However, except for this caveat, error  
15222       messages are unspecified. Furthermore, not all error messages need change for different settings  
15223       of this option.
- 15224       **warn**  
15225       [Default *set*]  
15226       If **warn** is set, and the contents of the edit buffer have been modified since they were last  
15227       completely written, the editor shall write a warning message before certain ! commands (see  
15228       **Escape** (on page 2599)).

- 15229       **window**
- 15230       [Default *see text*]
- 15231       A value used in open and visual mode, by the <control>-B and <control>-F commands, and, in  
15232       visual mode, to specify the number of lines displayed when the screen is repainted.
- 15233       If the **-w** command-line option is not specified, the default value shall be set to the value of the  
15234       *LINES* environment variable. If the *LINES* environment variable is empty or null, the default  
15235       shall be the number of lines in the display minus 1.
- 15236       Setting the **window** edit option to zero or to a value greater than the number of lines in the  
15237       display minus 1 (either explicitly or based on the **-w** option or the *LINES* environment variable)  
15238       shall cause the **window** edit option to be set to the number of lines in the display minus 1.
- 15239       The baud rate of the terminal line may change the default in an implementation-defined manner.
- 15240       **wrapmargin, wm**
- 15241       [Default 0]
- 15242       If the value of this edit option is zero, it shall have no effect.
- 15243       If not in the POSIX locale, the effect of this edit option is implementation-defined.
- 15244       Otherwise, it shall specify a number of columns from the ending margin of the terminal.
- 15245       During open and visual text input modes, for each character for which any part of the character  
15246       is displayed in a column that is less than **wrapmargin** columns from the ending margin of the  
15247       screen, the editor shall behave as follows:
- 15248       1. If the character triggering this event is a <blank> character, it, and all immediately  
15249       preceding <blank> characters on the current line entered during the execution of the  
15250       current text input command, shall be discarded, and the editor shall behave as if the user  
15251       had entered a single <newline> character instead. In addition, if the next user-entered  
15252       character is a <space> character, it shall be discarded as well.
  - 15253       2. Otherwise, if there are one or more <blank> characters on the current line immediately  
15254       preceding the last group of inserted non-<blank> characters which was entered during the  
15255       execution of the current text input command, the <blank> characters shall be replaced as if  
15256       the user had entered a single <newline> character instead.
- 15257       If the **autoindent** edit option is set, and the events described in 1. or 2. are performed, any  
15258       <blank> characters at or after the cursor in the current line shall be discarded.
- 15259       The ending margin shall be determined by the system or overridden by the user, as described for  
15260       *COLUMNS* in in the ENVIRONMENT VARIABLES section and the Base Definitions volume of  
15261       IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.
- 15262       **wrapscan, ws**
- 15263       [Default *set*]
- 15264       If **wrapscan** is set, searches (the *ex* / or ? addresses, or open and visual mode /, ?, N, and **n**  
15265       commands) shall wrap around the beginning or end of the edit buffer; when unset, searches  
15266       shall stop at the beginning or end of the edit buffer.

15267           **writeany, wa**  
 15268           [Default *unset*]  
 15269           If **writeany** is set, some of the checks performed when executing the **ex write** commands shall be  
 15270           inhibited, as described in editor option **autowrite**.

#### 15271 **EXIT STATUS**

15272           The following exit values shall be returned:

15273           0   Successful completion.

15274           >0  An error occurred.

#### 15275 **CONSEQUENCES OF ERRORS**

15276           When any error is encountered and the standard input is not a terminal device file, **ex** shall not  
 15277           write the file or return to command or text input mode, and shall terminate with a non-zero exit  
 15278           status.

15279           Otherwise, when an unrecoverable error is encountered, it shall be equivalent to a SIGHUP  
 15280           asynchronous event.

15281           Otherwise, when an error is encountered, the editor shall behave as specified in **Command Line**  
 15282           **Parsing in ex** (on page 2573).

#### 15283 **APPLICATION USAGE**

15284           If a SIGSEGV signal is received while **ex** is saving a file, the file might not be successfully saved.

15285           The **next** command can accept more than one file, so usage such as:

15286           next `ls [abc]\*`

15287           is valid; it would not be valid for the **edit** or **read** commands, for example, because they expect  
 15288           only one file and unspecified results occur.

#### 15289 **EXAMPLES**

15290           None.

#### 15291 **RATIONALE**

15292           The **ex/vi** specification is based on the historical practice found in the 4 BSD and System V  
 15293           implementations of **ex** and **vi**. A freely redistributable implementation of **ex/vi**, which is  
 15294           tracking IEEE Std. 1003.1-200x fairly closely, and demonstrates the intended changes between  
 15295           historical implementations and IEEE Std. 1003.1-200x, may be obtained by anonymous FTP  
 15296           from:

15297           ftp://ftp.rdg.opengroup/pub/mirrors/nvi

15298           A *restricted editor* (both the historical *red* utility and modifications to **ex**) were considered and  
 15299           rejected for inclusion. Neither option provided the level of security that users might expect.

15300           It is recognized that **ex** visual mode and related features would be difficult, if not impossible, to  
 15301           implement satisfactorily on a block-mode terminal, or a terminal without any form of cursor  
 15302           addressing; thus, it is not a mandatory requirement that such features should work on all  
 15303           terminals. It is the intention, however, that an **ex** implementation should provide the full set of  
 15304           capabilities on all terminals capable of supporting them.



15305 **Options**

15306 The `-c` replacement for `+command` was inspired by the `-e` option of `sed`. Historically, all such  
 15307 commands (see `edit` and `next` as well) were executed from the last line of the edit buffer. This  
 15308 meant, for example, that `+/pattern` would fail unless the `wrapsan` option was set.  
 15309 IEEE Std. 1003.1-200x requires conformance to historical practice. Historically, some  
 15310 implementations restricted the `ex` commands that could be listed as part of the command line  
 15311 arguments. For consistency, IEEE Std. 1003.1-200x does not permit these restrictions.

15312 In historical implementations of the editor, the `-R` option (and the `readonly` edit option) only  
 15313 prevented overwriting of files; appending to files was still permitted, mapping loosely into the  
 15314 `cs`h `noclobber` variable. Some implementations, however, have not followed this semantic, and  
 15315 `readonly` does not permit appending either. IEEE Std. 1003.1-200x follows the latter practice,  
 15316 believing that it is a more obvious and intuitive meaning of `readonly`.

15317 The `-s` option suppresses all interactive user feedback and is useful for editing scripts in batch  
 15318 jobs. The list of specific effects is historical practice. The terminal type “incapable of supporting  
 15319 open and visual modes” has historically been named “dumb”.

15320 The `-t` option was required because the `ctags` utility appears in IEEE Std. 1003.1-200x and the  
 15321 option is available in all historical implementations of `ex`.

15322 Historically, the `ex` and `vi` utilities accepted a `-x` option, which did encryption based on the  
 15323 algorithm found in the historical `crypt` utility. The `-x` option for encryption, and the associated  
 15324 `crypt` utility, were omitted because the algorithm used was not specifiable and the export control  
 15325 laws of some nations make it difficult to export cryptographic technology. In addition, it did not  
 15326 historically provide the level of security that users might expect.

15327 **Standard Input**

15328 An end-of-file condition is not equivalent to an end-of-file character. A common end-of-file  
 15329 character, `<control>-D`, is historically an `ex` command.

15330 There was no maximum line length in historical implementations of `ex`. Specifically, as it was  
 15331 parsed in chunks, the addresses had a different maximum length than the file names. Further,  
 15332 the maximum line buffer size was declared as `{BUFSIZ}`, which was different lengths on  
 15333 different systems. This version selected the value of `{LINE_MAX}` to impose a reasonable  
 15334 restriction on portable usage of `ex` and to aid test suite writers in their development of realistic  
 15335 tests that exercise this limit.

15336 **Input Files**

15337 It was an explicit decision by the standard developers that a `<newline>` character be added to  
 15338 any file lacking one. It was believed that this feature of `ex` and `vi` was relied on by users in order  
 15339 to make text files lacking a trailing `<newline>` more portable. It is recognized that this will  
 15340 require a user-specified option or extension for implementations that permit `ex` and `vi` to edit  
 15341 files of type other than text if such files are not otherwise identified by the system. It was agreed  
 15342 that the ability to edit files of arbitrary type can be useful, but it was not considered necessary to  
 15343 mandate that an `ex` or `vi` implementation be required to handle files other than text files.

15344 The paragraph in the INPUT FILES section, “By default, ...”, is intended to close a long-standing  
 15345 security problem in `ex` and `vi`, that of the “`modeline`” or “`modelines`” edit option. This feature  
 15346 allows any line in the first or last five lines of the file containing the strings `"ex:"` or `"vi:"`  
 15347 (and, apparently, `"ei:"` or `"vx:"`) to be a line containing editor commands, and `ex` interprets all  
 15348 the text up to the next `':'` or `<newline>` as a command. Consider the consequences, for  
 15349 example, of an unsuspecting user using `ex` or `vi` as the editor when replying to a mail message in  
 15350 which a line such as:

15351 `ex:! rm -rf :`

15352 appeared in the signature lines. The standard developers believed strongly that an editor should  
15353 not by default interpret any lines of a file. Vendors are strongly urged to delete this feature from  
15354 their implementations of *ex* and *vi*.

### 15355 **Asynchronous Events**

15356 The intention of the phrase “complete write” is that the entire edit buffer be written to stable  
15357 storage. The note regarding temporary files is intended for implementations that use temporary  
15358 files to back edit buffers unnamed by the user.

15359 Historically, SIGQUIT was ignored by *ex*, but was the equivalent of the **Q** command in visual  
15360 mode; that is, it exited visual mode and entered *ex* mode. IEEE Std. 1003.1-200x permits, but does  
15361 not require, this behavior. Historically, SIGINT was often used by *vi* users to terminate text  
15362 input mode (<control>-C is often easier to enter than <ESC>). Some implementations of *vi*  
15363 alerted the terminal on this event, and some did not. IEEE Std. 1003.1-200x requires that SIGINT  
15364 behave identically to <ESC>, and that the terminal not be alerted.

15365 Historically, suspending the *ex* editor during text input mode was similar to SIGINT, as  
15366 completed lines were retained, but any partial line discarded, and the editor returned to  
15367 command mode. IEEE Std. 1003.1-200x is silent on this issue; implementations are encouraged to  
15368 follow historical practice, where possible.

15369 Historically, the *vi* editor did not treat SIGTSTP as an asynchronous event, and it was therefore  
15370 impossible to suspend the editor in visual text input mode. There are two major reasons for this.  
15371 The first is that SIGTSTP is a broadcast signal on UNIX systems, and the chain of events where  
15372 the shell *execs* an application that then *execs vi* usually caused confusion for the terminal state if  
15373 SIGTSTP was delivered to the process group in the default manner. The second was that most  
15374 implementations of the UNIX *curses* package are not reentrant, and the receipt of SIGTSTP at the  
15375 wrong time will cause them to crash. IEEE Std. 1003.1-200x is silent on this issue;  
15376 implementations are encouraged to treat suspension as an asynchronous event if possible.

15377 Historically, modifications to the edit buffer made before SIGINT interrupted an operation were  
15378 retained; that is, anywhere from zero to all of the lines to be modified might have been modified  
15379 by the time the SIGINT arrived. These changes were not discarded by the arrival of SIGINT.  
15380 IEEE Std. 1003.1-200x permits this behavior, noting that the *undo* command is required to be able  
15381 to undo these partially completed commands.

15382 The action taken for signals other than SIGINT, SIGCONT, SIGHUP, and SIGTERM is  
15383 unspecified because some implementations attempt to save the edit buffer in a useful state when  
15384 other signals are received.

### 15385 **Standard Error**

15386 For *ex/vi*, diagnostic messages are those messages reported as a result of a failed attempt to  
15387 invoke *ex* or *vi*, such as invalid options or insufficient resources, or an abnormal termination  
15388 condition. Diagnostic messages should not be confused with the error messages generated by  
15389 inappropriate or illegal user commands.

15390 **Initialization in ex and vi**

15391 If an *ex* command (other than **cd**, **chdir**, or **source**) has a file name argument, one or both of the  
15392 alternate and current path names will be set. Informally, they are set as follows:

- 15393 1. If the *ex* command is one that replaces the contents of the edit buffer, and it succeeds, the  
15394 current path name will be set to the file name argument (the first file name argument in the  
15395 case of the **next** command) and the alternate path name will be set to the previous current  
15396 path name, if there was one.
- 15397 2. In the case of the file read/write forms of the **read** and **write** commands, if there is no  
15398 current path name, the current path name will be set to the file name argument.
- 15399 3. Otherwise, the alternate path name will be set to the file name argument.

15400 For example, **:edit foo** and **:recover foo**, when successful, set the current path name, and, if there  
15401 was a previous current path name, the alternate path name. The commands **:write**, **!command**,  
15402 and **:edit** set neither the current or alternate path names. If the **:edit foo** command were to fail  
15403 for some reason, the alternate path name would be set. The **read** and **write** commands set the  
15404 alternate path name to their *file* argument, unless the current path name is not set, in which case  
15405 they set the current path name to their *file* arguments. The alternate path name was not  
15406 historically set by the **:source** command. IEEE Std. 1003.1-200x requires conformance to  
15407 historical practice. Implementations adding commands that take file names as arguments are  
15408 encouraged to set the alternate path name as described here.

15409 Historically, *ex* and *vi* read the **.exrc** file in the *\$HOME* directory twice, if the editor was executed  
15410 in the *\$HOME* directory. IEEE Std. 1003.1-200x prohibits this behavior.

15411 Historically, the 4 BSD *ex* and *vi* read the *\$HOME* and local **.exrc** files if they were owned by the  
15412 real ID of the user, or the **sourceany** option was set, regardless of other considerations. This was  
15413 a security problem because it is possible to put normal UNIX system commands inside a **.exrc**  
15414 file. IEEE Std. 1003.1-200x does not specify the **sourceany** option, and historical implementations  
15415 are encouraged to delete it.

15416 The **.exrc** files must be owned by the real ID of the user, and not writeable by anyone other than  
15417 the owner. The appropriate privileges exception is intended to permit users to acquire special  
15418 privileges, but continue to use the **.exrc** files in their home directories.

15419 System V Release 3.2 and later *vi* implementations added the option **[no]exrc**. The behavior is  
15420 that local **.exrc** files are read-only if the **exrc** option is set. The default for the **exrc** option was off,  
15421 so by default, local **.exrc** files were not read. The problem this was intended to solve was that  
15422 System V permitted users to give away files, so there is no possible ownership or writeability  
15423 test to ensure that the file is safe. This is still a security problem on systems where users can give  
15424 away files, but there is nothing additional that IEEE Std. 1003.1-200x can do. The  
15425 implementation-defined exception is intended to permit groups to have local **.exrc** files that are  
15426 shared by users, by creating pseudo-users to own the shared files.

15427 IEEE Std. 1003.1-200x does not mention system-wide *ex* and *vi* start-up files. While they exist in  
15428 several implementations of *ex* and *vi*, they are not present in any implementations considered  
15429 historical practice by IEEE Std. 1003.1-200x. Implementations that have such files should use  
15430 them only if they are owned by the real user ID or an appropriate user (for example, root on  
15431 UNIX systems) and if they are not writeable by any user other than their owner. System-wide  
15432 start-up files should be read before the *EXINIT* variable, *\$HOME/.exrc* or local **.exrc** files are  
15433 evaluated.

15434 Historically, any *ex* command could be entered in the *EXINIT* variable or the **.exrc** file, although  
15435 ones requiring that the edit buffer already contain lines of text generally caused historical  
15436 implementations of the editor to drop core. IEEE Std. 1003.1-200x requires that any *ex* command

15437 be permitted in the *EXINIT* variable and *.exrc* files, for simplicity of specification and  
15438 consistency, although many of them will obviously fail under many circumstances.

15439 The initialization of the contents of the edit buffer uses the phrase “the effect shall be” with  
15440 regard to various *ex* commands. The intent of this phrase is that edit buffer contents loaded  
15441 during the initialization phase not be lost; that is, loading the edit buffer should fail if the *.exrc*  
15442 file read in the contents of a file and did not subsequently write the edit buffer. An additional  
15443 intent of this phrase is to specify that the initial current line and column is set as specified for the  
15444 individual *ex* commands.

15445 Historically, the *-t* option behaved as if the tag search were a *+command*; that is, it was executed  
15446 from the last line of the file specified by the tag. This resulted in the search failing if the pattern  
15447 was a forward search pattern and the *wrapsan* edit option was not set. IEEE Std. 1003.1-200x  
15448 does not permit this behavior, requiring that the search for the tag pattern be performed on the  
15449 entire file, and, if not found, that the current line be set to a more reasonable location in the file.

15450 Historically, the empty edit buffer presented for editing when a file was not specified by the user  
15451 was unnamed. This is permitted by IEEE Std. 1003.1-200x; however, implementations are  
15452 encouraged to provide users a temporary file name for this buffer because it permits them the  
15453 use of *ex* commands that use the current path name during temporary edit sessions.

15454 Historically, the file specified using the *-t* option was not part of the current argument list. This  
15455 practice is permitted by IEEE Std. 1003.1-200x; however, implementations are encouraged to  
15456 include its name in the current argument list for consistency.

15457 Historically, the *-c* command was generally not executed until a file that already exists was  
15458 edited. IEEE Std. 1003.1-200x requires conformance to this historical practice. Commands that  
15459 could cause the *-c* command to be executed include the *ex* commands **edit**, **next**, **recover**,  
15460 **rewind**, and **tag**, and the *vi* commands *<control>-^* and *<control>-]*. Historically, reading a file  
15461 into an edit buffer did not cause the *-c* command to be executed (even though it might set the  
15462 current path name) with the exception that it did cause the *-c* command to be executed if: the  
15463 editor was in *ex* mode, the edit buffer had no current path name, the edit buffer was empty, and  
15464 no read commands had yet been attempted. For consistency and simplicity of specification,  
15465 IEEE Std. 1003.1-200x does not permit this behavior.

15466 Historically, the *-r* option was the same as a normal edit session if there was no recovery  
15467 information available for the file. This allowed users to enter:

```
15468 vi -r *.c
```

15469 and recover whatever files were recoverable. In some implementations, recovery was attempted  
15470 only on the first file named, and the file was not entered into the argument list; in others,  
15471 recovery was attempted for each file named. In addition, some historical implementations  
15472 ignored *-r* if *-t* was specified or did not support command line *file* arguments with the *-t* option.  
15473 For consistency and simplicity of specification, IEEE Std. 1003.1-200x disallows these special  
15474 cases, and requires that recovery be attempted the first time each file is edited.

15475 Historically, *vi* initialized the ‘ and ’ marks, but *ex* did not. This meant that if the first command  
15476 in *ex* mode was **visual** or if an *ex* command was executed first (for example, *vi +10 file*), *vi*  
15477 was entered without the marks being initialized. Because the standard developers believed the marks  
15478 to be generally useful, and for consistency and simplicity of specification, IEEE Std. 1003.1-200x  
15479 requires that they always be initialized if in open or visual mode, or if in *ex* mode and the edit  
15480 buffer is not empty. Not initializing it in *ex* mode if the edit buffer is empty is historical practice;  
15481 however, it has always been possible to set (and use) marks in empty edit buffers in open and  
15482 visual mode edit sessions.

15483 **Addressing**

15484 Historically, *ex* and *vi* accepted the additional addressing forms '`\/'` and '`\?'`. They were  
 15485 equivalent to "`//`" and "`??`", respectively. They are not required by IEEE Std. 1003.1-200x,  
 15486 mostly because nobody can remember whether they ever did anything different historically.

15487 Historically, *ex* and *vi* permitted an address of zero for several commands, and permitted the %  
 15488 address in empty files for others. For consistency, IEEE Std. 1003.1-200x requires support for the  
 15489 former in the few commands where it makes sense, and disallows it otherwise. In addition,  
 15490 because IEEE Std. 1003.1-200x requires that % be logically equivalent to "`1,$`", it is also  
 15491 supported where it makes sense and disallowed otherwise.

15492 Historically, the % address could not be followed by further addresses. For consistency and  
 15493 simplicity of specification, IEEE Std. 1003.1-200x requires that additional addresses be  
 15494 supported.

15495 All of the following are valid *addresses*:

15496 `+++` Three lines after the current line.

15497 `/re/-` One line before the next occurrence of *re*.

15498 `-2` Two lines before the current line.

15499 `3 —— 2` Line one (note intermediate negative address).

15500 `1 2 3` Line six.

15501 Any number of addresses can be provided to commands taking addresses; for example,  
 15502 "`1,2,3,4,5p`" prints lines 4 and 5, because two is the greatest valid number of addresses  
 15503 accepted by the **print** command. This, in combination with the semicolon delimiter, permits  
 15504 users to create commands based on ordered patterns in the file. For example, the command  
 15505 **3/foo/+2print** will display the first line after line 3 that contains the pattern *foo*, plus the next  
 15506 two lines. Note that the address **3**; must be evaluated before being discarded because the search  
 15507 origin for the **/foo/** command depends on this.

15508 Historically, values could be added to addresses by including them after one or more <blank>  
 15509 characters; for example, **3 - 5p** wrote the seventh line of the file, and **/foo/ 5** was the same as  
 15510 **/foo/+5**. However, only absolute values could be added; for example, **5 /foo/** was an error.  
 15511 IEEE Std. 1003.1-200x requires conformance to historical practice. Address offsets are separately  
 15512 specified from addresses because they could historically be provided to visual mode search  
 15513 commands.

15514 Historically, any missing addresses defaulted to the current line. This was true for leading and  
 15515 trailing comma-delimited addresses, and for trailing semicolon-delimited addresses. For  
 15516 consistency, IEEE Std. 1003.1-200x requires it for leading semicolon addresses as well.

15517 Historically, *ex* and *vi* accepted the '`^`' character as both an address and as a flag offset for  
 15518 commands. In both cases it was identical to the '`-`' character. IEEE Std. 1003.1-200x does not  
 15519 require or prohibit this behavior.

15520 Historically, the enhancements to basic regular expressions could be used in addressing; for  
 15521 example, '`~`', '`\<`', and '`\>`'. IEEE Std. 1003.1-200x requires conformance to historical  
 15522 practice; that is, that regular expression usage be consistent, and that regular expression  
 15523 enhancements be supported wherever regular expressions are used.

15524 **Command Line Parsing in ex**

15525 Historical **ex** command parsing was even more complex than that described here.  
 15526 IEEE Std. 1003.1-200x requires the subset of the command parsing that the standard developers  
 15527 believed was documented and that users could reasonably be expected to use in a portable  
 15528 fashion, and that was historically consistent between implementations. (The discarded  
 15529 functionality is obscure, at best.) Historical implementations will require changes in order to  
 15530 comply with IEEE Std. 1003.1-200x; however, users are not expected to notice any of these  
 15531 changes. Most of the complexity in **ex** parsing is to handle three special termination cases:

- 15532 1. The **!**, **global**, **v**, and the filter versions of the **read** and **write** commands are delimited by  
 15533 <newline> characters (they can contain vertical-line characters that are usually shell pipes).
- 15534 2. The **ex**, **edit**, **next**, and **visual** in open and visual mode commands all take **ex** commands,  
 15535 optionally containing vertical-line characters, as their first arguments.
- 15536 3. The **s** command takes a regular expression as its first argument, and uses the delimiting  
 15537 characters to delimit the command.

15538 Historically, vertical-line characters in the *+command* argument of the **ex**, **edit**, **next**, **vi**, and  
 15539 **visual** commands, and in the *pattern* and *replacement* parts of the **s** command, did not delimit the  
 15540 command, and in the filter cases for **read** and **write**, and the **!**, **global**, and **v** commands, they did  
 15541 not delimit the command at all. For example, the following commands are all valid:

```
15542 :edit +25 | s/abc/ABC/ file.c
15543 :s/ | /PIPE/
15544 :read !spell % | columnate
15545 :global/pattern/p | l
15546 :s/a/b/ | s/c/d | set
```

15547 Historically, empty or <blank> filled lines in **.exrc** files and **sourced** files (as well as **EXINIT** |  
 15548 variables and **ex** command scripts) were treated as default commands; that is, **print** commands.  
 15549 IEEE Std. 1003.1-200x specifically requires that they be ignored when encountered in **.exrc** and  
 15550 **sourced** files to eliminate a common source of new user error.

15551 Historically, **ex** commands with multiple adjacent (or <blank>-separated) vertical lines were  
 15552 handled oddly when executed from **ex** mode. For example, the command | | | <carriage-return>,  
 15553 when the cursor was on line 1, displayed lines 2, 3, and 5 of the file. In addition, the command |  
 15554 would only display the line after the next line, instead of the next two lines. The former worked  
 15555 more logically when executed from **vi** mode, and displayed lines 2, 3, and 4.  
 15556 IEEE Std. 1003.1-200x requires the **vi** behavior; that is, a single default command and line  
 15557 number increment for each command separator, and trailing <newline> characters after  
 15558 vertical-line separators are discarded.

15559 Historically, **ex** permitted a single extra colon as a leading command character; for example,  
 15560 **:g/pattern/p** was a valid command. IEEE Std. 1003.1-200x generalizes this to require that any  
 15561 number of leading colon characters be stripped.

15562 Historically, any prefix of the **delete** command could be followed without intervening <blank>  
 15563 characters by a flag character because in the command **d p**, *p* is interpreted as the buffer *p*.  
 15564 IEEE Std. 1003.1-200x requires conformance to historical practice.

15565 Historically, the **k** command could be followed by the mark name without intervening <blank>  
 15566 characters. IEEE Std. 1003.1-200x requires conformance to historical practice.

15567 Historically, the **s** command could be immediately followed by flag and option characters; for  
 15568 example, **s/e/E/|s|sgc3p** was a valid command. However, flag characters could not stand alone;  
 15569 for example, the commands **sp** and **s l** would fail, while the command **sgp** and **s gl** would

15570 succeed. (Obviously, the '#' flag character was used as a delimiter character if it followed the  
 15571 command.) Another issue was that option characters had to precede flag characters even when  
 15572 the command was fully specified; for example, the command **s/e/E/pg** would fail, while the  
 15573 command **s/e/E/gp** would succeed. IEEE Std. 1003.1-200x requires conformance to historical  
 15574 practice.

15575 Historically, the first command name that had a prefix matching the input from the user was the  
 15576 executed command; for example, **ve**, **ver**, and **vers** all executed the **version** command.  
 15577 Commands were in a specific order, however, so that **a** matched **append**, not **abbreviate**.  
 15578 IEEE Std. 1003.1-200x requires conformance to historical practice. The restriction on command  
 15579 search order for implementations with extensions is to avoid the addition of commands such  
 15580 that the historical prefixes would fail to work portably.

15581 Historical implementations of *ex* and *vi* did not correctly handle multiple *ex* commands,  
 15582 separated by vertical-line characters, that entered or exited visual mode or the editor. Because  
 15583 implementations of *vi* exist that do not exhibit this failure mode, IEEE Std. 1003.1-200x does not  
 15584 permit it.

15585 The requirement that alphabetic command names consist of all following alphabetic characters  
 15586 up to the next non-alphabetic character means that alphabetic command names must be  
 15587 separated from their arguments by one or more non-alphabetic characters, normally a <blank>  
 15588 or '!' character, except as specified for the exceptions, the **delete**, **k**, and **s** commands.

15589 Historically, the repeated execution of the *ex* default **print** commands (<control>-D, *eof*,  
 15590 <newline>, <carriage-return>) erased any prompting character and displayed the next lines  
 15591 without scrolling the terminal; that is, immediately below any previously displayed lines. This  
 15592 provided a cleaner presentation of the lines in the file for the user. IEEE Std. 1003.1-200x does not  
 15593 require this behavior because it may be impossible in some situations; however,  
 15594 implementations are strongly encouraged to provide this semantic if possible.

15595 Historically, it was possible to change files in the middle of a command, and have the rest of the  
 15596 command executed in the new file; for example:

```
15597 :edit +25 file.c | s/abc/ABC/ | 1
```

15598 was a valid command, and the substitution was attempted in the newly edited file.  
 15599 IEEE Std. 1003.1-200x requires conformance to historical practice. The following commands are  
 15600 examples that exercise the *ex* parser:

```
15601 echo 'foo | bar' > file1; echo 'foo/bar' > file2;
```

```
15602 vi
```

```
15603 :edit +1 | s/|/PIPE/ | w file1 | e file2 | 1 | s/\/SLASH/ | wq
```

15604 Historically, there was no protection in editor implementations to avoid *ex* **global**, **v**, **@**, or **\***  
 15605 commands changing edit buffers during execution of their associated commands. Because this  
 15606 would almost invariably result in catastrophic failure of the editor, and implementations exist  
 15607 that do exhibit these problems, IEEE Std. 1003.1-200x requires that changing the edit buffer  
 15608 during a **global** or **v** command, or during a **@** or **\*** command for which there will be more than a  
 15609 single execution, be an error. Implementations supporting multiple edit buffers simultaneously  
 15610 are strongly encouraged to apply the same semantics to switching between buffers as well.

15611 The *ex* command quoting required by IEEE Std. 1003.1-200x is a superset of the quoting in  
 15612 historical implementations of the editor. For example, it was not historically possible to escape a  
 15613 <blank> character in a file name; for example, **:edit foo\\\ bar** would report that too many file  
 15614 names had been entered for the edit command, and there was no method of escaping a <blank>  
 15615 in the first argument of an **edit**, **ex**, **next**, or **visual** command at all. IEEE Std. 1003.1-200x extends  
 15616 historical practice, requiring that quoting behavior be made consistent across all *ex* commands,

15617 except for the **map**, **unmap**, **abbreviate**, and **unabbreviate** commands, which historically used  
 15618 <control>-V instead of backslashes for quoting. For those four commands, IEEE Std. 1003.1-200x  
 15619 requires conformance to historical practice.

15620 Backslash quoting in *ex* is non-intuitive. Backslash escapes are ignored unless they escape a  
 15621 special character; for example, when performing *file* argument expansion, the string "\\%" is  
 15622 equivalent to '\%', not "\<current path name>". This can be confusing for users because  
 15623 backslash is usually one of the characters that causes shell expansion to be performed, and  
 15624 therefore shell quoting rules must be taken into consideration. Generally, quoting characters are  
 15625 only considered if they escape a special character, and a quoting character must be provided for  
 15626 each layer of parsing for which the character is special. As another example, only a single  
 15627 backslash is necessary for the '\1' sequence in substitute replacement patterns, because the  
 15628 character '1' is not special to any parsing layer above it.

15629 <control>-V quoting in *ex* is slightly different from backslash quoting. In the four commands  
 15630 where <control>-V quoting applies (**abbreviate**, **unabbreviate**, **map**, and **unmap**), any character  
 15631 may be escaped by a <control>-V whether it would have a special meaning or not.  
 15632 IEEE Std. 1003.1-200x requires conformance to historical practice.

15633 Historical implementations of the editor did not require delimiters within character classes to be  
 15634 escaped; for example, the command :s/[/]// on the string "xxx/yyy" would delete the '/' from  
 15635 the string. IEEE Std. 1003.1-200x disallows this historical practice for consistency and because it  
 15636 places a large burden on implementations by requiring that knowledge of regular expressions be  
 15637 built into the editor parser.

15638 Historically, quoting <newline> characters in *ex* commands was handled inconsistently. In most  
 15639 cases, the <newline> always terminated the command, regardless of any preceding escape  
 15640 character, because backslash characters did not escape <newline> characters for most *ex*  
 15641 commands. However, some *ex* commands (for example, **s**, **map**, and **abbreviation**) permitted  
 15642 <newline> characters to be escaped (although in the case of **map** and **abbreviation**, <control>-V  
 15643 characters escaped them instead of backslashes). This was true in not only the command line,  
 15644 but also **.exrc** and **sourced** files. For example, the command:

```
15645 map = foo<control-V><newline>bar
```

15646 would succeed, although it was sometimes difficult to get the <control>-V and the inserted  
 15647 <newline> passed to the *ex* parser. For consistency and simplicity of specification,  
 15648 IEEE Std. 1003.1-200x requires that it be possible to escape <newline> characters in *ex* commands  
 15649 at all times, using backslashes for most *ex* commands, and using <control>-V characters for the  
 15650 **map** and **abbreviation** commands. For example, the command **print<newline>list** is required to  
 15651 be parsed as the single command **print<newline>list**. While this differs from historical practice,  
 15652 IEEE Std. 1003.1-200x developers believed it unlikely that any script or user depended on the  
 15653 historical behavior.

15654 Historically, an error in a command specified using the **-c** option did not cause the rest of the **-c**  
 15655 commands to be discarded. IEEE Std. 1003.1-200x disallows this for consistency with mapped  
 15656 keys, the **@**, **global**, **source**, and **v** commands, the *EXINIT* environment variable, and the **.exrc**  
 15657 files.



15658 **Input Editing in ex**

15659 One of the common uses of the historical *ex* editor is over slow network connections. Editors  
 15660 that run in canonical mode can require far less traffic to and from, and far less processing on, the  
 15661 host machine, as well as more easily supporting block-mode terminals. For these reasons,  
 15662 IEEE Std. 1003.1-200x requires that *ex* be implemented using canonical mode input processing,  
 15663 as was done historically.

15664 IEEE Std. 1003.1-200x does not require the historical 4 BSD input editing characters “word erase”  
 15665 or “literal next”. For this reason, it is unspecified how they are handled by *ex*, although they  
 15666 must have the required effect. Implementations that resolve them after the line has been ended  
 15667 using a <newline> or <control>-M character, and implementations that rely on the underlying  
 15668 system terminal support for this processing, are both conforming. Implementations are strongly  
 15669 urged to use the underlying system functionality, if at all possible, for compatibility with other  
 15670 system text input interfaces.

15671 Historically, when the *eof* character was used to decrement the **autoindent** level, the cursor  
 15672 moved to display the new end of the **autoindent** characters, but did not move the cursor to a  
 15673 new line, nor did it erase the <control>-D character from the line. IEEE Std. 1003.1-200x does not  
 15674 specify that the cursor remain on the same line or that the rest of the line is erased; however,  
 15675 implementations are strongly encouraged to provide the best possible user interface; that is, the  
 15676 cursor should remain on the same line, and any <control>-D character on the line should be  
 15677 erased.

15678 IEEE Std. 1003.1-200x does not require the historical 4 BSD input editing character “reprint”,  
 15679 traditionally <control>-R, which redisplayed the current input from the user. For this reason,  
 15680 and because the functionality cannot be implemented after the line has been terminated by the  
 15681 user, IEEE Std. 1003.1-200x makes no requirements about this functionality. Implementations are  
 15682 strongly urged to make this historical functionality available, if possible.

15683 Historically, <control>-Q did not perform a literal next function in *ex*, as it did in *vi*.  
 15684 IEEE Std. 1003.1-200x requires conformance to historical practice to avoid breaking historical *ex*  
 15685 scripts and **.exrc** files.

15686 **eof**

15687 Whether the *eof* character immediately modifies the **autoindent** characters in the prompt is left  
 15688 unspecified so that implementations can conform in the presence of systems that do not support  
 15689 this functionality. Implementations are encouraged to modify the line and redisplay it  
 15690 immediately, if possible.

15691 The specification of the handling of the *eof* character differs from historical practice only in that  
 15692 *eof* characters are not discarded if they follow normal characters in the text input. Historically,  
 15693 they were always discarded.

15694 **Command Descriptions in ex**

15695 Historically, several commands (for example, **global**, **v**, **visual**, **s**, **write**, **wq**, **yank**, **!**, **<**, **>**, **&**, and  
 15696 **->**) were executable in empty files (that is, the default address(es) were 0), or permitted explicit  
 15697 addresses of 0 (for example, 0 was a valid address, or 0,0 was a valid range). Addresses of 0, or  
 15698 command execution in an empty file, make sense only for commands that add new text to the  
 15699 edit buffer or write commands (because users may wish to write empty files).  
 15700 IEEE Std. 1003.1-200x requires this behavior for such commands and disallows it otherwise, for  
 15701 consistency and simplicity of specification.

15702 A count to an *ex* command has been historically corrected to be no greater than the last line in a  
 15703 file; for example, in a five-line file, the command **1,6print** would fail, but the command **1print300**

- 15704 would succeed. IEEE Std. 1003.1-200x requires conformance to historical practice.
- 15705 Historically, the use of flags in *ex* commands could be obscure. General historical practice was as  
 15706 described by IEEE Std. 1003.1-200x, but there were some special cases. For example, the **list**,  
 15707 **number**, and **print** commands ignored trailing address offsets; for example, **3p** +++# would  
 15708 display line 3, and 3 would be the current line after the execution of the command. The **open** and  
 15709 **visual** commands ignored both the trailing offsets and the trailing flags. Also, flags specified to  
 15710 the **open** and **visual** commands interacted badly with the **list** edit option, and setting and then  
 15711 unsetting it during the open/visual session would cause *vi* to stop displaying lines in the  
 15712 specified format. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not  
 15713 permit any of these exceptions to the general rule.
- 15714 IEEE Std. 1003.1-200x uses the word *copy* in several places when discussing buffers. This is not  
 15715 intended to imply implementation.
- 15716 Historically, *ex* users could not specify numeric buffers because of the ambiguity this would  
 15717 cause; for example, in the command **3 delete 2**, it is unclear whether 2 is a buffer name or a  
 15718 *count*. IEEE Std. 1003.1-200x requires conformance to historical practice by default, but does not  
 15719 preclude extensions.
- 15720 Historically, the contents of the unnamed buffer were frequently discarded after commands that  
 15721 did not explicitly affect it; for example, when using the **edit** command to switch files. For  
 15722 consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.
- 15723 The *ex* utility did not historically have access to the numeric buffers, and, furthermore, deleting  
 15724 lines in *ex* did not modify their contents. For example, if, after doing a delete in *vi*, the user  
 15725 switched to *ex*, did another delete, and then switched back to *vi*, the contents of the numeric  
 15726 buffers would not have changed. IEEE Std. 1003.1-200x requires conformance to historical  
 15727 practice. Numeric buffers are described in the *ex* utility in order to confine the description of  
 15728 buffers to a single location in IEEE Std. 1003.1-200x.
- 15729 The metacharacters that trigger shell expansion in *file* arguments match historical practice, as  
 15730 does the method for doing shell expansion. Implementations wishing to provide users with the  
 15731 flexibility to alter the set of metacharacters are encouraged to provide a **shellmeta** string edit  
 15732 option.
- 15733 Historically, *ex* commands executed from *vi* refreshed the screen when it did not strictly need to  
 15734 do so; for example, **!date > /dev/null** does not require a screen refresh because the output of the  
 15735 UNIX *date* command requires only a single line of the screen. IEEE Std. 1003.1-200x requires that  
 15736 the screen be refreshed if it has been overwritten, but makes no requirements as to how an  
 15737 implementation should make that determination. Implementations may prompt and refresh the  
 15738 screen regardless.
- 15739 **Abbreviate**
- 15740 Historical practice was that characters that were entered as part of an abbreviation replacement  
 15741 were subject to **map** expansions, the **showmatch** edit option, further abbreviation expansions,  
 15742 and so on; that is, they were logically pushed onto the terminal input queue, and were not a  
 15743 simple replacement. IEEE Std. 1003.1-200x requires conformance to historical practice.  
 15744 Historical practice was that whenever a non-word character (that had not been escaped by a  
 15745 <control>-V) was entered after a word character, *vi* would check for abbreviations. The check  
 15746 was based on the type of the character entered before the word character of the word/non-word  
 15747 pair that triggered the check. The word character of the word/non-word pair that triggered the  
 15748 check and all characters entered before the trigger pair that were of that type were included in  
 15749 the check, with the exception of <blank> characters, which always delimited the abbreviation.

15750 This means that, for the abbreviation to work, the *lhs* must end with a word character, there can  
 15751 be no transitions from word to non-word characters (or *vice versa*) other than between the last  
 15752 and next-to-last characters in the *lhs*, and there can be no <blank> characters in the *lhs*. In  
 15753 addition, because of the historical quoting rules, it was impossible to enter a literal <control>-V  
 15754 in the *lhs*. IEEE Std. 1003.1-200x requires conformance to historical practice. Historical  
 15755 implementations did not inform users when abbreviations that could never be used were  
 15756 entered; implementations are strongly encouraged to do so.

15757 For example, the following abbreviations will work:

```
15758 :ab (p REPLACE
15759 :ab p REPLACE
15760 :ab ((p REPLACE
```

15761 The following abbreviations will not work:

```
15762 :ab (REPLACE
15763 :ab (pp REPLACE
```

15764 Historical practice is that words on the *vi* colon command line were subject to abbreviation  
 15765 expansion, including the arguments to the **abbrev** (and more interestingly) the **unabbrev**  
 15766 command. Because there are implementations that do not do abbreviation expansion for the first  
 15767 argument to those commands, this is permitted, but not required, by IEEE Std. 1003.1-200x.  
 15768 However, the following sequence:

```
15769 :ab foo bar
15770 :ab foo baz
```

15771 resulted in the addition of an abbreviation of "baz" for the string "bar" in historical *ex/vi*, and  
 15772 the sequence:

```
15773 :ab foo1 bar
15774 :ab foo2 bar
15775 :unabbreviate foo2
```

15776 deleted the abbreviation "foo1", not "foo2". These behaviors are not permitted by  
 15777 IEEE Std. 1003.1-200x because they clearly violate the expectations of the user.

15778 It was historical practice that <control>-V, not backslash, characters be interpreted as escaping  
 15779 subsequent characters in the **abbreviate** command. IEEE Std. 1003.1-200x requires conformance  
 15780 to historical practice; however, it should be noted that an abbreviation containing a <blank> will  
 15781 never work.

## 15782 **Append**

15783 Historically, any text following a vertical-line command separator after an **append**, **change**, or  
 15784 **insert** command became part of the insert text. For example, in the command:

```
15785 :g/pattern/append|stuff1
```

15786 a line containing the text "stuff1" would be appended to each line matching pattern. It was  
 15787 also historically valid to enter:

```
15788 :append|stuff1
15789 stuff2
15790 .
```

15791 and the text on the *ex* command line would be appended along with the text inserted after it.  
 15792 There was an historical bug, however, that the user had to enter two terminating lines (the ' .'   
 15793 lines) to terminate text input mode in this case. IEEE Std. 1003.1-200x requires conformance to

15794 historical practice, but disallows the historical need for multiple terminating lines.

### 15795 **Change**

15796 See the RATIONALE for the **append** command. Historical practice for cursor positioning after  
15797 the change command when no text is input, is as described in IEEE Std. 1003.1-200x. However,  
15798 one System V implementation is known to have been modified such that the cursor is positioned  
15799 on the first address specified, and not on the line before the first address. IEEE Std. 1003.1-200x  
15800 disallows this modification for consistency.

15801 Historically, the **change** command did not support buffer arguments, although some  
15802 implementations allow the specification of an optional buffer. This behavior is neither required  
15803 nor disallowed by IEEE Std. 1003.1-200x.

### 15804 **Change Directory**

15805 A common extension in *ex* implementations is to use the elements of a **cdpath** edit option as  
15806 prefix directories for *path* arguments to **chdir** that are relative path names and that do not have  
15807 ' .' or " ." as their first component. Elements in the **cdpath** edit option are colon-separated.  
15808 The initial value of the **cdpath** edit option is the value of the shell *CDPATH* environment  
15809 variable. This feature was not included in IEEE Std. 1003.1-200x because it does not exist in any  
15810 of the implementations considered historical practice.

### 15811 **Copy**

15812 Historical implementations of *ex* permitted copies to lines inside of the specified range; for  
15813 example, **:2,5copy3** was a valid command. IEEE Std. 1003.1-200x requires conformance to  
15814 historical practice.

### 15815 **Delete**

15816 IEEE Std. 1003.1-200x requires support for the historical parsing of a **delete** command followed  
15817 by flags, without any intervening <blank> characters. For example:

15818 **1dp** Deletes the first line and prints the line that was second.

15819 **1delep** As for **1dp**.

15820 **1d** Deletes the first line, saving it in buffer *p*.

15821 **1d p11** (Pee-one-ell.) Deletes the first line, saving it in buffer *p*, and listing the line that was  
15822 second.

### 15823 **Edit**

15824 Historically, any *ex* command could be entered as a *+command* argument to the **edit** command,  
15825 although some (for example, **insert** and **append**) were known to confuse historical  
15826 implementations. For consistency and simplicity of specification, IEEE Std. 1003.1-200x requires  
15827 that any command be supported as an argument to the **edit** command.

15828 Historically, the command argument was executed with the current line set to the last line of the  
15829 file, regardless of whether the **edit** command was executed from visual mode or not.  
15830 IEEE Std. 1003.1-200x requires conformance to historical practice.

15831 Historically, the *+command* specified to the **edit** and **next** commands was delimited by the first  
15832 <blank> character, and there was no way to quote them. For consistency, IEEE Std. 1003.1-200x  
15833 requires that the usual *ex* backslash quoting be provided.

15834 Historically, specifying the *+command* argument to the edit command required a file name to be  
15835 specified as well; for example, **:edit +100** would always fail. For consistency and simplicity of  
15836 specification, IEEE Std. 1003.1-200x does not permit this usage to fail for that reason.

15837 Historically, only the cursor position of the last file edited was remembered by the editor.  
15838 IEEE Std. 1003.1-200x requires that this be supported; however, implementations are permitted  
15839 to remember and restore the cursor position for any file previously edited.

#### 15840 **File**

15841 Historical versions of the *ex* editor **file** command displayed a current line and number of lines in  
15842 the edit buffer of 0 when the file was empty, while the *vi* **<control>-G** command displayed a  
15843 current line and number of lines in the edit buffer of 1 in the same situation.  
15844 IEEE Std. 1003.1-200x does not permit this discrepancy, instead requiring that a message be  
15845 displayed indicating that the file is empty.

#### 15846 **Global**

15847 The two-pass operation of the **global** and **v** commands is not intended to imply implementation,  
15848 only the required result of the operation.

15849 The current line and column are set as specified for the individual *ex* commands. This  
15850 requirement is cumulative; that is, the current line and column must track across all the  
15851 commands executed by the **global** or **v** commands.

#### 15852 **Insert**

15853 See the RATIONALE for the **append** command.

15854 Historically, **insert** could not be used with an address of zero; that is, not when the edit buffer  
15855 was empty. IEEE Std. 1003.1-200x requires that this command behave consistently with the  
15856 **append** command.

#### 15857 **Join**

15858 The action of the **join** command in relation to the special characters is only defined for the  
15859 POSIX locale because the correct amount of white space after a period varies; in Japanese none is  
15860 required, in French only a single space, and so on.

#### 15861 **List**

15862 The historical output of the **list** command was potentially ambiguous. The standard developers  
15863 believed correcting this to be more important than adhering to historical practice, and  
15864 IEEE Std. 1003.1-200x requires unambiguous output.

#### 15865 **Map**

15866 Historically, command mode maps only applied to command names; for example, if the  
15867 character 'x' was mapped to 'y', the command **fx** searched for the 'x' character, not the 'y'  
15868 character. IEEE Std. 1003.1-200x requires this behavior. Historically, entering **<control>-V** as the  
15869 first character of a *vi* command was an error. Several implementations have extended the  
15870 semantics of *vi* such that **<control>-V** means that the subsequent command character is not  
15871 mapped. This is permitted, but not required, by IEEE Std. 1003.1-200x. Regardless, using  
15872 **<control>-V** to escape the second or later character in a sequence of characters that might match  
15873 a **map** command, or any character in text input mode, is historical practice, and stops the entered  
15874 keys from matching a map. IEEE Std. 1003.1-200x requires conformance to historical practice.

15875 Historical implementations permitted digits to be used as a **map** command *lhs*, but then ignored  
15876 the map. IEEE Std. 1003.1-200x requires that the mapped digits not be ignored.

15877 The historical implementation of the **map** command did not permit **map** commands that were  
15878 more than a single character in length if the first character was printable. This behavior is  
15879 permitted, but not required, by IEEE Std. 1003.1-200x.

15880 Historically, mapped characters were remapped unless the **remap** edit option was not set, or the  
15881 prefix of the mapped characters matched the mapping characters; for example, in the **map**:

```
15882 :map ab abcd
```

15883 the characters "ab" were used as is and were not remapped, but the characters "cd" were  
15884 mapped if appropriate. This can cause infinite loops in the *vi* mapping mechanisms.  
15885 IEEE Std. 1003.1-200x requires conformance to historical practice, and that such loops be  
15886 interruptible.

15887 Text input maps had the same problems with expanding the *lhs* for the **ex map!** and **unmap!**  
15888 command as did the **ex abbreviate** and **unabbreviate** commands. See the RATIONALE for the **ex**  
15889 **abbreviate** command. IEEE Std. 1003.1-200x requires similar modification of some historical  
15890 practice for the **map** and **unmap** commands, as described for the **abbreviate** and **unabbreviate**  
15891 commands.

15892 Historically, **maps** that were subsets of other **maps** behaved differently depending on the order  
15893 in which they were defined. For example:

```
15894 :map! ab short
15895 :map! abc long
```

15896 would always translate the characters "ab" to "short", regardless of how fast the characters  
15897 "abc" were entered. If the entry order was reversed:

```
15898 :map! abc long
15899 :map! ab short
```

15900 the characters "ab" would cause the editor to pause, waiting for the completing 'c' character,  
15901 and the characters might never be mapped to "short". For consistency and simplicity of  
15902 specification, IEEE Std. 1003.1-200x requires that the shortest match be used at all times.

15903 The length of time the editor spends waiting for the characters to complete the *lhs* is unspecified  
15904 because the timing capabilities of systems are often inexact and variable, and it may depend on  
15905 other factors such as the speed of the connection. The time should be long enough for the user to  
15906 be able to complete the sequence, but not long enough for the user to have to wait. Some  
15907 implementations of *vi* have added a **keytime** option, which permits users to set the number of  
15908 0,1 seconds the editor waits for the completing characters. Because mapped terminal function  
15909 and cursor keys tend to start with an <ESC> character, and <ESC> is the key ending *vi* text input  
15910 mode, **maps** starting with <ESC> characters are generally exempted from this timeout period,  
15911 or, at least timed out differently.

## 15912 **Mark**

15913 Historically, users were able to set the "previous context" marks explicitly. In addition, the **ex**  
15914 commands " and " and the *vi* commands ", ", and " all referred to the same mark. In addition,  
15915 the previous context marks were not set if the command, with which the address setting the  
15916 mark was associated, failed. IEEE Std. 1003.1-200x requires conformance to historical practice.  
15917 Historically, if marked lines were deleted, the mark was also deleted, but would reappear if the  
15918 change was undone. IEEE Std. 1003.1-200x requires conformance to historical practice.

15919 The description of the special events that set the ' and ' marks matches historical practice. For  
15920 example, historically the command `/a/,b/` did not set the ' and ' marks, but the command  
15921 `/a/,b/delete` did.

### 15922 **Next**

15923 Historically, any `ex` command could be entered as a `+command` argument to the `next` command,  
15924 although some (for example, `insert` and `append`) were known to confuse historical  
15925 implementations. IEEE Std. 1003.1-200x requires that any command be permitted and that it  
15926 behave as specified. The `next` command can accept more than one file, so usage such as:

```
15927 next `ls [abc] `
```

15928 is valid; it need not be valid for the `edit` or `read` commands, for example, because they expect  
15929 only one file name.

15930 Historically, the `next` command behaved differently from the `:rewind` command in that it  
15931 ignored the force flag if the `autowrite` flag was set. For consistency, IEEE Std. 1003.1-200x does  
15932 not permit this behavior.

15933 Historically, the `next` command positioned the cursor as if the file had never been edited before,  
15934 regardless. IEEE Std. 1003.1-200x does not permit this behavior, for consistency with the `edit`  
15935 command.

15936 Implementations wanting to provide a counterpart to the `next` command that edited the  
15937 previous file have used the command `prev[ious]`, which takes no `file` argument.  
15938 IEEE Std. 1003.1-200x does not require this command.

### 15939 **Open**

15940 Historically, the `open` command would fail if the `open` edit option was not set.  
15941 IEEE Std. 1003.1-200x does not mention the `open` edit option and does not require this behavior.  
15942 Some historical implementations do not permit entering open mode from open or visual mode,  
15943 only from `ex` mode. For consistency, IEEE Std. 1003.1-200x does not permit this behavior.

15944 Historically, entering open mode from the command line (that is, `vi +open`) resulted in  
15945 anomalous behaviors; for example, the `ex` file and `set` commands, and the `vi` command  
15946 `<control>-G` did not work. For consistency, IEEE Std. 1003.1-200x does not permit this behavior.

15947 Historically, the `open` command only permitted ' / ' characters to be used as the search pattern  
15948 delimiter. For consistency, IEEE Std. 1003.1-200x requires that the search delimiters used by the  
15949 `s`, `global`, and `v` commands be accepted as well.

### 15950 **Preserve**

15951 The `preserve` command does not historically cause the file to be considered unmodified for the  
15952 purposes of future commands that may exit the editor. IEEE Std. 1003.1-200x requires  
15953 conformance to historical practice.

15954 Historical documentation stated that mail was not sent to the user when `preserve` was executed;  
15955 however, historical implementations did send mail in this case. IEEE Std. 1003.1-200x requires  
15956 conformance to the historical implementations.

15957 **Print**

15958 The writing of NUL by the **print** command is not specified as a special case because the standard  
 15959 developers did not want to require *ex* to support NUL characters. Historically, characters were  
 15960 displayed using the ARPA standard mappings, which are as follows:

- 15961 1. Printable characters are left alone.
- 15962 2. Control characters less than \177 are represented as '^' followed by the character offset  
 15963 from the '@' character in the ASCII map; for example, \007 is represented as '^G'.
- 15964 3. \177 is represented as '^' followed by '?'.

15965 The display of characters having their eighth bit set was less standard. Existing implementations  
 15966 use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed bytes that had their  
 15967 eighth bit set as the two characters "M-" followed by the seven-bit display as described above.)  
 15968 The latter probably has the best claim to historical practice because it was used for the **-v** option  
 15969 of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

15970 No specific display format is required by IEEE Std. 1003.1-200x.

15971 Explicit dependence on the ASCII character set has been avoided where possible, hence the use  
 15972 of the phrase an "implementation-defined multi-character sequence" for the display of non-  
 15973 printable characters in preference to the historical usage of, for instance, "^I" for the <tab>  
 15974 character. Implementations are encouraged to conform to historical practice in the absence of  
 15975 any strong reason to diverge.

15976 Historically, all *ex* commands beginning with the letter 'p' could be entered using capitalized  
 15977 versions of the commands; for example, **P[rint]**, **Pre[serve]**, and **Pu[t]** were all valid command  
 15978 names. IEEE Std. 1003.1-200x permits, but does not require, this historical practice because  
 15979 capital forms of the commands are used by some implementations for other purposes.

15980 **Put**

15981 Historically, an *ex put* command, executed from open or visual mode, was the same as the open  
 15982 or visual mode **P** command, if the buffer was named and was cut in character mode, and the  
 15983 same as the **p** command if the buffer was named and cut in line mode. If the unnamed buffer  
 15984 was the source of the text, the entire line from which the text was taken was usually **put**, and the  
 15985 buffer was handled as if in line mode, but it was possible to get extremely anomalous behavior.  
 15986 In addition, using the **Q** command to switch into *ex* mode, and then doing a **put** often resulted in  
 15987 errors as well, such as appending text that was unrelated to the (supposed) contents of the  
 15988 buffer. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit  
 15989 these behaviors. All *ex put* commands are required to operate in line mode, and the contents of  
 15990 the buffers are not altered by changing the mode of the editor.

15991 **Read**

15992 Historically, an *ex read* command executed from open or visual mode, executed in an empty file,  
 15993 left an empty line as the first line of the file. For consistency and simplicity of specification,  
 15994 IEEE Std. 1003.1-200x does not permit this behavior. Historically, a **read** in open or visual mode  
 15995 from a program left the cursor at the last line read in, not the first. For consistency,  
 15996 IEEE Std. 1003.1-200x does not permit this behavior.

15997 Historical implementations of *ex* were unable to undo **read** commands that read from the output  
 15998 of a program. For consistency, IEEE Std. 1003.1-200x does not permit this behavior.

15999 Historically, the *ex* and *vi* message after a successful **read** or **write** command specified  
 16000 "characters", not "bytes". IEEE Std. 1003.1-200x requires that the number of bytes be displayed,



16001 not the number of characters, because it may be difficult in multi-byte implementations to  
 16002 determine the number of characters read. Implementations are encouraged to clarify the  
 16003 message displayed to the user.

16004 Historically, reads were not permitted on files other than type regular, except that FIFO files  
 16005 could be read (probably only because they did not exist when *ex* and *vi* were originally written).  
 16006 Because the historical *ex* evaluated **read!** and **read !** equivalently, there can be no optional way  
 16007 to force the read. IEEE Std. 1003.1-200x permits, but does not require, this behavior.

#### 16008 **Recover**

16009 Some historical implementations of the editor permitted users to recover the edit buffer contents  
 16010 from a previous edit session, and then exit without saving those contents (or explicitly  
 16011 discarding them). The intent of IEEE Std. 1003.1-200x in requiring that the edit buffer be treated  
 16012 as already modified is to prevent this user error.

#### 16013 **Rewind**

16014 Historical implementations supported the **rewind** command when the user was editing the first  
 16015 file in the list; that is, the file that the **rewind** command would edit. IEEE Std. 1003.1-200x  
 16016 requires conformance to historical practice.

#### 16017 **Substitute**

16018 Historically, *ex* accepted an **r** option to the **s** command. The effect of the **r** option was to use the  
 16019 last regular expression used in any command as the pattern, the same as the **~** command. The **r**  
 16020 option is not required by IEEE Std. 1003.1-200x. Historically, the **c** and **g** options were toggled;  
 16021 for example, the command **:s/abc/def/** was the same as **s/abc/def/ccccgggg**. For simplicity of  
 16022 specification, IEEE Std. 1003.1-200x does not permit this behavior.

16023 The tilde command is often used to replace the last search RE. For example, in the sequence:

```
16024 s/red/blue/
16025 /green
16026 ~
```

16027 the **~** command is equivalent to:

```
16028 s/green/blue/
```

16029 Historically, *ex* accepted all of the following forms:

```
16030 s/abc/def/
16031 s/abc/def
16032 s/abc/
16033 s/abc
```

16034 IEEE Std. 1003.1-200x requires conformance to this historical practice.

16035 The **s** command presumes that the **'^'** character only occupies a single column in the display.  
 16036 Much of the *ex* and *vi* specification presumes that the **<space>** character only occupies a single  
 16037 column in the display. There are no known character sets for which this is not true.

16038 Historically, the final column position for the substitute commands was based on previous  
 16039 column movements; a search for a pattern followed by a substitution would leave the column  
 16040 position unchanged, while a **0** command followed by a substitution would change the column  
 16041 position to the first non-**<blank>**. For consistency and simplicity of specification,  
 16042 IEEE Std. 1003.1-200x requires that the final column position always be set to the first non-  
 16043 **<blank>**.

16044

**Set**16045  
16046

Historical implementations redisplayed all of the options for each occurrence of the **all** keyword. IEEE Std. 1003.1-200x permits, but does not require, this behavior.

16047

**Tag**16048  
16049  
16050  
16051  
16052  
16053

No requirement is made as to where *ex* and *vi* shall look for the file referenced by the tag entry. Historical practice has been to look for the path found in the **tags** file, based on the current directory. A useful extension found in some implementations is to look based on the directory containing the tags file that held the entry, as well. No requirement is made as to which reference for the tag in the tags file is used. This is deliberate, in order to permit extensions such as multiple entries in a tags file for a tag.

16054  
16055  
16056  
16057

Because users often specify many different tags files, some of which need not be relevant or exist at any particular time, IEEE Std. 1003.1-200x requires that error messages about problem tags files be displayed only if the requested tag is not found, and then, only once for each time that the **tag** edit option is changed.

16058  
16059  
16060  
16061  
16062  
16063

The requirement that the current edit buffer be unmodified is only necessary if the file indicated by the tag entry is not the same as the current file (as defined by the current path name). Historically, the file would be reloaded if the file name had changed, as well as if the file name was different from the current path name. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior, requiring that the name be the only factor in the decision.

16064  
16065  
16066  
16067

Historically, *vi* only searched for tags in the current file from the current cursor to the end of the file, and therefore, if the **wrapscan** option was not set, tags occurring before the current cursor were not found. IEEE Std. 1003.1-200x considers this a bug, and implementations are required to search for the first occurrence in the file, regardless.

16068

**Undo**16069  
16070  
16071

The **undo** description deliberately uses the word “modified”. The **undo** command is not intended to undo commands that replace the contents of the edit buffer, such as **edit**, **next**, **tag**, or **recover**.

16072  
16073  
16074  
16075  
16076

Cursor positioning after the **undo** command was inconsistent in the historical *vi*, sometimes attempting to restore the original cursor position (**global**, **undo**, and **v** commands), and sometimes, in the presence of maps, placing the cursor on the last line added or changed instead of the first. IEEE Std. 1003.1-200x requires a simplified behavior for consistency and simplicity of specification.

16077

**Version**16078  
16079  
16080

The **version** command cannot be exactly specified since there is no widely-accepted definition of what the version information should contain. Implementations are encouraged to do something reasonably intelligent.

16081 **Write**

16082 Historically, the *ex* and *vi* message after a successful **read** or **write** command specified  
16083 “characters”, not “bytes”. IEEE Std. 1003.1-200x requires that the number of bytes be displayed,  
16084 not the number of characters because it may be difficult in multi-byte implementations to  
16085 determine the number of characters written. Implementations are encouraged to clarify the  
16086 message displayed to the user.

16087 Implementation-defined tests are permitted so that implementations can make additional  
16088 checks; for example, for locks or file modification times.

16089 Historically, attempting to append to a nonexistent file caused an error. It has been left  
16090 unspecified in IEEE Std. 1003.1-200x to permit implementations to let the **write** succeed, so that  
16091 the append semantics are similar to those of the historical *cs*.

16092 Historical *vi* permitted empty edit buffers to be written. However, since the way *vi* got around  
16093 dealing with “empty” files was to always have a line in the edit buffer, no matter what, it wrote  
16094 them as files of a single, empty line. IEEE Std. 1003.1-200x does not permit this behavior.

16095 Historically, *ex* restored standard output and standard error to their values as of when *ex* was  
16096 invoked, before writes to programs were performed. This could disturb the terminal  
16097 configuration as well as be a security issue for some terminals. IEEE Std. 1003.1-200x does not  
16098 permit this, requiring that the program output be captured and displayed as if by the *ex* **print**  
16099 command.

16100 **Adjust Window**

16101 Historically, the line count was set to the value of the **scroll** option if the type character was  
16102 end-of-file. This feature was broken on most historical implementations long ago, however, and  
16103 is not documented anywhere. For this reason, IEEE Std. 1003.1-200x is resolutely silent.

16104 Historically, the **z** command was <blank> character-sensitive and **z +** and **z -** did different  
16105 things than **z+** and **z-** because the type could not be distinguished from a flag. (The commands  
16106 **z .** and **z =** were historically invalid.) IEEE Std. 1003.1-200x requires conformance to this  
16107 historical practice.

16108 Historically, the **z** command was further <blank> character-sensitive in that the *count* could not  
16109 be <blank> character-delimited; for example, the commands **z= 5** and **z- 5** were also invalid.  
16110 Because the *count* is not ambiguous with respect to either the type character or the flags, this is  
16111 not permitted by IEEE Std. 1003.1-200x.

16112 **Escape**

16113 Historically, *ex* filter commands only read the standard output of the commands, letting  
16114 standard error appear on the terminal as usual. The *vi* utility, however, read both standard  
16115 output and standard error. IEEE Std. 1003.1-200x requires the latter behavior for both *ex* and *vi*,  
16116 for consistency.

16117 **Shift Left and Shift Right**

16118 Historically, it was possible to add shift characters to increase the effect of the command; for  
16119 example, <<< outdented (or >>> indented) the lines 3 levels of indentation instead of the default  
16120 1. IEEE Std. 1003.1-200x requires conformance to historical practice.

## 16121 &lt;control&gt;-D

16122 Historically, the <control>-D command erased the prompt, providing the user with an unbroken  
 16123 presentation of lines from the edit buffer. This is not required by IEEE Std. 1003.1-200x;  
 16124 implementations are encouraged to provide it if possible. Historically, the <control>-D  
 16125 command took, and then ignored, a *count*. IEEE Std. 1003.1-200x does not permit this behavior.

16126 **Write Line Number**

16127 Historically, the *ex =* command, when executed in *ex* mode in an empty edit buffer, reported 0,  
 16128 and from open or visual mode, reported 1. For consistency and simplicity of specification,  
 16129 IEEE Std. 1003.1-200x does not permit this behavior.

16130 **Execute**

16131 Historically, *ex* did not correctly handle the inclusion of text input commands (that is, **append**,  
 16132 **insert**, and **change**) in executed buffers. IEEE Std. 1003.1-200x does not permit this exclusion for  
 16133 consistency.

16134 Historically, the logical contents of the buffer being executed did not change if the buffer itself  
 16135 were modified by the commands being executed; that is, buffer execution did not support self-  
 16136 modifying code. IEEE Std. 1003.1-200x requires conformance to historical practice.

16137 Historically, the @ command took a range of lines, and the @ buffer was executed once per line,  
 16138 with the current line ( ' . ' ) set to each specified line. IEEE Std. 1003.1-200x requires conformance  
 16139 to historical practice.

16140 Some historical implementations did not notice if errors occurred during buffer execution. This,  
 16141 coupled with the ability to specify a range of lines for the *ex @* command, makes it trivial to  
 16142 cause them to drop core. IEEE Std. 1003.1-200x requires that implementations stop buffer  
 16143 execution if any error occurs, if the specified line doesn't exist, or if the contents of the edit buffer  
 16144 itself are replaced (for example, the buffer executes the *ex :edit* command).

16145 **Regular Expressions in ex**

16146 Historical practice is that the characters in the replacement part of the last *s* command—that is,  
 16147 those matched by entering a '~' in the regular expression—were not further expanded by the  
 16148 regular expression engine. So, if the characters contained the string "a . , " they would match  
 16149 'a' followed by " . , " and not 'a' followed by any character. IEEE Std. 1003.1-200x requires  
 16150 conformance to historical practice.

16151 **Edit Options in ex**

16152 The following paragraphs describe the historical behavior of some edit options that were not, for  
 16153 whatever reason, included in IEEE Std. 1003.1-200x. Implementations are strongly encouraged  
 16154 to only use these names if the functionality described here is fully supported.

16155 **extended** The **extended** edit option has been used in some implementations of *vi* to provide  
 16156 extended regular expressions instead of basic regular expressions. This option was  
 16157 omitted from IEEE Std. 1003.1-200x because it is not widespread historical practice.

16158 **flash** The **flash** edit option historically caused the screen to flash instead of beeping on  
 16159 error. This option was omitted from IEEE Std. 1003.1-200x because it is not found  
 16160 in some historical implementations.

16161 **hardtabs** The **hardtabs** edit option historically defined the number of columns between  
 16162 hardware tab settings. This option was omitted from IEEE Std. 1003.1-200x  
 16163 because it was believed to no longer be generally useful.

|       |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16164 | <b>modeline</b>       | The <b>modeline</b> (sometimes named <b>modelines</b> ) edit option historically caused <i>ex</i> or <i>vi</i> to read the five first and last lines of the file for editor commands. This option is a security problem, and vendors are strongly encouraged to delete it from historical implementations.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 16165 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16166 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16167 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16168 | <b>open</b>           | The <b>open</b> edit option historically disallowed the <i>ex</i> <b>open</b> and <b>visual</b> commands. This edit option was omitted because these commands are required by IEEE Std. 1003.1-200x.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 16169 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16170 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16171 | <b>optimize</b>       | The <b>optimize</b> edit option historically expedited text throughput by setting the terminal to not do automatic carriage returns when printing more than one logical line of output. This option was omitted from IEEE Std. 1003.1-200x because it was intended for terminals without addressable cursors, which are rarely, if ever, still used.                                                                                                                                                                                                                                                                                                                                                                                  |
| 16172 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16173 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16174 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16175 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16176 | <b>ruler</b>          | The <b>ruler</b> edit option has been used in some implementations of <i>vi</i> to present a current row/column ruler for the user. This option was omitted from IEEE Std. 1003.1-200x because it is not widespread historical practice.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 16177 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16178 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16179 | <b>sourceany</b>      | The <b>sourceany</b> edit option historically caused <i>ex</i> or <i>vi</i> to source start-up files that were owned by users other than the user running the editor. This option is a security problem, and vendors are strongly encouraged to remove it from their implementations.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 16180 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16181 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16182 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16183 | <b>timeout</b>        | The <b>timeout</b> edit option historically enabled the (now standard) feature of only waiting for a short period before returning keys that could be part of a macro. This feature was omitted from IEEE Std. 1003.1-200x because its behavior is now standard, it is not widely useful, and it was rarely documented.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 16184 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16185 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16186 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16187 | <b>verbose</b>        | The <b>verbose</b> edit option has been used in some implementations of <i>vi</i> to cause <i>vi</i> to output error messages for common errors; for example, attempting to move the cursor past the beginning or end of the line instead of only alerting the screen. (The historical <i>vi</i> only alerted the terminal and presented no message for such errors. The historical editor option <b>terse</b> did not select when to present error messages, it only made existing error messages more or less verbose.) This option was omitted from IEEE Std. 1003.1-200x because it is not widespread historical practice; however, implementors are encouraged to use it if they wish to provide error messages for naive users. |
| 16188 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16189 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16190 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16191 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16192 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16193 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16194 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16195 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16196 | <b>wraplen</b>        | The <b>wraplen</b> edit option has been used in some implementations of <i>vi</i> to specify an automatic margin measured from the left margin instead of from the right margin. This is useful when multiple screen sizes are being used to edit a single file. This option was omitted from IEEE Std. 1003.1-200x because it is not widespread historical practice; however, implementors are encouraged to use it if they add this functionality.                                                                                                                                                                                                                                                                                  |
| 16197 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16198 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16199 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16200 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16201 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16202 | <b>autoindent, ai</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16203 |                       | Historically, the command <b>Oa</b> did not do any autoindentation, regardless of the current indentation of line 1. IEEE Std. 1003.1-200x requires that any indentation present in line 1 be used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 16204 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 16205 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

16206 **autoprint, ap**

16207 Historically, the **autoprint** edit option was not completely consistent or based solely on  
 16208 modifications to the edit buffer. Exceptions were the **read** command (when reading from a file,  
 16209 but not from a filter), the **append, change, insert, global,** and **v** commands, all of which were not  
 16210 affected by **autoprint**, and the **tag** command, which was affected by **autoprint**.  
 16211 IEEE Std. 1003.1-200x requires conformance to historical practice.

16212 Historically, the **autoprint** option only applied to the last of multiple commands entered using  
 16213 vertical-bar delimiters; for example, **delete** <newline> was affected by **autoprint**, but  
 16214 **delete|version** <newline> was not. IEEE Std. 1003.1-200x requires conformance to historical  
 16215 practice.

16216 **autowrite, aw**

16217 Appending the '!' character to the *ex* **next** command to avoid performing an automatic write  
 16218 was not supported in historical implementations. IEEE Std. 1003.1-200x requires that the  
 16219 behavior match the other *ex* commands for consistency.

16220 **ignorecase, ic**

16221 Historical implementations of case-insensitive matching (the **ignorecase** edit option) lead to  
 16222 counterintuitive situations when uppercase characters were used in range expressions.  
 16223 Historically, the process was as follows:

- 16224 1. Take a line of text from the edit buffer.
- 16225 2. Convert uppercase to lowercase in text line.
- 16226 3. Convert uppercase to lowercase in regular expressions, except in character class  
 16227 specifications.
- 16228 4. Match regular expressions against text.

16229 This would mean that, with **ignorecase** in effect, the text:

16230 The cat sat on the mat

16231 would be matched by

16232 /^the/

16233 but not by:

16234 /^[A-Z]he/

16235 For consistency with other commands implementing regular expressions, IEEE Std. 1003.1-200x  
 16236 does not permit this behavior.

16237 **paragraphs, para**

16238 Earlier versions of IEEE Std. 1003.1-200x made the default **paragraphs** and **sections** edit options  
 16239 implementation-defined, arguing they were historically oriented to the UNIX system *troff* text  
 16240 formatter, and a “portable user” could use the {, }, [[, ]], (, and ) commands in open or visual  
 16241 mode and have the cursor stop in unexpected places. IEEE Std. 1003.1-200x specifies their values  
 16242 in the POSIX locale because the unusual grouping (they only work when grouped into two  
 16243 characters at a time) means that they cannot be used for general purpose movement, regardless.

- 16244           **readonly**
- 16245           Implementations are encouraged to provide the best possible information to the user as to the  
16246           read-only status of the file, with the exception that they should not consider the current special  
16247           privileges of the process. This provides users a safety net because they must force the overwrite  
16248           of read-only files, even when running with additional privileges.
- 16249           The **readonly** edit option specification largely conforms to historical practice. The only  
16250           difference is that historical implementations did not notice that the user had set the **readonly**  
16251           edit option in cases where the file was already marked read-only for some reason, and would  
16252           therefore reinitialize the **readonly** edit option the next time the contents of the edit buffer were  
16253           replaced. This behavior is disallowed by IEEE Std. 1003.1-200x.
- 16254           **report**
- 16255           The requirement that lines copied to a buffer interact differently than deleted lines is historical  
16256           practice. For example, if the **report** edit option is set to 3, deleting 3 lines will cause a report to be  
16257           written, but 4 lines must be copied before a report is written.
- 16258           The requirement that the **ex global**, **v**, **open**, **undo**, and **visual** commands present reports based  
16259           on the total number of lines added or deleted during the command execution, and that  
16260           commands executed by the **global** and **v** commands not present reports, is historical practice.  
16261           IEEE Std. 1003.1-200x extends historical practice by requiring that buffer execution be treated  
16262           similarly. The reasons for this are two-fold. Historically, only the report by the last command  
16263           executed from the buffer would be seen by the user, as each new report would overwrite the  
16264           last. In addition, the standard developers believed that buffer execution had more in common  
16265           with **global** and **v** commands than it did with other **ex** commands, and should behave similarly,  
16266           for consistency and simplicity of specification.
- 16267           **showmatch, sm**
- 16268           The length of time the cursor spends on the matching character is unspecified because the  
16269           timing capabilities of systems are often inexact and variable. The time should be long enough for  
16270           the user to notice, but not long enough for the user to become annoyed. Some implementations  
16271           of **vi** have added a **matchtime** option that permits users to set the number of 0,1 second intervals  
16272           the cursor pauses on the matching character.
- 16273           **showmode**
- 16274           The **showmode** option has been used in some historical implementations of **ex** and **vi** to display  
16275           the current editing mode when in open or visual mode. The editing modes have generally  
16276           included “command” and “input”, and sometimes other modes such as “replace” and  
16277           “change”. The string was usually displayed on the bottom line of the screen at the far right-hand  
16278           corner. In addition, a preceding ‘\*’ character often denoted if the contents of the edit buffer had  
16279           been modified. The latter display has sometimes been part of the **showmode** option, and  
16280           sometimes based on another option. This option was not available in the 4 BSD historical  
16281           implementation of **vi**, but was viewed as generally useful, particularly to novice users, and is  
16282           required by IEEE Std. 1003.1-200x.
- 16283           The **smd** shorthand for the **showmode** option was not present in all historical implementations  
16284           of the editor. IEEE Std. 1003.1-200x requires it, for consistency.
- 16285           Not all historical implementations of the editor displayed a mode string for command mode,  
16286           differentiating command mode from text input mode by the absence of a mode string.  
16287           IEEE Std. 1003.1-200x permits this behavior for consistency with historical practice, but  
16288           implementations are encouraged to provide a display string for both modes.

16289 **slowopen**

16290 Historically the **slowopen** option was automatically set if the terminal baud rate was less than  
 16291 1 200 baud, or if the baud rate was 1 200 baud and the **redraw** option was not set. The **slowopen**  
 16292 option had two effects. First, when inserting characters in the middle of a line, characters after  
 16293 the cursor would not be pushed ahead, but would appear to be overwritten. Second, when  
 16294 creating a new line of text, lines after the current line would not be scrolled down, but would  
 16295 appear to be overwritten. In both cases, ending text input mode would cause the screen to be  
 16296 refreshed to match the actual contents of the edit buffer. Finally, terminals that were sufficiently  
 16297 intelligent caused the editor to ignore the **slowopen** option. IEEE Std. 1003.1-200x permits most  
 16298 historical behavior, extending historical practice to require **slowopen** behaviors if the edit option  
 16299 is set by the user.

16300 **tags**

16301 The default path for tags files is left unspecified as implementations may have their own **tags**  
 16302 implementations that do not correspond to the historical ones. The default **tags** option value  
 16303 should probably at least include the file **./tags**.

16304 **term**

16305 Historical implementations of *ex* and *vi* ignored changes to the **term** edit option after the initial  
 16306 terminal information was loaded. This is permitted by IEEE Std. 1003.1-200x; however,  
 16307 implementations are encouraged to permit the user to modify their terminal type at any time.

16308 **terse**

16309 Historically, the **terse** edit option optionally provided a shorter, less descriptive error message,  
 16310 for some error messages. This is permitted, but not required, by IEEE Std. 1003.1-200x.  
 16311 Historically, most common visual mode errors (for example, trying to move the cursor past the  
 16312 end of a line) did not result in an error message, but simply alerted the terminal.  
 16313 Implementations wishing to provide messages for novice users are urged to do so based on the  
 16314 **edit** option **verbose**, and not **terse**.

16315 **window**

16316 In historical implementations, the default for the **window** edit option was based on the baud  
 16317 rate as follows:

16318 1. If the baud rate was less than 1 200, the **edit** option **w300** set the window value; for  
 16319 example, the line:

```
16320 set w300=12
```

16321 would set the window option to 12 if the baud rate was less than 1 200.

16322 2. If the baud rate was equal to 1 200, the **edit** option **w1200** set the window value.

16323 3. If the baud rate was greater than 1 200, the **edit** option **w9600** set the window value.

16324 The **w300**, **w1200**, and **w9600** options do not appear in IEEE Std. 1003.1-200x because of their  
 16325 dependence on specific baud rates.

16326 In historical implementations, the size of the window displayed by various commands was  
 16327 related to, but not necessarily the same as, the **window** edit option. For example, the size of the  
 16328 window was set by the *ex* command **visual 10**, but it did not change the value of the **window**  
 16329 edit option. However, changing the value of the **window** edit option did change the number of  
 16330 lines that were displayed when the screen was repainted. IEEE Std. 1003.1-200x does not permit



16331 this behavior in the interests of consistency and simplicity of specification, and requires that all  
 16332 commands that change the number of lines that are displayed do it by setting the value of the  
 16333 **window** edit option.

#### 16334 **wrapmargin, wm**

16335 Historically, the **wrapmargin** option did not affect maps inserting characters that also had  
 16336 associated *counts*; for example **:map K 5aABC DEF**. Unfortunately, there are widely used  
 16337 maps that depend on this behavior. For consistency and simplicity of specification,  
 16338 IEEE Std. 1003.1-200x does not permit this behavior.

16339 Historically, **wrapmargin** was calculated using the column display width of all characters on the  
 16340 screen. For example, an implementation using "**^I**" to represent <tab> characters when the **list**  
 16341 edit option was set, where '**^**' and '**I**' each took up a single column on the screen, would  
 16342 calculate the **wrapmargin** based on a value of 2 for each <tab> character. The **number** edit  
 16343 option similarly changed the effective length of the line as well. IEEE Std. 1003.1-200x requires  
 16344 conformance to historical practice.

#### 16345 **FUTURE DIRECTIONS**

16346 None.

#### 16347 **SEE ALSO**

16348 *ed, sed, stty, vi*, the System Interfaces volume of IEEE Std. 1003.1-200x, *access()*

#### 16349 **CHANGE HISTORY**

16350 First released in Issue 2.

#### 16351 **Issue 4**

16352 Aligned with the ISO/IEC 9945-2:1993 standard.

#### 16353 **Issue 5**

16354 The FUTURE DIRECTIONS section is added.

#### 16355 **Issue 6**

16356 This utility is now marked as part of the User Portability Utilities option.

16357 The obsolescent SYNOPSIS is removed, removing the *+command* and *-* options.

16358 The following new requirements on POSIX implementations derive from alignment with the  
 16359 Single UNIX Specification:

- 16360 • The **-l** option is added.
- 16361 • In the *map* command description, the sequence *#digit* is added.
- 16362 • The **directory**, **edcompatible**, **redraw**, **slowopen**, and **lisp** edit options are added.

16363 The *ex* utility is extensively changed for alignment with the IEEE P1003.2b draft standard. This  
 16364 includes changes as a result of the IEEE PASC Interpretations 1003.2 #31, #38, #49, #50, #51, #52,  
 16365 #55, #56, #57, #61, #62, #63, #64, #65, and #78.

16366 **NAME**

16367           expand — convert tabs to spaces

16368 **SYNOPSIS**16369 UP        expand [-t *tablist*][*file* ...]

16370

16371 **DESCRIPTION**

16372       The *expand* utility shall write files or the standard input to the standard output with <tab>  
16373 characters replaced with one or more <space> characters needed to pad to the next tab stop. Any  
16374 <backspace> characters shall be copied to the output and cause the column position count for  
16375 tab stop calculations to be decremented; the column position count shall not be decremented  
16376 below zero.

16377 **OPTIONS**

16378       The *expand* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
16379 12.2, Utility Syntax Guidelines.

16380       The following option shall be supported:

16381       -t *tablist*     Specify the tab stops. The application shall ensure that the argument *tablist*  
16382 consists of either a single positive decimal integer or a list of tabstops. If a single  
16383 number is given, tabs shall be set that number of column positions apart instead of  
16384 the default 8.

16385       If a list of tabstops is given, the application shall ensure that it consists of a list of  
16386 two or more positive decimal integers, separated by <blank> characters or comms,  
16387 in ascending order. The tabs shall be set at those specific column positions. Each  
16388 tab stop *N* shall be an integer value greater than zero, and the list is in strictly  
16389 ascending order. This is taken to mean that, from the start of a line of output,  
16390 tabbing to position *N* shall cause the next character output to be in the (*N*+1)th  
16391 column position on that line.

16392       In the event of *expand* having to process a <tab> character at a position beyond the  
16393 last of those specified in a multiple tab-stop list, the <tab> character shall be  
16394 replaced by a single <space> character in the output.

16395 **OPERANDS**

16396       The following operand shall be supported:

16397       *file*            The path name of a text file to be used as input.

16398 **STDIN**

16399       See the INPUT FILES section.

16400 **INPUT FILES**

16401       Input files shall be text files.

16402 **ENVIRONMENT VARIABLES**

16403       The following environment variables shall affect the execution of *expand*:

16404       *LANG*            Provide a default value for the internationalization variables that are unset or null.  
16405 If *LANG* is unset or null, the corresponding value from the implementation-  
16406 defined default locale shall be used. If any of the internationalization variables  
16407 contains an invalid setting, the utility shall behave as if none of the variables had  
16408 been defined.

16409       *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
16410 internationalization variables.

- 16411 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
16412 characters (for example, single-byte as opposed to multi-byte characters in  
16413 arguments and input files), the processing of <tab> and <space> characters, and  
16414 for the determination of the width in column positions each character would  
16415 occupy on an output device.
- 16416 **LC\_MESSAGES**  
16417 Determine the locale that should be used to affect the format and contents of  
16418 diagnostic messages written to standard error.
- 16419 **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 16420 **ASYNCHRONOUS EVENTS**  
16421 Default.
- 16422 **STDOUT**  
16423 The standard output shall be equivalent to the input files with <tab> characters converted into  
16424 the appropriate number of <space> characters.
- 16425 **STDERR**  
16426 Used only for diagnostic messages.
- 16427 **OUTPUT FILES**  
16428 None.
- 16429 **EXTENDED DESCRIPTION**  
16430 None.
- 16431 **EXIT STATUS**  
16432 The following exit values shall be returned:  
16433 0 Successful completion  
16434 >0 An error occurred.
- 16435 **CONSEQUENCES OF ERRORS**  
16436 The *expand* utility shall terminate with an error message and non-zero exit status upon  
16437 encountering difficulties accessing one of the *file* operands.
- 16438 **APPLICATION USAGE**  
16439 None.
- 16440 **EXAMPLES**  
16441 None.
- 16442 **RATIONALE**  
16443 The *expand* utility is useful for preprocessing text files (before sorting, looking at specific  
16444 columns, and so on) that contain <tab>s.  
16445 See the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.106, Column Position.  
16446 The *tablist* option-argument consists of integers in ascending order. Utility Syntax Guideline 8  
16447 mandates that *expand* shall accept the integers (within the single argument) separated using  
16448 either commas or <blank>s.
- 16449 **FUTURE DIRECTIONS**  
16450 None.

16451 **SEE ALSO**16452 *tabs, unexpand*16453 **CHANGE HISTORY**

16454 First released in Issue 4.

16455 **Issue 6**

16456 This utility is now marked as part of the User Portability Utilities option.

16457 The APPLICATION USAGE section is added.

16458 The obsolescent SYNOPSIS is removed.

16459 The *LC\_CTYPE* environment variable description is updated to align with the IEEE P1003.2b  
16460 draft standard.

16461 The normative text is reworded to avoid use of the term “must” for application requirements.

16462 **NAME**16463           *expr* — evaluate arguments as an expression16464 **SYNOPSIS**16465           *expr operand*16466 **DESCRIPTION**16467           The *expr* utility shall evaluate an expression and write the result to standard output.16468 **OPTIONS**

16469           None.

16470 **OPERANDS**

16471           The single expression evaluated by *expr* shall be formed from the operands, as described in the  
 16472           EXTENDED DESCRIPTION section. The application shall ensure that each of the expression  
 16473           operator symbols:

16474           ( ) | &amp; = &gt; &gt;= &lt; &lt;= != + - \* / % :

16475           and the symbols *integer* and *string* in the table are provided as separate arguments to *expr*.16476 **STDIN**

16477           Not used.

16478 **INPUT FILES**

16479           None.

16480 **ENVIRONMENT VARIABLES**16481           The following environment variables shall affect the execution of *expr*:

16482           *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 16483           If *LANG* is unset or null, the corresponding value from the implementation-  
 16484           defined default locale shall be used. If any of the internationalization variables  
 16485           contains an invalid setting, the utility shall behave as if none of the variables had  
 16486           been defined.

16487           *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 16488           internationalization variables.

16489           *LC\_COLLATE*

16490           Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 16491           character collating elements within regular expressions and by the string  
 16492           comparison operators.

16493           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 16494           characters (for example, single-byte as opposed to multi-byte characters in  
 16495           arguments) and the behavior of character classes within regular expressions.

16496           *LC\_MESSAGES*

16497           Determine the locale that should be used to affect the format and contents of  
 16498           diagnostic messages written to standard error.

16499 *XSI*       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

16500 **ASYNCHRONOUS EVENTS**

16501           Default.

16502 **STDOUT**

16503 The *expr* utility shall evaluate the expression and write the result, followed by a <newline>  
 16504 character, to standard output.

16505 **STDERR**

16506 Used only for diagnostic messages.

16507 **OUTPUT FILES**

16508 None.

16509 **EXTENDED DESCRIPTION**

16510 The formation of the expression to be evaluated is shown in the following table. The symbols  
 16511 *expr*, *expr1*, and *expr2* represent expressions formed from *integer* and *string* symbols and the  
 16512 expression operator symbols (all separate arguments) by recursive application of the constructs  
 16513 described in the table. The expressions are listed in order of increasing precedence, with equal-  
 16514 precedence operators grouped between horizontal lines. All of the operators shall be left-  
 16515 associative.

| Expression                                                                                                                                                                                | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>expr1</i>   <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if it is neither null nor zero; otherwise, returns the evaluation of <i>expr2</i> if it is not null; otherwise, zero.                                                                                                                                                                                                                                               |
| <i>expr1</i> & <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if neither expression evaluates to null or zero; otherwise, returns zero.                                                                                                                                                                                                                                                                                           |
| <i>expr1</i> = <i>expr2</i><br><i>expr1</i> > <i>expr2</i><br><i>expr1</i> >= <i>expr2</i><br><i>expr1</i> < <i>expr2</i><br><i>expr1</i> <= <i>expr2</i><br><i>expr1</i> != <i>expr2</i> | Returns the result of a decimal integer comparison if both arguments are integers; otherwise, returns the result of a string comparison using the locale-specific collation sequence. The result of each comparison is 1 if the specified relationship is true, or 0 if the relationship is false.<br>Equal.<br>Greater than.<br>Greater than or equal.<br>Less than.<br>Less than or equal.<br>Not equal. |
| <i>expr1</i> + <i>expr2</i><br><i>expr1</i> - <i>expr2</i>                                                                                                                                | Addition of decimal integer-valued arguments.<br>Subtraction of decimal integer-valued arguments.                                                                                                                                                                                                                                                                                                          |
| <i>expr1</i> * <i>expr2</i><br><i>expr1</i> / <i>expr2</i><br><i>expr1</i> % <i>expr2</i>                                                                                                 | Multiplication of decimal integer-valued arguments.<br>Integer division of decimal integer-valued arguments, producing an integer result.<br>Remainder of integer division of decimal integer-valued arguments.                                                                                                                                                                                            |
| <i>expr1</i> : <i>expr2</i>                                                                                                                                                               | Matching expression; see below.                                                                                                                                                                                                                                                                                                                                                                            |
| ( <i>expr</i> )                                                                                                                                                                           | Grouping symbols. Any expression can be placed within parentheses. Parentheses can be nested to a depth of {EXPR_NEST_MAX}.                                                                                                                                                                                                                                                                                |
| <i>integer</i><br><i>string</i>                                                                                                                                                           | An argument consisting only of an (optional) unary minus followed by digits.<br>A string argument; see below.                                                                                                                                                                                                                                                                                              |

16547 **Matching Expression**

16548 The '=' matching operator shall compare the string resulting from the evaluation of *expr1* with  
 16549 the regular expression pattern resulting from the evaluation of *expr2*. Regular expression syntax  
 16550 shall be that defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3, Basic  
 16551 Regular Expressions, except that all patterns are anchored to the beginning of the string (that is,  
 16552 only sequences starting at the first character of a string are matched by the regular expression)  
 16553 and, therefore, it is unspecified whether '^' is a special character in that context. Usually, the  
 16554 matching operator shall return a string representing the number of characters matched ('0' on  
 16555 failure). Alternatively, if the pattern contains at least one regular expression subexpression  
 16556 "[\\(\\.\\.\\.\\)]", the string corresponding to "\\1" shall be returned.

16557 **String Operand**

16558 A string argument is an argument that cannot be identified as an *integer* argument or as one of  
 16559 the expression operator symbols shown in the OPERANDS section.

16560 The use of string arguments **length**, **substr**, **index**, or **match** produces unspecified results.

16561 **EXIT STATUS**

16562 The following exit values shall be returned:

16563 0 The *expression* evaluates to neither null nor zero.

16564 1 The *expression* evaluates to null or zero.

16565 2 Invalid *expression*.

16566 >2 An error occurred.

16567 **CONSEQUENCES OF ERRORS**

16568 Default.

16569 **APPLICATION USAGE**

16570 After argument processing by the shell, *expr* is not required to be able to tell the difference  
 16571 between an operator and an operand except by the value. If "\$a" is '=', the command:

```
16572 expr $a = '='
```

16573 looks like:

```
16574 expr = = =
```

16575 as the arguments are passed to *expr* (and they all may be taken as the '=' operator). The  
 16576 following works reliably:

```
16577 expr X$a = X=
```

16578 Also note that this volume of IEEE Std. 1003.1-200x permits implementations to extend utilities.  
 16579 The *expr* utility permits the integer arguments to be preceded with a unary minus. This means  
 16580 that an integer argument could look like an option. Therefore, the portable application must  
 16581 employ the "—" construct of Guideline 10 of the Base Definitions volume of  
 16582 IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines to protect its operands if there is  
 16583 any chance the first operand might be a negative integer (or any string with a leading minus).

16584 **EXAMPLES**

16585 The *expr* utility has a rather difficult syntax:

- 16586 • Many of the operators are also shell control operators or reserved words, so they have to be  
 16587 escaped on the command line.

- Each part of the expression is composed of separate arguments, so liberal usage of <blank> characters is required. For example:

16588  
16589  
16590  
16591  
16592  
16593  
16594

| Invalid                       | Valid                              |
|-------------------------------|------------------------------------|
| <code>expr 1+2</code>         | <code>expr 1 + 2</code>            |
| <code>expr "1 + 2"</code>     | <code>expr 1 + 2</code>            |
| <code>expr 1 + (2 * 3)</code> | <code>expr 1 + \( 2 \* 3 \)</code> |

16595 In many cases, the arithmetic and string features provided as part of the shell command  
16596 language are easier to use than their equivalents in *expr*. Newly written scripts should avoid  
16597 *expr* in favor of the new features within the shell; see Section 2.5 (on page 2241) and Section 2.6.4  
16598 (on page 2248).

16599 The following command:

```
16600 a=$(expr $a + 1)
```

16601 adds 1 to the variable *a*.

16602 The following command, for "*\$a*" equal to either */usr/abc/file* or just *file*:

```
16603 expr $a : '.*\/(.*\)' \| $a
```

16604 returns the last segment of a path name (that is, *file*). Applications should avoid the character  
16605 */* used alone as an argument: *expr* may interpret it as the division operator.

16606 The following command:

```
16607 expr "//$a" : '.*\/(.*\)'
```

16608 is a better representation of the previous example. The addition of the *"/*" characters  
16609 eliminates any ambiguity about the division operator and simplifies the whole expression. Also  
16610 note that path names may contain characters contained in the *IFS* variable and should be quoted  
16611 to avoid having "*\$a*" expand into multiple arguments.

16612 The following command:

```
16613 expr "$VAR" : '.*'
```

16614 returns the number of characters in *VAR*.

#### 16615 RATIONALE

16616 In an early proposal, EREs were used in the matching expression syntax. This was changed to  
16617 BREs to avoid breaking historical applications.

16618 The use of a leading circumflex in the BRE is unspecified because many historical  
16619 implementations have treated it as a special character, despite their system documentation. For  
16620 example:

```
16621 expr foo : ^foo expr ^foo : ^foo
```

16622 return 3 and 0, respectively, on those systems; their documentation would imply the reverse.  
16623 Thus, the anchoring condition is left unspecified to avoid breaking historical scripts relying on  
16624 this undocumented feature.

#### 16625 FUTURE DIRECTIONS

16626 None.



16627 **SEE ALSO**

16628 Section 2.6.4

16629 **CHANGE HISTORY**

16630 First released in Issue 2.

16631 **Issue 4**

16632 Aligned with the ISO/IEC 9945-2: 1993 standard.

16633 **Issue 5**

16634 FUTURE DIRECTIONS section added.

16635 **Issue 6**16636 The *expr* utility is aligned with the IEEE P1003.2b draft standard, to include resolution of IEEE |

16637 PASC Interpretation 1003.2 #104. |

16638 The normative text is reworded to avoid use of the term “must” for application requirements.

16639 **NAME**

16640 false — return false value

16641 **SYNOPSIS**

16642 false

16643 **DESCRIPTION**

16644 The *false* utility shall return with a non-zero exit code.

16645 **OPTIONS**

16646 None.

16647 **OPERANDS**

16648 None.

16649 **STDIN**

16650 Not used.

16651 **INPUT FILES**

16652 None.

16653 **ENVIRONMENT VARIABLES**

16654 None.

16655 **ASYNCHRONOUS EVENTS**

16656 Default.

16657 **STDOUT**

16658 Not used.

16659 **STDERR**

16660 None.

16661 **OUTPUT FILES**

16662 None.

16663 **EXTENDED DESCRIPTION**

16664 None.

16665 **EXIT STATUS**

16666 The *false* utility always shall exit with a value other than zero.

16667 **CONSEQUENCES OF ERRORS**

16668 Default.

16669 **APPLICATION USAGE**

16670 None.

16671 **EXAMPLES**

16672 None.

16673 **RATIONALE**

16674 None.

16675 **FUTURE DIRECTIONS**

16676 None.

16677 **SEE ALSO**

16678 *true*

16679 **CHANGE HISTORY**

16680 First released in Issue 2.

16681 **Issue 4**

16682 Aligned with the ISO/IEC 9945-2: 1993 standard.

## 16683 NAME

16684 fc — process the command history list

## 16685 SYNOPSIS

16686 UP `fc [-r][-e editor] [first[last]]`16687 `fc -l[-nr] [first[last]]`16688 `fc -s[old=new][first]`

16689

## 16690 DESCRIPTION

16691 The *fc* utility shall list, or shall edit and re-execute, commands previously entered to an  
16692 interactive *sh*.

16693 The command history list shall reference commands by number. The first number in the list is  
16694 selected arbitrarily. The relationship of a number to its command shall not change except when  
16695 the user logs in and no other process is accessing the list, at which time the system may reset the  
16696 numbering to start the oldest retained command at another number (usually 1). When the  
16697 number reaches an implementation-defined upper limit, which shall be no smaller than the  
16698 value in *HISTSIZE* or 32 767 (whichever is greater), the shell may wrap the numbers, starting the  
16699 next command with a lower number (usually 1). However, despite this optional wrapping of  
16700 numbers, *fc* shall maintain the time-ordering sequence of the commands. For example, if four  
16701 commands in sequence are given the numbers 32 766, 32 767, 1 (wrapped), and 2 as they are  
16702 executed, command 32 767 is considered the command previous to 1, even though its number is  
16703 higher.

16704 When commands are edited (when the *-l* option is not specified), the resulting lines shall be  
16705 entered at the end of the history list and then re-executed by *sh*. The *fc* command that caused the  
16706 editing shall not be entered into the history list. If the editor returns a non-zero exit status, this  
16707 shall suppress the entry into the history list and the command re-execution. Any command line  
16708 variable assignments or redirection operators used with *fc* shall affect both the *fc* command itself  
16709 as well as the command that results; for example:

16710 `fc -s -- -l 2>/dev/null`16711 reinvokes the previous command, suppressing standard error for both *fc* and the previous  
16712 command.

## 16713 OPTIONS

16714 The *fc* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
16715 Utility Syntax Guidelines.

16716 The following options shall be supported:

16717 **-e editor** Use the editor named by *editor* to edit the commands. The *editor* string is a utility  
16718 name, subject to search via the *PATH* variable (see the Base Definitions volume of  
16719 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables). The value in the *FCEDIT*  
16720 variable shall be used as a default when *-e* is not specified. If *FCEDIT* is null or  
16721 unset, *ed* shall be used as the editor.

16722 **-l** (The letter ell.) List the commands rather than invoking an editor on them. The  
16723 commands shall be written in the sequence indicated by the *first* and *last* operands,  
16724 as affected by *-r*, with each command preceded by the command number.

16725 **-n** Suppress command numbers when listing with *-l*.

16726 **-r** Reverse the order of the commands listed (with *-l*) or edited (with neither *-l* nor  
16727 *-s*).

- 16728            -s            Reexecute the command without invoking an editor.
- 16729 **OPERANDS**
- 16730            The following operands shall be supported:
- 16731            *first, last*
- 16732                            Select the commands to list or edit. The number of previous commands that can be  
16733                            accessed shall be determined by the value of the *HISTSIZE* variable. The value of  
16734                            *first* or *last* or both shall be one of the following:
- 16735            [+]*number*    A positive number representing a command number; command  
16736                            numbers can be displayed with the -l option.
- 16737            -*number*        A negative decimal number representing the command that was  
16738                            executed *number* of commands previously. For example, -1 is the  
16739                            immediately previous command.
- 16740            *string*        A string indicating the most recently entered command that begins  
16741                            with that string. If the *old=new* operand is not also specified with -s,  
16742                            the string form of the *first* operand cannot contain an embedded  
16743                            equal sign.
- 16744                            When the synopsis form with -s is used:
- 16745
  - If *first* is omitted, the previous command shall be used.
- 16746                            For the synopsis forms without -s:
- 16747
  - If *last* is omitted, *last* shall default to the previous command when -l is  
16748                            specified; otherwise, it shall default to *first*.
  - If *first* and *last* are both omitted, the previous 16 commands shall be listed or  
16750                            the previous single command shall be edited (based on the -l option).
  - If *first* and *last* are both present, all of the commands from *first* to *last* shall be  
16751                            edited (without -l) or listed (with -l). Editing multiple commands shall be  
16752                            accomplished by presenting to the editor all of the commands at one time, each  
16753                            command starting on a new line. If *first* represents a newer command than *last*,  
16754                            the commands shall be listed or edited in reverse sequence, equivalent to using  
16755                            -r. For example, the following commands on the first line are equivalent to the  
16756                            corresponding commands on the second:  
16757

```
16758 fc -r 10 20 fc 30 40
16759 fc 20 10 fc -r 40 30
```

16760                            
  - When a range of commands is used, it shall not be an error to specify *first* or *last*  
16761                            values that are not in the history list; *fc* shall substitute the value representing  
16762                            the oldest or newest command in the list, as appropriate. For example, if there  
16763                            are only ten commands in the history list, numbered 1 to 10:

```
16764 fc -l
16765 fc 1 99
```

16766                            shall list and edit, respectively, all ten commands.

16767            *old=new*        Replace the first occurrence of string *old* in the commands to be re-executed by the  
16768                            string *new*.

16769 **STDIN**

16770 Not used.

16771 **INPUT FILES**

16772 None.

16773 **ENVIRONMENT VARIABLES**16774 The following environment variables shall affect the execution of *fc*:

16775 **FCEDIT** This variable, when expanded by the shell, shall determine the default value for  
 16776 the *-e editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be  
 16777 used as the editor.

16778 **HISTFILE** Determine a path name naming a command history file. If the *HISTFILE* variable is  
 16779 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 16780 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 16781 and write access to, or create, the history file, it shall use an unspecified  
 16782 mechanism that allows the history to operate properly. (References to history  
 16783 "file" in this section shall be understood to mean this unspecified mechanism in  
 16784 such cases.) An implementation may choose to access this variable only when  
 16785 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 16786 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 16787 by the user, the file named by the *ENV* variable, or implementation-defined system  
 16788 start-up files. In some historical shells, the history file is initialized just after the  
 16789 *ENV* file has been processed. Therefore, it is implementation-defined whether  
 16790 changes made to *HISTFILE* after the history file has been initialized are effective.  
 16791 Implementations may choose to disable the history list mechanism for users with  
 16792 appropriate privileges who do not set *HISTFILE*; the specific circumstances under  
 16793 which this occurs are implementation-defined. If more than one instance of the  
 16794 shell is using the same history file, it is unspecified how updates to the history file  
 16795 from those shells interact. As entries are deleted from the history file, they shall be  
 16796 deleted oldest first. It is unspecified when history file entries are physically  
 16797 removed from the history file.

16798 **HISTSIZE** Determine a decimal number representing the limit to the number of previous  
 16799 commands that are accessible. If this variable is unset, an unspecified default  
 16800 greater than or equal to 128 shall be used. The maximum number of commands in  
 16801 the history list is unspecified, but shall be at least 128. An implementation may  
 16802 choose to access this variable only when initializing the history file, as described  
 16803 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSIZE*  
 16804 after the history file has been initialized are effective.

16805 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 16806 If *LANG* is unset or null, the corresponding value from the implementation-  
 16807 defined default locale shall be used. If any of the internationalization variables  
 16808 contains an invalid setting, the utility shall behave as if none of the variables had  
 16809 been defined.

16810 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 16811 internationalization variables.

16812 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 16813 characters (for example, single-byte as opposed to multi-byte characters in  
 16814 arguments and input files).

16815 **LC\_MESSAGES**

16816 Determine the locale that should be used to affect the format and contents of

- 16817 diagnostic messages written to standard error.
- 16818 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 16819 **ASYNCHRONOUS EVENTS**
- 16820 Default.
- 16821 **STDOUT**
- 16822 When the **-l** option is used to list commands, the format of each command in the list shall be as
- 16823 follows:
- 16824 "%d\t%s\n", <line number>, <command>
- 16825 If both the **-l** and **-n** options are specified, the format of each command shall be:
- 16826 "\t%s\n", <command>
- 16827 If the <command> consists of more than one line, the lines after the first shall be displayed as:
- 16828 "\t%s\n", <continued-command>
- 16829 **STDERR**
- 16830 Used only for diagnostic messages.
- 16831 **OUTPUT FILES**
- 16832 None.
- 16833 **EXTENDED DESCRIPTION**
- 16834 None.
- 16835 **EXIT STATUS**
- 16836 The following exit values shall be returned:
- 16837 0 Successful completion of the listing.
- 16838 >0 An error occurred.
- 16839 Otherwise, the exit status shall be that of the commands executed by *fc*.
- 16840 **CONSEQUENCES OF ERRORS**
- 16841 Default.
- 16842 **APPLICATION USAGE**
- 16843 Since editors sometimes use file descriptors as integral parts of their editing, redirecting their file
- 16844 descriptors as part of the *fc* command can produce unexpected results. For example, if *vi* is the
- 16845 *FCEDIT* editor, the command:
- 16846 `fc -s | more`
- 16847 does not work correctly on many systems.
- 16848 Users on windowing systems may want to have separate history files for each window by
- 16849 setting *HISTFILE* as follows:
- 16850 `HISTFILE=$HOME/.sh_hist$$`
- 16851 **EXAMPLES**
- 16852 None.
- 16853 **RATIONALE**
- 16854 This utility is based on the *fc* built-in of the KornShell.
- 16855 An early proposal specified the **-e** option as [**-e** *editor* [*old= new* ]], which is not historical
- 16856 practice. Historical practice in *fc* of either [**-e** *editor*] or [**-e** - [*old= new* ]] is acceptable, but not

16857 both together. To clarify this, a new option `-s` was introduced replacing the `[-e -]`. This resolves  
16858 the conflict and makes `fc` conform to the Utility Syntax Guidelines.

16859 **HISTFILE** Some implementations of the KornShell check for the superuser and do not create  
16860 a history file unless `HISTFILE` is set. This is done primarily to avoid creating  
16861 unlinked files in the root file system when logging in during single-user mode.  
16862 `HISTFILE` must be set for the superuser to have history.

16863 **HISTSIZE** Needed to limit the size of history files. It is the intent of the standard developers  
16864 that when two shells share the same history file, commands that are entered in one  
16865 shell shall be accessible by the other shell. Because of the difficulties of  
16866 synchronization over a network, the exact nature of the interaction is unspecified.

16867 The initialization process for the history file can be dependent on the system start-up files, in  
16868 that they may contain commands that effectively preempt the settings the user has for `HISTFILE`  
16869 and `HISTSIZE`. For example, function definition commands are recorded in the history file. If the  
16870 system administrator includes function definitions in some system start-up file called before the  
16871 `ENV` file, the history file is initialized before the user can influence its characteristics. In some  
16872 historical shells, the history file is initialized just after the `ENV` file has been processed. Because  
16873 of these situations, the text requires the initialization process to be implementation-defined.

16874 Consideration was given to omitting the `fc` utility in favor of the command line editing feature in  
16875 `sh`. For example, in `vi` editing mode, typing "`<ESC> v`" is equivalent to:

```
16876 EDITOR=vi fc
```

16877 However, the `fc` utility allows the user the flexibility to edit multiple commands simultaneously  
16878 (such as `fc 10 20`) and to use editors other than those supported by `sh` for command line editing.

16879 In the KornShell, the alias `r` ("re-do") is preset to `fc -e -` (equivalent to the POSIX `fc -s`). This is  
16880 probably an easier command name to remember than `fc` ("fix command"), but it does not meet  
16881 the Utility Syntax Guidelines. Renaming `fc` to `hist` or `redo` was considered, but since this  
16882 description closely matches historical KornShell practice already, such a renaming was seen as  
16883 gratuitous. Users are free to create aliases whenever odd historical names such as `fc`, `awk`, `cat`,  
16884 `grep`, or `yacc` are standardized by POSIX.

16885 Command numbers have no ordering effects; they are like serial numbers. The `-r` option and  
16886 `-number` operand address the sequence of command execution, regardless of serial numbers. So,  
16887 for example, if the command number wrapped back to 1 at some arbitrary point, there would be  
16888 no ambiguity associated with traversing the wrap point. For example, if the command history  
16889 were:

```
16890 32766: echo 1
16891 32767: echo 2
16892 1: echo 3
```

16893 the number `-2` refers to command 32 767 because it is the second previous command, regardless  
16894 of serial number.

#### 16895 **FUTURE DIRECTIONS**

16896 None.

#### 16897 **SEE ALSO**

16898 `sh`



16899 **CHANGE HISTORY**

16900 First released in Issue 4.

16901 **Issue 5**

16902 FUTURE DIRECTIONS section added.

16903 **Issue 6**

16904 This utility is now marked as part of the User Portability Utilities option.

16905 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
16906 “directory referred to by the *HOME* environment variable”.

16907 **NAME**

16908 fg — run jobs in the foreground

16909 **SYNOPSIS**16910 UP fg [*job\_id*]

16911

16912 **DESCRIPTION**16913 If job control is enabled (see the description of *set -m*), the *fg* utility shall move a background job  
16914 from the current environment (see Section 2.13 (on page 2273)) into the foreground.16915 Using *fg* to place a job into the foreground shall remove its process ID from the list of those  
16916 “known in the current shell execution environment”; see Section 2.9.3.1 (on page 2259).16917 **OPTIONS**

16918 None.

16919 **OPERANDS**

16920 The following operand shall be supported:

16921 *job\_id* Specify the job to be run as a foreground job. If no *job\_id* operand is given, the  
16922 *job\_id* for the job that was most recently suspended, placed in the background or  
16923 run as a background job, shall be used. The format of *job\_id* is described in the Base  
16924 Definitions volume of IEEE Std. 1003.1-200x, Section 3.205, Job Control Job ID.16925 **STDIN**

16926 Not used.

16927 **INPUT FILES**

16928 None.

16929 **ENVIRONMENT VARIABLES**16930 The following environment variables shall affect the execution of *fg*:16931 *LANG* Provide a default value for the internationalization variables that are unset or null.  
16932 If *LANG* is unset or null, the corresponding value from the implementation-  
16933 defined default locale shall be used. If any of the internationalization variables  
16934 contains an invalid setting, the utility shall behave as if none of the variables had  
16935 been defined.16936 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
16937 internationalization variables.16938 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
16939 characters (for example, single-byte as opposed to multi-byte characters in  
16940 arguments).16941 *LC\_MESSAGES*16942 Determine the locale that should be used to affect the format and contents of  
16943 diagnostic messages written to standard error.16944 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.16945 **ASYNCHRONOUS EVENTS**

16946 Default.

16947 **STDOUT**16948 The *fg* utility shall write the command line of the job to standard output in the following format:16949 "%s\n", <*command*>

16950 **STDERR**

16951           Used only for diagnostic messages.

16952 **OUTPUT FILES**

16953           None.

16954 **EXTENDED DESCRIPTION**

16955           None.

16956 **EXIT STATUS**

16957           The following exit values shall be returned:

16958           0   Successful completion.

16959           >0  An error occurred.

16960 **CONSEQUENCES OF ERRORS**

16961           If job control is disabled, the *fg* utility shall exit with an error and no job shall be placed in the foreground.

16963 **APPLICATION USAGE**

16964           The *fg* utility does not work as expected when it is operating in its own utility execution environment because that environment has no applicable jobs to manipulate. See the APPLICATION USAGE section for *bg* (on page 2422). For this reason, *fg* is generally implemented as a shell regular built-in.

16968 **EXAMPLES**

16969           None.

16970 **RATIONALE**

16971           The extensions to the shell specified in this volume of IEEE Std. 1003.1-200x have mostly been based on features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs* are also based on the KornShell. The standard developers examined the characteristics of the C shell versions of these utilities and found that differences exist. Despite widespread use of the C shell, the KornShell versions were selected for this volume of IEEE Std. 1003.1-200x to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command line editing features).

16978 **FUTURE DIRECTIONS**

16979           None.

16980 **SEE ALSO**

16981           *bg*, *kill*, *jobs*, *wait*

16982 **CHANGE HISTORY**

16983           First released in Issue 4.

16984 **Issue 6**

16985           This utility is now marked as part of the User Portability Utilities option.

16986           The APPLICATION USAGE section is added.

16987           The JC marking is removed from the SYNOPSIS since job control is mandatory in this issue.

## 16988 NAME

16989 file — determine file type

## 16990 SYNOPSIS

16991 UP file [-dhi][-M file][-m file] file ...

16992

## 16993 DESCRIPTION

16994 The *file* utility shall perform a series of tests on each specified *file* in an attempt to classify it:16995 1. If the file is not a regular file, its file type shall be identified. The file types directory, FIFO,  
16996 block special, and character special shall be identified as such. Other implementation-  
16997 defined file types may also be identified.

16998 2. If the file is a regular file, and:

16999 a. The file is zero-length, it shall be identified as an empty file.

17000 b. The file is not zero-length, *file* shall examine an initial segment of the file and shall  
17001 make a guess at identifying its contents or whether it is an executable binary file.  
17002 (The answer is not guaranteed to be correct.)17003 If *file* does not exist, cannot be read, or its file status could not be determined, the output shall  
17004 indicate that the file was processed, but that its type could not be determined.17005 If *file* is a symbolic link, by default the link shall be resolved and *file* shall test the type of file  
17006 referenced by the symbolic link.

## 17007 OPTIONS

17008 The *file* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
17009 12.2, Utility Syntax Guidelines.

17010 The following options shall be supported by the implementation:

17011 **-d** Apply any default system tests to the file.17012 **-h** When a symbolic link is encountered, identify the file as a symbolic link. If **-h** is  
17013 not specified and *file* is a symbolic link that refers to a nonexistent file, *file* shall  
17014 identify the file as a symbolic link, as if **-h** had been specified.17015 **-i** If a file is a regular file, do not attempt to classify the type of the file further, but  
17016 identify the file as specified in the STDOUT section, using a *<type>* string that  
17017 contains the string "regular file".17018 **-M file** Specify the name of a file containing tests that shall be applied to a file in order to  
17019 classify it (see the EXTENDED DESCRIPTION). No default system tests shall be  
17020 applied.17021 **-m file** Specify the name of a file containing tests that shall be applied to a file in order to  
17022 classify it (see the EXTENDED DESCRIPTION).17023 If multiple instances of the **-m**, **-d**, or **-M** options are specified, the concatenation of the tests  
17024 specified, in the order specified, shall be the set of tests that are applied. If a **-M** option is  
17025 specified, no tests other than those specified using the **-d**, **-M**, and **-m** options shall be applied  
17026 to the file. If neither the **-d** nor **-M** options are specified, any default system tests shall be  
17027 applied after any tests specified using the **-m** option.

17028 **OPERANDS**

17029           The following operand shall be supported:

17030           *file*           A path name of a file to be tested.

17031 **STDIN**

17032           Not used.

17033 **INPUT FILES**

17034           The *file* can be any file type.

17035 **ENVIRONMENT VARIABLES**

17036           The following environment variables shall affect the execution of *file*:

17037           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
 17038                           If *LANG* is unset or null, the corresponding value from the implementation-  
 17039                           defined default locale shall be used. If any of the internationalization variables  
 17040                           contains an invalid setting, the utility shall behave as if none of the variables had  
 17041                           been defined.

17042           *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 17043                           internationalization variables.

17044           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 17045                           characters (for example, single-byte as opposed to multi-byte characters in  
 17046                           arguments and input files).

17047           *LC\_MESSAGES*

17048                           Determine the locale that should be used to affect the format and contents of  
 17049                           diagnostic messages written to standard error and informative messages written to  
 17050                           standard output.

17051 *XSI*           *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

17052 **ASYNCHRONOUS EVENTS**

17053           Default.

17054 **STDOUT**

17055           In the POSIX locale, the following format shall be used to identify each operand, *file* specified:

17056           "%s: %s\n", <*file*>, <*type*>

17057           The values for <*type*> are unspecified, except that in the POSIX locale, if *file* is identified as one  
 17058           of the types listed in the following table, <*type*> shall contain (but is not limited to) the  
 17059           corresponding string. Each space shown in the strings shall be exactly one <space> character.

17060

Table 4-8 File Utility Output Strings

17061

| If file is a:                                                | <type> shall contain the string: |
|--------------------------------------------------------------|----------------------------------|
| Directory                                                    | directory                        |
| FIFO                                                         | fifo                             |
| Block special                                                | block special                    |
| Character special                                            | character special                |
| Executable binary                                            | executable                       |
| Empty regular file                                           | empty                            |
| Symbolic link                                                | symbolic link to                 |
| <i>ar</i> archive library (see <i>ar</i> )                   | archive                          |
| Extended <i>cpio</i> format (see <i>pax</i> )                | cpio archive                     |
| Extended <i>tar</i> format (see <b>ustar</b> in <i>pax</i> ) | tar archive                      |
| Shell script                                                 | commands text                    |
| C-language source                                            | c program text                   |
| FORTRAN source                                               | fortran program text             |

17062

17063

17064

17065

17066

17067

17068

17069

17070

17071

17072

17073

17074

17075

If *file* is identified as a symbolic link (see **-h**), the following alternative output format shall be used:

17076

17077

"%s: %s %s\n", <file>, <type>, <contents of link>"

17078

If the file named by the *file* operand does not exist or cannot be read, the string "cannot open" shall be included as part of the <type> field, but this shall not be considered an error that affects the exit status. If the type of the file named by the *file* operand cannot be determined, the string "data" shall be included as part of the <type> field, but this shall not be considered an error that affects the exit status.

17079

17080

17081

17082

17083 **STDERR**

17084

Used only for diagnostic messages.

17085 **OUTPUT FILES**

17086

None.

17087 **EXTENDED DESCRIPTION**

17088

A file specified as an option-argument to the **-m** or **-M** options shall contain one test per line, which shall be applied to the file. If the test succeeds, the message field of the line shall be printed and no further tests shall be applied, with the exception that tests on immediately following lines beginning with a single '**>**' character shall be applied.

17089

17090

17091

17092

Each line shall be composed of the following four <blank>-separated fields:

17093

*offset*

An unsigned number (optionally preceded by a single '**>**' character) specifying the *offset*, in bytes, of the value in the file that is to be compared against the *value* field of the line. If the file is shorter than the specified offset, the test shall fail.

17094

17095

17096

17097

17098

17099

17100

17101

If the *offset* begins with the character '**>**', the test contained in the line shall not be applied to the file unless the test on the last line for which the *offset* did not begin with a '**>**' was successful. By default, the *offset* shall be interpreted as an unsigned decimal number. With a leading 0x or 0X, the *offset* shall be interpreted as a hexadecimal number; otherwise, with a leading 0, the *offset* shall be interpreted as an octal number.

17102

*type*

The type of the value in the file to be tested. The type shall consist of the type specification characters **c**, **d**, **f**, **s**, and **u**, specifying character, signed decimal, floating point, string, and unsigned decimal, respectively.

17103

17104

17105 The *type* string shall be interpreted as the bytes from the file starting at the  
 17106 specified *offset* and including the same number of bytes specified by the *value* field.  
 17107 If insufficient bytes remain in the file past the *offset* to match the *value* field, the test  
 17108 shall fail.

17109 The type specification characters **d**, **f**, and **u** can be followed by an optional  
 17110 unsigned decimal integer that specifies the number of bytes represented by the  
 17111 type. The type specification character **f** can be followed by an optional **F**, **D**, or **L**,  
 17112 indicating that the value is of type **float**, **double**, or **long double**, respectively. The  
 17113 type specification characters **d** and **u** can be followed by an optional **C**, **S**, **I**, or **L**,  
 17114 indicating that the value is of type **char**, **short**, **int**, or **long**, respectively.

17115 The default number of bytes represented by the type specifiers **d**, **f**, and **u** shall  
 17116 correspond to their respective C-language types as follows. If the system claims  
 17117 conformance to the C-Language Development Utilities option, those specifiers  
 17118 shall correspond to the default sizes used in the *c99* utility. Otherwise, the default  
 17119 sizes shall be implementation-defined.

17120 For the type specifier characters **d** and **u**, the default number of bytes shall  
 17121 correspond to the size of a basic integer type of the implementation. For these  
 17122 specifier characters, the implementation shall support values of the optional  
 17123 number of bytes to be converted corresponding to the number of bytes in the C-  
 17124 language types **char**, **short**, **int**, or **long**. These numbers can also be specified by an  
 17125 application as the characters **C**, **S**, **I**, and **L**, respectively. The byte order used when  
 17126 interpreting numeric values is implementation-defined, but shall correspond to the  
 17127 order in which a constant of the corresponding type is stored in memory on the  
 17128 system.

17129 For the type specifier **f**, the default number of bytes shall correspond to the number  
 17130 of bytes in the basic double precision floating-point data type of the underlying  
 17131 implementation. The implementation shall support values of the optional number  
 17132 of bytes to be converted corresponding to the number of bytes in the C-language  
 17133 types **float**, **double**, and **long double**. These numbers can also be specified by an  
 17134 application as the characters **F**, **D**, and **L**, respectively.

17135 All type specifiers, except for **s**, can be followed by a mask specifier of the form  
 17136 *&number*. The mask value shall be AND'ed with the value before the comparison  
 17137 with the value from the file is made. By default, the mask shall be interpreted as an  
 17138 unsigned decimal number. With a leading 0x or 0X, the mask shall be interpreted  
 17139 as an unsigned hexadecimal number; otherwise, with a leading 0, the mask shall be  
 17140 interpreted as an unsigned octal number.

17141 The strings **byte**, **short**, **long**, and **string** shall also be supported as type fields,  
 17142 being interpreted as **dC**, **dS**, **dL**, and **s**, respectively.

17143       *value*       The *value* to be compared with the value from the file.

17144 Any value that contains a character that is not a digit, other than a leading sign  
 17145 ('+' or '-') or a leading 0x or 0X, shall be interpreted as a string. The test shall  
 17146 succeed only when a string value exactly matches the bytes from the file.

17147 If the *value* is a string, it can contain the following sequences:

17148       \*character*       The backslash-escape sequences as specified in the Base  
 17149                           Definitions volume of IEEE Std. 1003.1-200x, Table 5-1, Escape  
 17150                           Sequences and Associated Actions ('\\', '\a', '\b', '\f',  
 17151                           '\n', '\r', '\t', '\v'). The results of using any other

17152 character, other than an octal digit, following the backslash are  
 17153 unspecified.

17154 *\octal* Octal sequences that can be used to represent characters with  
 17155 specific coded values. An octal sequence shall consist of a  
 17156 backslash followed by the longest sequence of one, two, or three  
 17157 octal-digit characters (01234567). If the size of a byte on the  
 17158 system is greater than 9 bits, the valid escape sequence used to  
 17159 represent a byte is implementation-defined.

17160 By default, any value that is not a string shall be interpreted as a signed decimal  
 17161 number. Any such value, with a leading 0x or 0X, shall be interpreted as an  
 17162 unsigned hexadecimal number; otherwise, with a leading zero, the value shall be  
 17163 interpreted as an unsigned octal number.

17164 If the value is not a string, it can be preceded by a character indicating the  
 17165 comparison to be performed. Permissible characters and the comparisons they  
 17166 specify are as follows:

17167 = The test shall succeed if the value from the file equals the *value* field.  
 17168 < The test shall succeed if the value from the file is less than the *value* field.  
 17169 > The test shall succeed if the value from the file is greater than the *value* field.  
 17170 & The test shall succeed if all of the bits in the *value* field are set in the value  
 17171 from the file.

17172 ^ The test shall succeed if at least one of the bits in the *value* field is not set in the  
 17173 value from the file.

17174 x The test shall succeed if there is any value in the file.

17175 *message* The *message* to be printed if the test succeeds. The *message* shall be interpreted  
 17176 using the notation for the *printf* formatting specification; see *printf*. If the *value*  
 17177 field was a string, then the value from the file shall be the argument for the *printf*  
 17178 formatting specification; otherwise, the value from the file shall be the argument.

#### 17179 EXIT STATUS

17180 The following exit values shall be returned:

17181 0 Successful completion.

17182 >0 An error occurred.

#### 17183 CONSEQUENCES OF ERRORS

17184 Default.

#### 17185 APPLICATION USAGE

17186 The *file* utility can only be required to guess at many of the file types because only exhaustive  
 17187 testing can determine some types with certainty. For example, binary data on some systems  
 17188 might match the initial segment of an executable or a *tar* archive.

17189 Note that the table indicates that the output contains the stated string. Systems may add text  
 17190 before or after the string. For executables, as an example, the machine architecture and various  
 17191 facts about how the file was link-edited may be included.



17192 **EXAMPLES**

17193 Determine whether an argument is a binary executable file:

```
17194 file "$1" | grep -Fq executable &&
17195 printf "%s is executable.\n" "$1"
```

17196 **RATIONALE**

17197 The `-f` option was omitted because the same effect can (and should) be obtained using the *xargs*  
17198 utility.

17199 Historical versions of the *file* utility attempt to identify the following types of files: symbolic link,  
17200 directory, character special, block special, socket, *tar* archive, *cpio* archive, *SCCS* archive, archive  
17201 library, empty, *compress* output, *pack* output, binary data, C source, FORTRAN source, assembler  
17202 source, *nroff*/*troff*/*eqn*/*tbl* source *troff* output, shell script, C shell script, English text, ASCII text,  
17203 various executables, APL workspace, compiled terminfo entries, and *CURSES* screen images.  
17204 Only those types that are reasonably well specified in POSIX or are directly related to POSIX  
17205 utilities are listed in the table.

17206 Implementations that support symbolic links are encouraged to use the string "symbolic  
17207 link" to identify them.

17208 Historical systems have used a "magic file" named `/etc/magic` to help identify file types. Because  
17209 it is generally useful for users and scripts to be able to identify special file types, the `-m` flag and  
17210 a portable format for user-created magic files has been specified. No requirement is made that an  
17211 implementation of *file* use this method of identifying files, only that users be permitted to add  
17212 their own classifying tests.

17213 In addition, three options have been added to historical practice. The `-d` flag has been added to  
17214 permit users to cause their tests to follow any default system tests. The `-i` flag has been added to  
17215 permit users to test portably for regular files in shell scripts. The `-M` flag has been added to  
17216 permit users to ignore any default system tests.

17217 The historical `-c` option was omitted as not particularly useful to users or portable shell scripts.  
17218 In addition, a reasonable implementation of the *file* utility would report any errors found each  
17219 time the magic file is read.

17220 The historical format of the magic file was the same as that specified by the Rationale in the  
17221 previous version of IEEE Std. 1003.1-200x for the *offset*, *value*, and *message* fields; however, it  
17222 used less precise type fields than the format specified by the current normative text. The new  
17223 type field values are a superset of the historical ones.

17224 The following is an example magic file:

```
17225 0 short 070707 cpio archive
17226 0 short 0143561 Byte-swapped cpio archive
17227 0 string 070707 ASCII cpio archive
17228 0 long 0177555 Very old archive
17229 0 short 0177545 Old archive
17230 0 short 017437 Old packed data
17231 0 string \037\036 Packed data
17232 0 string \377\037 Compacted data
17233 0 string \037\235 Compressed data
17234 >2 byte&0x80 >0 Block compressed
17235 >2 byte&0x1f x %d bits
17236 0 string \032\001 Compiled Terminfo Entry
17237 0 short 0433 Curses screen image
17238 0 short 0434 Curses screen image
```

|       |   |        |                    |                                                                                               |
|-------|---|--------|--------------------|-----------------------------------------------------------------------------------------------|
| 17239 | 0 | string | <ar>               | System V Release 1 archive                                                                    |
| 17240 | 0 | string | !<arch>\n__.SYMDEF | Archive random library                                                                        |
| 17241 | 0 | string | !<arch>            | Archive                                                                                       |
| 17242 | 0 | string | ARF_BEGARF         | PHIGS clear text archive                                                                      |
| 17243 | 0 | long   | 0x137A2950         | Scalable OpenFont binary                                                                      |
| 17244 | 0 | long   | 0x137A2951         | Encrypted scalable OpenFont binary                                                            |
| 17245 |   |        |                    | The use of a basic integer data type is intended to allow the implementation to choose a word |
| 17246 |   |        |                    | size commonly used by applications on that architecture.                                      |
| 17247 |   |        |                    | <b>FUTURE DIRECTIONS</b>                                                                      |
| 17248 |   |        |                    | None.                                                                                         |
| 17249 |   |        |                    | <b>SEE ALSO</b>                                                                               |
| 17250 |   |        |                    | <i>ls</i>                                                                                     |
| 17251 |   |        |                    | <b>CHANGE HISTORY</b>                                                                         |
| 17252 |   |        |                    | First released in Issue 4.                                                                    |
| 17253 |   |        |                    | <b>Issue 6</b>                                                                                |
| 17254 |   |        |                    | This utility is now marked as part of the User Portability Utilities option.                  |
| 17255 |   |        |                    | Options and an EXTENDED DESCRIPTION are added as specified in the IEEE P1003.2b draft         |
| 17256 |   |        |                    | standard.                                                                                     |

17257 **NAME**

17258 find — find files

17259 **SYNOPSIS**17260 find [-H | -L] *path* ... [*operand\_expression* ...]17261 **DESCRIPTION**

17262 The *find* utility shall recursively descend the directory hierarchy from each file specified by *path*,  
 17263 evaluating a Boolean expression composed of the primaries described in the OPERANDS section  
 17264 for each file encountered.

17265 The *find* utility shall be able to descend to arbitrary depths in a file hierarchy and shall not fail  
 17266 due to path length limitations (unless a *path* operand specified by the application exceeds  
 17267 {PATH\_MAX} requirements).

17268 The *find* utility shall detect infinite loops; that is, entering a previously visited directory that is an  
 17269 ancestor of the last file encountered. When it detects an infinite loop, *find* shall write a  
 17270 diagnostic message to standard error and shall either recover its position in the hierarchy or  
 17271 terminate.

17272 **OPTIONS**

17273 The *find* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 17274 12.2, Utility Syntax Guidelines.

17275 The following options shall be supported by the implementation:

17276 **-H** Cause the file information and file type evaluated for each symbolic link  
 17277 encountered on the command line to be those of the file referenced by the link, and  
 17278 not the link itself. If the referenced file does not exist, the file information and type  
 17279 shall be for the link itself. File information for all symbolic links not on the  
 17280 command line shall be that of the link itself.

17281 **-L** Cause the file information and file type evaluated for each symbolic link to be  
 17282 those of the file referenced by the link, and not the link itself.

17283 Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered  
 17284 an error. The last option specified shall determine the behavior of the utility.

17285 **OPERANDS**

17286 The following operands shall be supported:

17287 The *path* operand is a path name of a starting point in the directory hierarchy.

17288 The first argument that starts with a '-', or is a '!' or a '(', and all subsequent arguments  
 17289 shall be interpreted as an *expression* made up of the following primaries and operators. In the  
 17290 descriptions, wherever *n* is used as a primary argument, it shall be interpreted as a decimal  
 17291 integer optionally preceded by a plus ('+') or minus ('-') sign, as follows:

17292 **+n** More than *n*.

17293 **n** Exactly *n*.

17294 **-n** Less than *n*.

17295 The following primaries shall be supported:

17296 **-name** *pattern*

17297 The primary shall evaluate as true if the basename of the file name being examined  
 17298 matches *pattern* using the pattern matching notation described in Section 2.14 (on  
 17299 page 2274).

- 17300       **-nouser**       The primary shall evaluate as true if the file belongs to a user ID for which the  
17301       *getpwuid()* function defined in the System Interfaces volume of  
17302       IEEE Std. 1003.1-200x (or equivalent) returns NULL.
- 17303       **-nogroup**       The primary shall evaluate as true if the file belongs to a group ID for which the  
17304       *getgrgid()* function defined in the System Interfaces volume of  
17305       IEEE Std. 1003.1-200x (or equivalent) returns NULL.
- 17306       **-xdev**         The primary always shall evaluate as true; it shall cause *find* not to continue  
17307       descending past directories that have a different device ID (*st\_dev*, see the *stat()*  
17308       function defined in the System Interfaces volume of IEEE Std. 1003.1-200x). If any  
17309       **-xdev** primary is specified, it shall apply to the entire expression even if the **-xdev**  
17310       primary would not normally be evaluated.
- 17311       **-prune**        The primary always shall evaluate as true; it shall cause *find* not to descend the  
17312       current path name if it is a directory. If the **-depth** primary is specified, the **-prune**  
17313       primary shall have no effect.
- 17314       **-perm [-]mode**  
17315       The *mode* argument is used to represent file mode bits. It shall be identical in  
17316       format to the *symbolic\_mode* operand described in *chmod* (on page 2450), and shall  
17317       be interpreted as follows. To start, a template shall be assumed with all file mode  
17318       bits cleared. An *op* symbol of '+' shall set the appropriate mode bits in the  
17319       template; '-' shall clear the appropriate bits; '=' shall set the appropriate mode  
17320       bits, without regard to the contents of process' file mode creation mask. The *op*  
17321       symbol of '-' cannot be the first character of *mode*; this avoids ambiguity with the  
17322       optional leading hyphen. Since the initial mode is all bits off, there are not any  
17323       symbolic modes that need to use '-' as the first character.
- 17324       If the hyphen is omitted, the primary shall evaluate as true when the file  
17325       permission bits exactly match the value of the resulting template.
- 17326       Otherwise, if *mode* is prefixed by a hyphen, the primary shall evaluate as true if at  
17327       least all the bits in the resulting template are set in the file permission bits.
- 17328       **-perm [-]onum**  
17329       If the hyphen is omitted, the primary shall evaluate as true when the file  
17330       permission bits exactly match the value of the octal number *onum* and only the bits  
17331       corresponding to the octal mask 07777 shall be compared. (See the description of  
17332       the octal *mode* in *chmod* (on page 2450).) Otherwise, if *onum* is prefixed by a  
17333       hyphen, the primary shall evaluate as true if at least all of the bits specified in *onum*  
17334       that are also set in the octal mask 07777 are set.
- 17335       **-type c**       The primary shall evaluate as true if the type of the file is *c*, where *c* is 'b', 'c',  
17336       'd', 'l', 'p', 'f', or 's' for block special file, character special file, directory,  
17337       symbolic link, FIFO, regular file, or socket, respectively.
- 17338       **-links n**       The primary shall evaluate as true if the file has *n* links.
- 17339       **-user uname**   The primary shall evaluate as true if the file belongs to the user *uname*. If *uname* is  
17340       a decimal integer and the *getpwnam()* (or equivalent) function does not return a  
17341       valid user name, *uname* shall be interpreted as a user ID.
- 17342       **-group gname**  
17343       The primary shall evaluate as true if the file belongs to the group *gname*. If *gname*  
17344       is a decimal integer and the *getgrnam()* (or equivalent) function does not return a  
17345       valid group name, *gname* shall be interpreted as a group ID.

- 17346        **-size** *n*[*c*]     The primary shall evaluate as true if the file size in bytes, divided by 512 and  
 17347 rounded up to the next integer, is *n*. If *n* is followed by the character '*c*', the size  
 17348 shall be in bytes.
- 17349        **-atime** *n*       The primary shall evaluate as true if the file access time subtracted from the  
 17350 initialization time, divided by 86 400 (with any remainder discarded), is *n*.
- 17351        **-ctime** *n*       The primary shall evaluate as true if the time of last change of file status  
 17352 information subtracted from the initialization time, divided by 86 400 (with any  
 17353 remainder discarded), is *n*.
- 17354        **-mtime** *n*       The primary shall evaluate as true if the file modification time subtracted from the  
 17355 initialization time, divided by 86 400 (with any remainder discarded), is *n*.
- 17356        **-exec** *utility\_name* [*argument . . .*];  
 17357            The primary shall evaluate as true if the invoked utility *utility\_name* returns a zero  
 17358 value as exit status. The end of the primary expression shall be punctuated by a  
 17359 semicolon. A *utility\_name* or *argument* containing only the two characters "{}"  
 17360 shall be replaced by the current path name. If a *utility\_name* or argument string  
 17361 contains the two characters "{}", but not just the two characters "{}", it is  
 17362 implementation-defined whether *find* replaces those two characters with the  
 17363 current path name or uses the string without change. The current directory for the  
 17364 invocation of *utility\_name* shall be the same as the current directory when the *find*  
 17365 utility was started. If the *utility\_name* names any of the special built-in utilities in  
 17366 Section 2.15 (on page 2276), the results are undefined.
- 17367        **-ok** *utility\_name* [*argument . . .*];  
 17368            The **-ok** primary shall be equivalent to **-exec**, except that *find* shall request  
 17369 affirmation of the invocation of *utility\_name* using the current file as an argument  
 17370 by writing to standard error as described in the STDERR section. If the response on  
 17371 standard input is affirmative, the utility shall be invoked. Otherwise, the command  
 17372 shall not be invoked and the value of the **-ok** operand shall be false.
- 17373        **-print**        The primary always shall evaluate as true; it shall cause the current path name to  
 17374 be written to standard output.
- 17375        **-newer file**    The primary shall evaluate as true if the modification time of the current file is  
 17376 more recent than the modification time of the file named by the path name *file*.
- 17377        **-depth**        The primary shall always evaluate as true; it shall cause descent of the directory  
 17378 hierarchy to be done so that all entries in a directory are acted on before the  
 17379 directory itself. If a **-depth** primary is not specified, all entries in a directory shall  
 17380 be acted on after the directory itself. If any **-depth** primary is specified, it shall  
 17381 apply to the entire expression even if the **-depth** primary would not normally be  
 17382 evaluated.
- 17383        The primaries can be combined using the following operators (in order of decreasing  
 17384 precedence):
- 17385        (*expression*)   True if *expression* is true.
- 17386        !*expression*   Negation of a primary; the unary NOT operator.
- 17387        *expression* [**-a**] *expression*  
 17388            Conjunction of primaries; the AND operator is implied by the juxtaposition of two  
 17389 primaries or made explicit by the optional **-a** operator. The second expression  
 17390 shall not be evaluated if the first expression is false.

- 17391 *expression -o expression*  
 17392 Alternation of primaries; the OR operator. The second expression shall not be  
 17393 evaluated if the first expression is true.
- 17394 If no *expression* is present, **-print** shall be used as the expression. Otherwise, if the given  
 17395 expression does not contain any of the primaries **-exec**, **-ok**, or **-print**, the given expression shall  
 17396 be effectively replaced by:
- 17397 ( *given\_expression* ) -print
- 17398 The **-user**, **-group**, and **-newer** primaries each shall evaluate their respective arguments only  
 17399 once.
- 17400 **STDIN**
- 17401 If the **-ok** primary is used, the response shall be read from the standard input. An entire line  
 17402 shall be read as the response. Otherwise, the standard input shall not be used.
- 17403 **INPUT FILES**
- 17404 None.
- 17405 **ENVIRONMENT VARIABLES**
- 17406 The following environment variables shall affect the execution of *find*:
- 17407 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 17408 If *LANG* is unset or null, the corresponding value from the implementation-  
 17409 defined default locale shall be used. If any of the internationalization variables  
 17410 contains an invalid setting, the utility shall behave as if none of the variables had  
 17411 been defined.
- 17412 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 17413 internationalization variables.
- 17414 *LC\_COLLATE*  
 17415 Determine the locale for the behavior of ranges, equivalence classes and multi-  
 17416 character collating elements used in the pattern matching notation for the **-n**  
 17417 option and in the extended regular expression defined for the **yesexpr** locale  
 17418 keyword in the *LC\_MESSAGES* category.
- 17419 *LC\_CTYPE* This variable determines the locale for the interpretation of sequences of bytes of  
 17420 text data as characters (for example, single-byte as opposed to multi-byte  
 17421 characters in arguments), the behavior of character classes within the pattern  
 17422 matching notation used for the **-n** option, and the behavior of character classes  
 17423 within regular expressions used in the extended regular expression defined for the  
 17424 **yesexpr** locale keyword in the *LC\_MESSAGES* category.
- 17425 *LC\_MESSAGES*  
 17426 Determine the locale for the processing of affirmative responses that should be  
 17427 used to affect the format and contents of diagnostic messages written to standard  
 17428 error.
- 17429 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 17430 *PATH* Determine the location of the *utility\_name* for the **-exec** and **-ok** primaries, as  
 17431 described in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8,  
 17432 Environment Variables.

17433 **ASYNCHRONOUS EVENTS**

17434 Default.

17435 **STDOUT**

17436 The **-print** primary shall cause the current path names to be written to standard output. The  
 17437 format shall be:

17438 "%s\n", &lt;path&gt;

17439 **STDERR**

17440 The **-ok** primary shall write a prompt to standard error containing at least the *utility\_name* to be  
 17441 invoked and the current path name. In the POSIX locale, the last non-<blank> character in the  
 17442 prompt shall be '?'. The exact format used is unspecified.

17443 Otherwise, the standard error shall be used only for diagnostic messages.

17444 **OUTPUT FILES**

17445 None.

17446 **EXTENDED DESCRIPTION**

17447 None.

17448 **EXIT STATUS**

17449 The following exit values shall be returned:

17450 0 All *path* operands were traversed successfully.

17451 &gt;0 An error occurred.

17452 **CONSEQUENCES OF ERRORS**

17453 Default.

17454 **APPLICATION USAGE**

17455 When used in operands, pattern matching notation, semicolons, opening parentheses, and  
 17456 closing parentheses are special to the shell and must be quoted (see Section 2.2 (on page 2236)).

17457 The bit that is traditionally used for sticky (historically 01000) is specified in the **-perm** primary  
 17458 using the octal number argument form. Since this bit is not defined by this volume of  
 17459 IEEE Std. 1003.1-200x, applications must not assume that it actually refers to the traditional  
 17460 sticky bit.

17461 **EXAMPLES**

17462 1. The following commands are equivalent:

17463 `find .`17464 `find . -print`

17465 They both write out the entire directory hierarchy from the current directory.

17466 2. The following command:

17467 `find / \( -name tmp -o -name '*.xx' \) -atime +7 -exec rm {} \;`

17468 removes all files named **tmp** or ending in **.xx** that have not been accessed for seven or more  
 17469 24-hour periods.

17470 3. The following command:

17471 `find . -perm -o+w,+s`

17472 prints (**-print** is assumed) the names of all files in or below the current directory, with all  
 17473 of the file permission bits **S\_ISUID**, **S\_ISGID**, and **S\_IWOTH** set.

- 17474 4. The following command:
- ```
17475 find . -name SCCS -prune -o -print
```
- 17476 recursively prints path names of all files in the current directory and below, but skips
17477 directories named SCCS and files in them.
- 17478 5. The following command:
- ```
17479 find . -print -name SCCS -prune
```
- 17480 behaves as in the previous example, but prints the names of the SCCS directories.
- 17481 6. The following command is roughly equivalent to the **-nt** extension to *test*:
- ```
17482 if [ -n "$(find file1 -prune -newer file2)" ]; then
17483     printf %s\n "file1 is newer than file2"
17484 fi
```
- 17485 7. The descriptions of **-atime**, **-ctime**, and **-mtime** use the terminology *n* “86 400 second
17486 periods (days)”. For example, a file accessed at 23:59 is selected by:
- ```
17487 find . -atime -1 -print
```
- 17488 at 00:01 the next day (less than 24 hours later, not more than one day ago); the midnight  
17489 boundary between days has no effect on the 24-hour calculation.

#### 17490 RATIONALE

17491 The **-a** operator was retained as an optional operator for compatibility with historical shell  
17492 scripts, even though it is redundant with expression concatenation.

17493 The descriptions of the ‘-’ modifier on the *mode* and *onum* arguments to the **-perm** primary  
17494 agree with historical practice on BSD and System V implementations. System V and BSD  
17495 documentation both describe it in terms of checking additional bits; in fact, it uses the same bits,  
17496 but checks for having at least all of the matching bits set instead of having exactly the matching  
17497 bits set.

17498 The exact format of the interactive prompts is unspecified. Only the general nature of the  
17499 contents of prompts are specified because:

- 17500 • Implementations may desire more descriptive prompts than those used on historical  
17501 implementations.
- 17502 • Since the historical prompt strings do not terminate with <newline>s, there is no portable  
17503 way for another program to interact with the prompts of this utility via pipes.

17504 Therefore, an application using this prompting option relies on the system to provide the most  
17505 suitable dialog directly with the user, based on the general guidelines specified.

17506 The **-name file** operand was changed to use the shell pattern matching notation so that *find* is  
17507 consistent with other utilities using pattern matching.

17508 The **-size** operand refers to the size of a file, rather than the number of blocks it may occupy in  
17509 the file system. The intent is that the *st\_size* field defined in the System Interfaces volume of  
17510 IEEE Std. 1003.1-200x should be used, not the *st\_blocks* found in historical implementations.  
17511 There are at least two reasons for this:

- 17512 1. In both System V and BSD, *find* only uses *st\_size* in size calculations for the operands  
17513 specified by this volume of IEEE Std. 1003.1-200x. (BSD uses *st\_blocks* only when  
17514 processing the **-ls** primary.)



17515 2. Users usually think of file size in terms of bytes, which is also the unit used by the *ls* utility  
 17516 for the output from the `-l` option. (In both System V and BSD, *ls* uses *st\_size* for the `-l`  
 17517 option size field and uses *st\_blocks* for the *ls -s* calculations. This volume of  
 17518 IEEE Std. 1003.1-200x does not specify *ls -s*.)

17519 The descriptions of `-atime`, `-ctime`, and `-mtime` were changed from the SVID description of *n*  
 17520 “days” to “24-hour periods”. The description is also different in terms of the exact timeframe for  
 17521 the *n* case (*versus* the *+n* or *-n*), but it matches all known historical implementations. It refers to  
 17522 one 86 400 second period in the past, not any time from the beginning of that period to the  
 17523 current time. For example, `-atime 3` is true if the file was accessed any time in the period from 72  
 17524 hours to 48 hours ago.

17525 Historical implementations do not modify “{ }” when it appears as a substring of an `-exec` or  
 17526 `-ok utility_name` or argument string. There have been numerous user requests for this extension,  
 17527 so this volume of IEEE Std. 1003.1-200x allows the desired behavior. At least one recent  
 17528 implementation does support this feature, but encountered several problems in managing  
 17529 memory allocation and dealing with multiple occurrences of “{ }” in a string while it was being  
 17530 developed, so it is not yet required behavior.

17531 Assuming the presence of `-print` was added to correct a historical pitfall that plagues novice  
 17532 users, it is entirely upward-compatible from the historical System V *find* utility. In its simplest  
 17533 form (*find directory*), it could be confused with the historical BSD fast *find*. The BSD developers  
 17534 agreed that adding `-print` as a default expression was the correct decision and have added the  
 17535 fast *find* functionality within a new utility called *locate*.

17536 Historically, the `-L` option was implemented using the primary `-follow`. The `-H` and `-L` options  
 17537 were added for two reasons. First, they offer a finer granularity of control and consistency with  
 17538 other programs that walk file hierarchies. Second, the `-follow` primary always evaluated to true.  
 17539 As they were historically really global variables that took effect before the traversal began, some  
 17540 valid expressions had unexpected results. An example is the expression `-print -o -follow`.  
 17541 Because `-print` always evaluates to true, the standard order of evaluation implies that `-follow`  
 17542 would never be evaluated. This was never the case. Historical practice for the `-follow` primary,  
 17543 however, is not consistent. Some implementations always follow symbolic links on the  
 17544 command line whether `-follow` is specified or not. Others follow symbolic links on the  
 17545 command line only if `-follow` is specified. Both behaviors are provided by the `-H` and `-L`  
 17546 options, but scripts using the current `-follow` primary would be broken if the `-follow` option is  
 17547 specified to work either way.

17548 Since the `-L` option resolves all symbolic links and the `-type l` primary is true for symbolic links  
 17549 that still exist after symbolic links have been resolved, the command:

```
17550 find -L . -type l
```

17551 prints a list of symbolic links reachable from the current directory that do not resolve to  
 17552 accessible files.

#### 17553 FUTURE DIRECTIONS

17554 None.

#### 17555 SEE ALSO

17556 *chmod*, *pax*, *sh*, *test*, the System Interfaces volume of IEEE Std. 1003.1-200x, *stat()*

#### 17557 CHANGE HISTORY

17558 First released in Issue 2.

17559 **Issue 4**

17560 Aligned with the ISO/IEC 9945-2: 1993 standard.

17561 **Issue 5**

17562 FUTURE DIRECTIONS section added.

17563 **Issue 6**

17564 The following new requirements on POSIX implementations derive from alignment with the  
17565 Single UNIX Specification:

17566 • The `-perm [-]onum` primary is supported.

17567 The `find` utility is aligned with the IEEE P1003.2b draft standard, to include processing of  
17568 symbolic links and changes to the description of the `atime`, `ctime`, and `mtime` operands.

17569 **NAME**

17570 fold — filter for folding lines

17571 **SYNOPSIS**17572 fold [-bs][-w *width*][*file...*]17573 **DESCRIPTION**

17574 The *fold* utility is a filter that shall fold lines from its input files, breaking the lines to have a  
 17575 maximum of *width* column positions (or bytes, if the **-b** option is specified). Lines shall be  
 17576 broken by the insertion of a <newline> character such that each output line (referred to later in  
 17577 this section as a *segment*) is the maximum width possible that does not exceed the specified  
 17578 number of column positions (or bytes). A line shall not be broken in the middle of a character.  
 17579 The behavior is undefined if *width* is less than the number of columns any single character in the  
 17580 input would occupy.

17581 If the <carriage-return>, <backspace>, or <tab> characters are encountered in the input, and the  
 17582 **-b** option is not specified, they shall be treated specially:

17583 <backspace> The current count of line width shall be decremented by one, although the count  
 17584 never shall become negative. The *fold* utility shall not insert a <newline> character  
 17585 immediately before or after any <backspace> character.

17586 <carriage-return>  
 17587 The current count of line width shall be set to zero. The *fold* utility shall not insert a  
 17588 <newline> character immediately before or after any <carriage-return> character.

17589 <tab> Each <tab> character encountered shall advance the column position pointer to the  
 17590 next tab stop. Tab stops shall be at each column position *n* such that *n* modulo 8  
 17591 equals 1.

17592 **OPTIONS**

17593 The *fold* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 17594 12.2, Utility Syntax Guidelines.

17595 The following options shall be supported:

17596 **-b** Count *width* in bytes rather than column positions.

17597 **-s** If a segment of a line contains a <blank> character within the first *width* column  
 17598 positions (or bytes), break the line after the last such <blank> character meeting the  
 17599 width constraints. If there is no <blank> character meeting the requirements, the **-s**  
 17600 option shall have no effect for that output segment of the input line.

17601 **-w *width*** Specify the maximum line length, in column positions (or bytes if **-b** is specified).  
 17602 The results are unspecified if *width* is not a positive decimal number. The default  
 17603 value shall be 80.

17604 **OPERANDS**

17605 The following operand shall be supported:

17606 *file* A path name of a text file to be folded. If no *file* operands are specified, the  
 17607 standard input shall be used.

17608 **STDIN**

17609 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES  
 17610 section.

17611 **INPUT FILES**

17612           If the **-b** option is specified, the input files shall be text files except that the lines are not limited  
17613           to {**LINE\_MAX**} bytes in length. If the **-b** option is not specified, the input files shall be text files.

17614 **ENVIRONMENT VARIABLES**

17615           The following environment variables shall affect the execution of *fold*:

17616           **LANG**       Provide a default value for the internationalization variables that are unset or null.  
17617                       If *LANG* is unset or null, the corresponding value from the implementation-  
17618                       defined default locale shall be used. If any of the internationalization variables  
17619                       contains an invalid setting, the utility shall behave as if none of the variables had  
17620                       been defined.

17621           **LC\_ALL**     If set to a non-empty string value, override the values of all the other  
17622                       internationalization variables.

17623           **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
17624                       characters (for example, single-byte as opposed to multi-byte characters in  
17625                       arguments and input files), and for the determination of the width in column  
17626                       positions each character would occupy on a constant-width font output device.

17627           **LC\_MESSAGES**

17628                       Determine the locale that should be used to affect the format and contents of  
17629                       diagnostic messages written to standard error.

17630 XSI           **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

17631 **ASYNCHRONOUS EVENTS**

17632           Default.

17633 **STDOUT**

17634           The standard output shall be a file containing a sequence of characters whose order shall be  
17635           preserved from the input files, possibly with inserted <newline> characters.

17636 **STDERR**

17637           Used only for diagnostic messages.

17638 **OUTPUT FILES**

17639           None.

17640 **EXTENDED DESCRIPTION**

17641           None.

17642 **EXIT STATUS**

17643           The following exit values shall be returned:

17644           0   All input files were processed successfully.

17645           >0 An error occurred.

17646 **CONSEQUENCES OF ERRORS**

17647           Default.

17648 **APPLICATION USAGE**

17649 The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths. The  
17650 *cut* utility should be used when the number of lines (or records) needs to remain constant. The  
17651 *fold* utility should be used when the contents of long lines need to be kept contiguous.

17652 The *fold* utility is frequently used to send text files to printers that truncate, rather than fold, lines  
17653 wider than the printer is able to print (usually 80 or 132 column positions).

17654 **EXAMPLES**

17655 An example invocation that submits a file of possibly long lines to the printer (under the  
17656 assumption that the user knows the line width of the printer to be assigned by *lp*):

```
17657 fold -w 132 bigfile | lp
```

17658 **RATIONALE**

17659 Although terminal input in canonical processing mode requires the erase character (frequently  
17660 set to <backspace>) to erase the previous character (not byte or column position), terminal  
17661 output is not buffered and is extremely difficult, if not impossible, to parse correctly; the  
17662 interpretation depends entirely on the physical device that actually displays/prints/stores the  
17663 output. In all known internationalized implementations, the utilities producing output for mixed  
17664 column-width output assume that a <backspace> backs up one column position and outputs  
17665 enough <backspace>s to return to the start of the character when <backspace> is used to  
17666 provide local line motions to support underlining and emboldening operations. Since *fold*  
17667 without the **-b** option is dealing with these same constraints, <backspace> is always treated as  
17668 backing up one column position rather than backing up one character.

17669 Historical versions of the *fold* utility assumed 1 byte was one character and occupied one column  
17670 position when written out. This is no longer always true. Since the most common usage of *fold* is  
17671 believed to be folding long lines for output to limited-length output devices, this capability was  
17672 preserved as the default case. The **-b** option was added so that applications could *fold* files with  
17673 arbitrary length lines into text files that could then be processed by the standard utilities. Note  
17674 that although the width for the **-b** option is in bytes, a line is never split in the middle of a  
17675 character. (It is unspecified what happens if a width is specified that is too small to hold a single  
17676 character found in the input followed by a <newline>.)

17677 The tab stops are hardcoded to be every eighth column to meet historical practice. No new  
17678 method of specifying other tab stops was invented.

17679 **FUTURE DIRECTIONS**

17680 None.

17681 **SEE ALSO**

17682 *cut*

17683 **CHANGE HISTORY**

17684 First released in Issue 4.

17685 **Issue 6**

17686 The normative text is reworded to avoid use of the term “must” for application requirements.

## 17687 NAME

17688 fort77 — FORTRAN compiler (**FORTRAN**)

## 17689 SYNOPSIS

```
17690 FD fort77 [-c][-g][-L directory]... [-O optlevel][-o outfile][-s][-w]
17691 operand...
```

17692

## 17693 DESCRIPTION

17694 The *fort77* utility is the interface to the FORTRAN compilation system; it shall accept the full  
 17695 FORTRAN-77 language defined by the ANSI X3.9-1978 standard. The system conceptually  
 17696 consists of a compiler and link editor. The files referenced by *operands* are compiled and linked  
 17697 to produce an executable file. It is unspecified whether the linking occurs entirely within the  
 17698 operation of *fort77*; some systems may produce objects that are not fully resolved until the file is  
 17699 executed.

17700 If the **-c** option is present, for all path name operands of the form *file.f*, the files:

17701 \$(basename *pathname.f*).o

17702 shall be created or overwritten as the result of successful compilation. If the **-c** option is not  
 17703 specified, it is unspecified whether such *.o* files are created or deleted for the *file.f* operands.

17704 If there are no options that prevent link editing (such as **-c**) and all operands compile and link  
 17705 without error, the resulting executable file shall be written into the file named by the **-o** option  
 17706 (if present) or to the file **a.out**. The executable file shall be created as specified in the System  
 17707 Interfaces volume of IEEE Std. 1003.1-200x, except that the file permissions shall be set to:

17708 S\_IRWXO | S\_IRWXG | S\_IRWXU

17709 and that the bits specified by the *umask* of the process shall be cleared.

## 17710 OPTIONS

17711 The *fort77* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 17712 12.2, Utility Syntax Guidelines, except that:

- 17713 • The **-l** *library* operands have the format of options, but their position within a list of  
 17714 operands affects the order in which libraries are searched.
- 17715 • The order of specifying the multiple **-L** options is significant.
- 17716 • Portable applications shall specify each option separately; that is, grouping option letters (for  
 17717 example, **-cg**) need not be recognized by all implementations.

17718 The following options shall be supported:

17719 **-c** Suppress the link-edit phase of the compilation, and do not remove any object files  
 17720 that are produced.

17721 **-g** Produce symbolic information in the object or executable files; the nature of this  
 17722 information is unspecified, and may be modified by implementation-defined  
 17723 interactions with other options.

17724 **-s** Produce object or executable files, or both, from which symbolic and other  
 17725 information not required for proper execution using the *exec* family of functions  
 17726 defined in the System Interfaces volume of IEEE Std. 1003.1-200x has been  
 17727 removed (stripped). If both **-g** and **-s** options are present, the action taken is  
 17728 unspecified.

17729 **-o** *outfile* Use the path name *outfile*, instead of the default **a.out**, for the executable file  
 17730 produced. If the **-o** option is present with **-c**, the result is unspecified.

17731        **-L *directory***   Change the algorithm of searching for the libraries named in **-I** operands to look in  
 17732                    the directory named by the *directory* path name before looking in the usual places.  
 17733                    Directories named in **-L** options shall be searched in the specified order. At least  
 17734                    ten instances of this option shall be supported in a single *fort77* command  
 17735                    invocation. If a directory specified by a **-L** option contains a file named **libf.a**, the  
 17736                    results are unspecified.

17737        **-O *optlevel***   Specify the level of code optimization. If the *optlevel* option-argument is the digit  
 17738                    '0', all special code optimizations shall be disabled. If it is the digit '1', the  
 17739                    nature of the optimization is unspecified. If the **-O** option is omitted, the nature of  
 17740                    the system's default optimization is unspecified. It is unspecified whether code  
 17741                    generated in the presence of the **-O 0** option is the same as that generated when  
 17742                    **-O** is omitted. Other *optlevel* values may be supported.

17743        **-w**                Suppress warnings.

17744        Multiple instances of **-L** options can be specified.

#### 17745 OPERANDS

17746        An *operand* is either in the form of a path name or the form **-I *library***. At least one operand of the  
 17747        path name form shall be specified. The following operands shall be supported:

17748        ***file.f***            The path name of a FORTRAN source file to be compiled and optionally passed to  
 17749                    the link editor. The file name operand shall be of this form if the **-c** option is used.

17750        ***file.a***            A library of object files typically produced by *ar*, and passed directly to the link  
 17751                    editor. Implementations may recognize implementation-defined suffixes other  
 17752                    than **.a** as denoting object file libraries.

17753        ***file.o***            An object file produced by *fort77 -c* and passed directly to the link editor.  
 17754                    Implementations may recognize implementation-defined suffixes other than **.o** as  
 17755                    denoting object files.

17756        The processing of other files is implementation-defined.

17757        **-I *library***       (The letter ell.) Search the library named:

17758                            *liblibrary.a*

17759                    A library is searched when its name is encountered, so the placement of a **-I**  
 17760                    operand is significant. Several standard libraries can be specified in this manner, as  
 17761                    described in the EXTENDED DESCRIPTION section. Implementations may  
 17762                    recognize implementation-defined suffixes other than **.a** as denoting libraries.

#### 17763 STDIN

17764        Not used.

#### 17765 INPUT FILES

17766        The input file shall be one of the following: a text file containing FORTRAN source code; an  
 17767        object file in the format produced by *fort77 -c*; or a library of object files, in the format produced  
 17768        by archiving zero or more object files, using *ar*. Implementations may supply additional utilities  
 17769        that produce files in these formats. Additional input files are implementation-defined.

17770        A **<tab>** character encountered within the first six characters on a line of source code shall cause  
 17771        the compiler to interpret the following character as if it were the seventh character on the line  
 17772        (that is, in column 7).

## 17773 ENVIRONMENT VARIABLES

17774 The following environment variables shall affect the execution of *fort77*:

17775 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 17776 If *LANG* is unset or null, the corresponding value from the implementation-  
 17777 defined default locale shall be used. If any of the internationalization variables  
 17778 contains an invalid setting, the utility shall behave as if none of the variables had  
 17779 been defined.

17780 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 17781 internationalization variables.

17782 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 17783 characters (for example, single-byte as opposed to multi-byte characters in  
 17784 arguments and input files).

17785 **LC\_MESSAGES**

17786 Determine the locale that should be used to affect the format and contents of  
 17787 diagnostic messages written to standard error.

17788 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

17789 **TMPDIR** Determine the path name that should override the default directory for temporary  
 17790 files, if any.

## 17791 ASYNCHRONOUS EVENTS

17792 Default.

17793 **STDOUT**

17794 Not used.

17795 **STDERR**

17796 Used only for diagnostic messages. If more than one *file* operand ending in *.f* (or possibly other  
 17797 unspecified suffixes) is given, for each such file:

17798 "%s:\n", <*file*>

17799 may be written to allow identification of the diagnostic message with the appropriate input file.

17800 This utility may produce warning messages about certain conditions that do not warrant  
 17801 returning an error (non-zero) exit value.

17802 **OUTPUT FILES**

17803 Object files, listing files and executable files shall be produced in unspecified formats.

17804 **EXTENDED DESCRIPTION**17805 **Standard Libraries**

17806 The *fort77* utility shall recognize the following **-l** operand for the standard library:

17807 **-l f** This library contains all library functions referenced in the ANSI X3.9-1978  
 17808 standard. This operand shall not be required to be present to cause a search of this  
 17809 library.

17810 In the absence of options that inhibit invocation of the link editor, such as **-c**, the *fort77* utility  
 17811 shall cause the equivalent of a **-l f** operand to be passed to the link editor as the last **-l** operand,  
 17812 causing it to be searched after all other object files and libraries are loaded.

17813 It is unspecified whether the library **libf.a** exists as a regular file. The implementation may  
 17814 accept as **-l** operands names of objects that do not exist as regular files.



17815 **External Symbols**

17816 The FORTRAN compiler and link editor shall support the significance of external symbols up to  
17817 a length of at least 31 bytes; case folding is permitted. The action taken upon encountering  
17818 symbols exceeding the implementation-defined maximum symbol length is unspecified.

17819 The compiler and link editor shall support a minimum of 511 external symbols per source or  
17820 object file, and a minimum of 4095 external symbols total. A diagnostic message is written to  
17821 standard output if the implementation-defined limit is exceeded; other actions are unspecified.

17822 **EXIT STATUS**

17823 The following exit values shall be returned:

17824 0 Successful compilation or link edit.

17825 >0 An error occurred.

17826 **CONSEQUENCES OF ERRORS**

17827 When *fort77* encounters a compilation error, it shall write a diagnostic to standard error and  
17828 continue to compile other source code operands. It shall return a non-zero exit status, but it is  
17829 implementation-defined whether an object module is created. If the link edit is unsuccessful, a  
17830 diagnostic message shall be written to standard error, and *fort77* shall exit with a non-zero  
17831 status.

17832 **APPLICATION USAGE**

17833 None.

17834 **EXAMPLES**

17835 The following usage example compiles *xyz.f* and creates the executable file *foo*:

17836 `fort77 -o foo xyz.f`

17837 The following example compiles *xyz.f* and creates the object file *xyz.o*:

17838 `fort77 -c xyz.f`

17839 The following example compiles *xyz.f* and creates the executable file *a.out*:

17840 `fort77 xyz.f`

17841 The following example compiles *xyz.f*, links it with *b.o*, and creates the executable *a.out*:

17842 `fort77 xyz.f b.o`

17843 **RATIONALE**

17844 The name of this utility was chosen as *fort77* to parallel the renaming of the C compiler. The  
17845 name *f77* was not chosen to avoid problems with historical implementations. The  
17846 ANSI X3.9-1978 standard was selected as a normative reference because the ISO/IEC version of  
17847 FORTRAN-77 has been superseded by the ISO/IEC 1539: 1990 standard (Fortran-90).

17848 The file inclusion and symbol definition **#define** mechanisms used by the *c99* utility were not  
17849 included in this volume of IEEE Std. 1003.1-200x—even though they are commonly  
17850 implemented—since there is no requirement that the FORTRAN compiler use the C  
17851 preprocessor.

17852 The **-onetrip** option was not included in this volume of IEEE Std. 1003.1-200x, even though  
17853 many historical compilers support it, because it is derived from FORTRAN-66; it is an  
17854 anachronism that should not be perpetuated.

17855 Some implementations produce compilation listings. This aspect of FORTRAN has been left  
17856 unspecified because there was controversy concerning the various methods proposed for  
17857 implementing it: a **-V** option overlapped with historical vendor practice and a naming

17858 convention of creating files with `.l` suffixes collided with historical *lex* file naming practice.

17859 There is no `-I` option in this version of this volume of IEEE Std. 1003.1-200x to specify a directory  
17860 for file inclusion. An `INCLUDE` directive has been a part of the Fortran-90 discussions, but an  
17861 interface supporting that standard is not in the current scope.

17862 It is noted that many FORTRAN compilers produce an object module even when compilation  
17863 errors occur; during a subsequent compilation, the compiler may patch the object module rather  
17864 than recompiling all the code. Consequently, it is left to the implementor whether or not an  
17865 object file is created.

17866 A reference to MIL-STD-1753 was removed from an early proposal in response to a request from  
17867 the POSIX FORTRAN-binding standard developers. It was not the intention of the standard  
17868 developers to require certification of the FORTRAN compiler, and IEEE Std. 1003.9-1992 does  
17869 not specify the military standard or any special preprocessing requirements. Furthermore, use of  
17870 that document would have been inappropriate for an international standard.

17871 The specification of optimization has been subject to changes through early proposals. At one  
17872 time, `-O` and `-N` were Booleans: optimize and do not optimize (with an unspecified default).  
17873 Some historical practice lead this to be changed to:

17874 `-O 0` No optimization.

17875 `-O 1` Some level of optimization.

17876 `-O n` Other, unspecified levels of optimization.

17877 It is not always clear whether “good code generation” is the same thing as optimization. Simple  
17878 optimizations of local actions do not usually affect the semantics of a program. The `-O 0` option  
17879 has been included to accommodate the very particular nature of scientific calculations in a  
17880 highly optimized environment; compilers make errors. Some degree of optimization is expected,  
17881 even if it is not documented here, and the ability to shut it off completely could be important  
17882 when porting an application. An implementation may treat `-O 0` as “do less than normal” if it  
17883 wishes, but this is only meaningful if any of the operations it performs can affect the semantics  
17884 of a program. It is highly dependent on the implementation whether doing less than normal is  
17885 logical. It is not the intent of the `-O 0` option to ask for inefficient code generation, but rather to  
17886 assure that any semantically visible optimization is suppressed.

17887 The specification of standard library access is consistent with the C compiler specification.  
17888 Implementations are not required to have `/usr/lib/libf.a`, as many historical implementations do,  
17889 but if not they are required to recognize `f` as a token.

17890 External symbol size limits are in normative text; portable applications need to know these  
17891 limits. However, the minimum maximum symbol length should be taken as a constraint on a  
17892 portable application, not on an implementation, and consequently the action taken for a symbol  
17893 exceeding the limit is unspecified. The minimum size for the external symbol table was added  
17894 for similar reasons.

17895 The CONSEQUENCES OF ERRORS section clearly specifies the behavior of the compiler when  
17896 compilation or link-edit errors occur. The behavior of several historical implementations was  
17897 examined, and the choice was made to be silent on the status of the executable, or `a.out`, file in  
17898 the face of compiler or linker errors. If a linker writes the executable file, then links it on disk  
17899 with `lseek()`s and `write()`s, the partially linked executable file can be left on disk and its execute  
17900 bits turned off if the link edit fails. However, if the linker links the image in memory before  
17901 writing the file to disk, it need not touch the executable file (if it already exists) because the link  
17902 edit fails. Since both approaches are historical practice, a portable application shall rely on the  
17903 exit status of *fort77*, rather than on the existence or mode of the executable file.

- 17904 The `-g` and `-s` options are not specified as mutually-exclusive. Historically these two options  
17905 have been mutually-exclusive, but because both are so loosely specified, it seemed appropriate  
17906 to leave their interaction unspecified.
- 17907 The requirement that portable applications specify compiler options separately is to reserve the  
17908 multi-character option name space for vendor-specific compiler options, which are known to  
17909 exist in many historical implementations. Implementations are not required to recognize, for  
17910 example, `-gc` as if it were `-g -c`; nor are they forbidden from doing so. The SYNOPSIS shows all  
17911 of the options separately to highlight this requirement on applications.
- 17912 Echoing file names to standard error is considered a diagnostic message because it would  
17913 otherwise be difficult to associate an error message with the erring file. They are described with  
17914 “may” to allow implementations to use other methods of identifying files and to parallel the  
17915 description in *c99*.
- 17916 **FUTURE DIRECTIONS**
- 17917 A compilation system based on the ISO/IEC 1539:1990 standard (Fortran-90) may be considered  
17918 for a future issue; it may have a different utility name from *fort77*.
- 17919 **SEE ALSO**
- 17920 *ar, asa, c99, umask*
- 17921 **CHANGE HISTORY**
- 17922 First released in Issue 4.
- 17923 **Issue 6**
- 17924 This utility is now marked as part of the FORTRAN Development Utilities option.
- 17925 The normative text is reworded to avoid use of the term “must” for application requirements.

17926 **NAME**

17927 fuser — list process IDs of all processes that have one or more files open

17928 **SYNOPSIS**

17929 xSI fuser [ -cfu ] file ...

17930

17931 **DESCRIPTION**

17932 The *fuser* utility shall write to standard output the process IDs of processes running on the local  
 17933 system that have one or more named files open. For block special devices, all processes using  
 17934 any file on that device are listed.

17935 The *fuser* utility shall write to standard error additional information about the named files  
 17936 indicating how the file is being used.

17937 Any output for processes running on remote systems that have a named file open is unspecified.

17938 A user may need appropriate privilege to invoke the *fuser* utility.

17939 **OPTIONS**

17940 The *fuser* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 17941 12.2, Utility Syntax Guidelines.

17942 The following options shall be supported:

17943 **-c** The file is treated as a mount point and the utility shall report on any files open in  
 17944 the file system.

17945 **-f** The report shall be only for the named files.

17946 **-u** The user name, in parentheses, associated with each process ID written to standard  
 17947 output shall be written to standard error.

17948 **OPERANDS**

17949 The following operand shall be supported:

17950 *file* A path name on which the file or file system is to be reported.

17951 **STDIN**

17952 Not used.

17953 **INPUT FILES**

17954 The user database.

17955 **ENVIRONMENT VARIABLES**

17956 The following environment variables shall affect the execution of *fuser*:

17957 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 17958 If *LANG* is unset or null, the corresponding value from the implementation-  
 17959 defined default locale shall be used. If any of the internationalization variables  
 17960 contain an invalid setting, the utility behaves as if none of the variables had been  
 17961 set.

17962 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 17963 internationalization variables.

17964 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 17965 characters (for example, single-byte as opposed to multi-byte characters in  
 17966 arguments).

17967 **LC\_MESSAGES**

17968 Determine the locale that should be used to affect the format and contents of

- 17969 diagnostic messages written to standard error.
- 17970 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 17971 **ASYNCHRONOUS EVENTS**
- 17972 Default.
- 17973 **STDOUT**
- 17974 The *fuser* utility shall write the process ID for each process using each file given as an operand to standard output in the following format:
- 17975
- 17976 "%d", <*process\_id*>
- 17977 **STDERR**
- 17978 The *fuser* utility shall write diagnostic messages to standard error.
- 17979 The *fuser* utility also shall write the following to standard error:
- 17980
- The path name of each named file is written followed immediately by a colon.
  - For each process ID written to standard output, the character 'c' shall be written to standard error if the process is using the file as its current directory and the character 'r' shall be written to standard error if the process is using the file as its root directory. Implementations may write other alphabetic characters to indicate other uses of files.
  - When the **-u** option is specified, characters indicating the use of the file shall be followed immediately by the user name, in parentheses, corresponding to the process' real user ID. If the user name cannot be resolved from the process' real user ID, the process' real user ID shall be written instead of the user name.
- 17989 When standard output and standard error are directed to the same file, the output shall be interspersed so that the file name appears at the start of each line, followed by the process ID and characters indicating the use of the file. Then, if the **-u** option is specified, the user name or user ID for each process using that file shall be written.
- 17990
- 17991
- 17992
- 17993 A <newline> character shall be written to standard error after the last output described above
- 17994 for each *file* operand.
- 17995 **OUTPUT FILES**
- 17996 None.
- 17997 **EXTENDED DESCRIPTION**
- 17998 None.
- 17999 **EXIT STATUS**
- 18000 The following exit values shall be returned:
- 18001 0 Successful completion.
- 18002 >0 An error occurred.
- 18003 **CONSEQUENCES OF ERRORS**
- 18004 Default.

18005 **APPLICATION USAGE**

18006           None.

18007 **EXAMPLES**

18008           The command:

18009           fuser -fu .

18010           writes to standard output the process IDs of processes that are using the current directory and

18011           writes to standard error an indication of how those processes are using the directory and the

18012           user names associated with the processes that are using the current directory.

18013 **RATIONALE**

18014           None.

18015 **FUTURE DIRECTIONS**

18016           None.

18017 **SEE ALSO**

18018           None.

18019 **CHANGE HISTORY**

18020           First released in Issue 5.

18021 **NAME**

18022 gencat — generate a formatted message catalog

18023 **SYNOPSIS**18024 XSI `gencat catfile msgfile...`

18025

18026 **DESCRIPTION**

18027 The *gencat* utility shall merge the message text source files *msgfile* into a formatted message  
 18028 catalog *catfile*. The file *catfile* shall be created if it does not already exist. If *catfile* does exist, its  
 18029 messages shall be included in the new *catfile*. If set and message numbers collide, the new  
 18030 message text defined in *msgfile* shall replace the old message text currently contained in *catfile*.

18031 **OPTIONS**

18032 None.

18033 **OPERANDS**

18034 The following operands shall be supported:

18035 *catfile* A path name of the formatted message catalog. If '-' is specified, standard output  
 18036 shall be used. The format of the message catalog produced is unspecified.

18037 *msgfile* A path name of a message text source file. If '-' is specified for an instance of  
 18038 *msgfile*, standard input shall be used. The format of message text source files is  
 18039 defined in the EXTENDED DESCRIPTION section.

18040 **STDIN**18041 The standard input shall not be used unless a *msgfile* operand is specified as '-'.18042 **INPUT FILES**

18043 The input files shall be text files.

18044 **ENVIRONMENT VARIABLES**18045 The following environment variables shall affect the execution of *gencat*:

18046 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 18047 If *LANG* is unset or null, the corresponding value from the implementation-  
 18048 defined default locale shall be used. If any of the internationalization variables  
 18049 contains an invalid setting, the utility shall behave as if none of the variables had  
 18050 been defined.

18051 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 18052 internationalization variables.

18053 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 18054 characters (for example, single-byte as opposed to multi-byte characters in  
 18055 arguments and input files).

18056 *LC\_MESSAGES*

18057 Determine the locale that should be used to affect the format and contents of  
 18058 diagnostic messages written to standard error.

18059 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

18060 **ASYNCHRONOUS EVENTS**

18061 Default.

18062 **STDOUT**

18063 The standard output shall not be used unless the *catfile* operand is specified as '- '.

18064 **STDERR**

18065 Used only for diagnostic messages.

18066 **OUTPUT FILES**

18067 None.

18068 **EXTENDED DESCRIPTION**

18069 The application shall ensure that the format of a message text source file is defined as follows.

18070 Note that the fields of a message text source line are separated by a single <blank> character.

18071 Any other <blank> characters are considered as being part of the subsequent field.

18072 **\$set *n* comment**

18073 This line specifies the set identifier of the following messages until the next **\$set** or  
 18074 end-of-file appears. The *n* denotes the set identifier, which is defined as a number  
 18075 in the range [1, {NL\_SETMAX}] (see the <limits.h> header defined in the System  
 18076 Interfaces volume of IEEE Std. 1003.1-200x). The application shall ensure that set  
 18077 identifiers are presented in ascending order within a single source file, but need  
 18078 not be contiguous. Any string following the set identifier shall be treated as a  
 18079 comment. If no **\$set** directive is specified in a message text source file, all messages  
 18080 shall be located in an implementation-defined default message set NL\_SETD (see  
 18081 the <nl\_types.h> header defined in the System Interfaces volume of  
 18082 IEEE Std. 1003.1-200x).

18083 **\$delset *n* comment**

18084 This line deletes message set *n* from an existing message catalog. The *n* denotes the  
 18085 set number [1, {NL\_SETMAX}]. Any string following the set number shall be  
 18086 treated as a comment.

18087 **\$ comment** A line beginning with '\$ ' followed by a <blank> character shall be treated as a  
 18088 comment.

18089 ***m* message-text**

18090 The *m* denotes the message identifier, which is defined as a number in the range [1,  
 18091 {NL\_MSGMAX}] (see the <limits.h> header defined in the System Interfaces  
 18092 volume of IEEE Std. 1003.1-200x). The *message-text* shall be stored in the message  
 18093 catalog with the set identifier specified by the last **\$set** directive, and with message  
 18094 identifier *m*. If the *message-text* is empty, and a <blank> character field separator is  
 18095 present, an empty string shall be stored in the message catalog. If a message source  
 18096 line has a message number, but neither a field separator nor *message-text*, the  
 18097 existing message with that number (if any) shall be deleted from the catalog. The  
 18098 application shall ensure that message identifiers are in ascending order within a  
 18099 single set, but need not be contiguous. The application shall ensure that the length  
 18100 of *message-text* is in the range [0, {NL\_TEXTMAX}] (see the <limits.h> header  
 18101 defined in the System Interfaces volume of IEEE Std. 1003.1-200x).

18102 **\$quote *n*** This line specifies an optional quote character *c*, which can be used to surround  
 18103 *message-text* so that trailing spaces or null (empty) messages are visible in a  
 18104 message source line. By default, or if an empty **\$quote** directive is supplied, no  
 18105 quoting of *message-text* shall be recognized.

18106 Empty lines in a message text source file shall be ignored. The effects of lines starting with any  
 18107 character other than those defined above are implementation-defined.



18108 Text strings can contain the special characters and escape sequences defined in the following  
18109 table:

18110  
18111  
18112  
18113  
18114  
18115  
18116  
18117  
18118  
18119

| Description       | Symbol | Sequence |
|-------------------|--------|----------|
| <newline>         | NL(LF) | \n       |
| Horizontal tab    | HT     | \t       |
| <vertical-tab>    | VT     | \v       |
| <backspace>       | BS     | \b       |
| <carriage-return> | CR     | \r       |
| <form-feed>       | FF     | \f       |
| Backslash         | \      | \\       |
| Bit pattern       | ddd    | \ddd     |

18120 The escape sequence "\ddd" consists of backslash followed by one, two, or three octal digits,  
18121 which shall be taken to specify the value of the desired character. If the character following a  
18122 backslash is not one of those specified, the backslash shall be ignored.

18123 Backslash ('\') followed by a <newline> character is also used to continue a string on the  
18124 following line. Thus, the following two lines describe a single message string:

18125 1 This line continues \  
18126 to the next line

18127 which is equivalent to:

18128 1 This line continues to the next line

#### 18129 EXIT STATUS

18130 The following exit values shall be returned:

18131 0 Successful completion.

18132 >0 An error occurred.

#### 18133 CONSEQUENCES OF ERRORS

18134 Default.

#### 18135 APPLICATION USAGE

18136 Message catalogs produced by *gencat* are binary encoded, meaning that their portability cannot  
18137 be guaranteed between different types of machine. Thus, just as C programs need to be  
18138 recompiled for each type of machine, so message catalogs must be recreated via *gencat*.

#### 18139 EXAMPLES

18140 None.

#### 18141 RATIONALE

18142 None.

#### 18143 FUTURE DIRECTIONS

18144 None.

#### 18145 SEE ALSO

18146 *iconv*, the System Interfaces volume of IEEE Std. 1003.1-200x, <**limits.h**>

#### 18147 CHANGE HISTORY

18148 First released in Issue 3.

18149 **Issue 4**

18150           Format reorganized.

18151           Internationalized environment variable support mandated.

18152 **Issue 6**

18153           The normative text is reworded to avoid use of the term “must” for application requirements.

## 18154 NAME

18155 `get` — get a version of an SCCS file (**DEVELOPMENT**)

## 18156 SYNOPSIS

```
18157 xSI get [-begkmlPst][-c cutoff][-i list][-r SID][-x list] file...
```

18158

## 18159 DESCRIPTION

18160 The `get` utility shall generate a text file from each named SCCS *file* according to the specifications  
18161 given by its options.

18162 The generated text is normally written into a file called the **g-file** whose name is derived from  
18163 the SCCS file name by simply removing the leading "s . ".

## 18164 OPTIONS

18165 The `get` utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
18166 12.2, Utility Syntax Guidelines.

18167 The following options shall be supported:

18168 **-r *SID*** Indicate the SCCS Identification String (*SID*) of the version (delta) of an SCCS file  
18169 to be retrieved. The table shows, for the most useful cases, what version of an  
18170 SCCS file is retrieved (as well as the *SID* of the version to be eventually created by  
18171 *delta* if the **-e** option is also used), as a function of the *SID* specified.

18172 **-c *cutoff*** Indicate the *cutoff* date-time, in the form:

```
18173 YY[MM[DD[HH[MM[SS]]]]]
```

18174 For the *YY* component, values in the range [69-99] shall refer to years in the  
18175 twentieth century (1969 to 1999 inclusive); values in the range [00-68] shall refer to  
18176 years in the twenty-first century (2000 to 2068 inclusive).

18177 No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
18178 date-time are included in the generated text file. Units omitted from the date-time  
18179 default to their maximum possible values; for example, **-c 7502** is equivalent to **-c**  
18180 **750228235959**.

18181 Any number of non-numeric characters may separate the various 2-digit pieces of  
18182 the *cutoff* date-time. This feature allows the user to specify a *cutoff* date in the form:  
18183 **-c "77/2/2 9:22:25"**.

18184 **-e** Indicate that the `get` is for the purpose of editing or making a change (delta) to the  
18185 SCCS file via a subsequent use of *delta*. The **-e** option used in a `get` for a particular  
18186 version (*SID*) of the SCCS file shall prevent further `get` commands from editing on  
18187 the same *SID* until *delta* is executed or the **j** (joint edit) flag is set in the SCCS file.  
18188 Concurrent use of `get -e` for different *SIDs* is always allowed.

18189 If the **g-file** generated by `get` with a **-e** option is accidentally ruined in the process  
18190 of editing, it may be regenerated by re-executing the `get` command with the **-k**  
18191 option in place of the **-e** option.

18192 SCCS file protection specified via the ceiling, floor, and authorized user list stored  
18193 in the SCCS file shall be enforced when the **-e** option is used.

18194 **-b** Use with the **-e** option to indicate that the new delta should have an *SID* in a new  
18195 branch as shown in the table below. This option shall be ignored if the **b** flag is not  
18196 present in the file or if the retrieved delta is not a leaf delta. (A leaf delta is one that  
18197 has no successors on the SCCS file tree.)

- 18198                   **Note:**       A branch delta may always be created from a non-leaf delta.
- 18199           **-i list**       Indicate a *list* of deltas to be included (forced to be applied) in the creation of the  
18200                   generated file. The *list* has the following syntax:
- 18201                   <list> ::= <range> | <list> , <range>  
18202                   <range> ::= SID | SID - SID
- 18203                   SID, the SCCS Identification of a delta, may be in any form shown in the "SID  
18204                   Specified" column of the table in the EXTENDED DESCRIPTION section. Partial  
18205                   SIDs are interpreted as shown in the "SID Retrieved" column of the table.
- 18206           **-x list**       Indicate a *list* of deltas to be excluded (forced not to be applied) in the creation of  
18207                   the generated file. See the **-i** option for the *list* format.
- 18208           **-k**           Suppress replacement of identification keywords (see below) in the retrieved text  
18209                   by their value. The **-k** option is implied by the **-e** option.
- 18210           **-l**           Write a delta summary into an **l-file**.
- 18211           **-L**           Write a delta summary to standard output. All informative output that normally is  
18212                   written to standard output shall be written to standard error instead, unless the **-s**  
18213                   option is used, in which case it shall be suppressed.
- 18214           **-p**           Write the text retrieved from the SCCS file to the standard output. No **g-file** shall  
18215                   be created. All informative output that normally goes to the standard output shall  
18216                   go to standard error instead, unless the **-s** option is used, in which case it  
18217                   disappears.
- 18218           **-s**           Suppress all informative output normally written to standard output. However,  
18219                   fatal error messages (which shall always be written to the standard error) remain  
18220                   unaffected.
- 18221           **-m**           Precede each text line retrieved from the SCCS file by the SID of the delta that  
18222                   inserted the text line in the SCCS file. The format is:
- 18223                   "%s\t%s", <SID>, <text line>
- 18224           **-n**           Precede each generated text line with the %M% identification keyword value (see  
18225                   below). The format is:
- 18226                   "%s\t%s", <%M% value>, <text line>
- 18227                   When both the **-m** and **-n** options are used, the <text line> shall be replaced by the  
18228                   **-m** option-generated format.
- 18229           **-g**           Suppress the actual retrieval of text from the SCCS file. It is primarily used to  
18230                   generate an **l-file**, or to verify the existence of a particular SID.
- 18231           **-t**           Use to access the most recently created (top) delta in a given release (for example,  
18232                   **-r 1**), or release and level (for example, **-r 1.2**).

18233 **OPERANDS**

18234           The following operands shall be supported:

- 18235           **file**       A path name of an existing SCCS file or a directory. If *file* is a directory, the *get*  
18236                   utility shall behave as though each file in the directory were specified as a named  
18237                   file, except that non-SCCS files (last component of the path name does not begin  
18238                   with **s.**) and unreadable files shall be silently ignored.
- 18239                   If a single instance *file* is specified as **'-'**, the standard input is read; each line of  
18240                   the standard input is taken to be the name of an SCCS file to be processed. Non-

- 18241                   SCCS files and unreadable files shall be silently ignored.
- 18242 **STDIN**
- 18243                   The standard input shall be a text file used only if the *file* operand is specified as '-'. Each line  
18244                   of the text file shall be interpreted as an SCCS path name.
- 18245 **INPUT FILES**
- 18246                   The SCCS files are files of an unspecified format.
- 18247 **ENVIRONMENT VARIABLES**
- 18248                   The following environment variables shall affect the execution of *get*:
- 18249                   *LANG*       Provide a default value for the internationalization variables that are unset or null.  
18250                   If *LANG* is unset or null, the corresponding value from the implementation-  
18251                   defined default locale shall be used. If any of the internationalization variables  
18252                   contains an invalid setting, the utility shall behave as if none of the variables had  
18253                   been defined.
- 18254                   *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
18255                   internationalization variables.
- 18256                   *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
18257                   characters (for example, single-byte as opposed to multi-byte characters in  
18258                   arguments and input files).
- 18259                   *LC\_MESSAGES*
- 18260                   Determine the locale that should be used to affect the format and contents of  
18261                   diagnostic messages written to standard error, and informative messages written  
18262                   to standard output (or standard error, if the **-p** option is used).
- 18263                   *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 18264 **ASYNCHRONOUS EVENTS**
- 18265                   Default.
- 18266 **STDOUT**
- 18267                   For each file processed, *get* shall write to standard output the SID being accessed and the number  
18268                   of lines retrieved from the SCCS file, in the following format:
- 18269                   "*%s\n%d lines\n*", *<SID>*, *<number of lines>*
- 18270                   If the **-e** option is used, the SID of the delta to be made shall appear after the SID accessed and  
18271                   before the number of lines generated, in the POSIX locale:
- 18272                   "*%s\nnew delta %s\n%d\n*", *<SID accessed>*, *<SID to be made>*,  
18273                   *<number of lines>*
- 18274                   If there is more than one named file or if a directory or standard input is named, each path name  
18275                   shall be written before each of the lines shown in one of the preceding formats:
- 18276                   "*\n%s:\n*", *<pathname>*
- 18277                   If the **-L** option is used, a delta summary shall be written following the format specified below  
18278                   for **l-files**.
- 18279                   If the **-i** option is used, included deltas are listed following the notation, in the POSIX locale:
- 18280                   "*Included:\n*"
- 18281                   If the **-x** option is used, excluded deltas are listed following the notation, in the POSIX locale:

18282 "Excluded:\n"

18283 If the **-p** or **-L** options are specified, the standard output consists of the text retrieved from the  
18284 SCCS file.

#### 18285 **STDERR**

18286 The standard error shall be used only for diagnostic messages, except if the **-p** or **-L** options are  
18287 specified, it includes all informative messages normally sent to standard output.

#### 18288 **OUTPUT FILES**

18289 Several auxiliary files may be created by *get*. These files are known generically as the **g-file**, **l-**  
18290 **file**, **p-file**, and **z-file**. The letter before the hyphen is called the *tag*. An auxiliary file name is  
18291 formed from the SCCS file name: the application shall ensure that the last component of all  
18292 SCCS file names is of the form *s.module-name*; the auxiliary files are named by replacing the  
18293 leading *s* with the tag. The **g-file** is an exception to this scheme: the **g-file** is named by removing  
18294 the *s*. prefix. For example, for *s.xyz.c*, the auxiliary file names would be *xyz.c*, *l.xyz.c*, *p.xyz.c*,  
18295 and *z.xyz.c*, respectively.

18296 The **g-file**, which contains the generated text, is created in the current directory (unless the **-p**  
18297 option is used). A **g-file** is created in all cases, whether or not any lines of text were generated by  
18298 the *get*. It is owned by the real user. If the **-k** option is used or implied, it is writable by the  
18299 owner only (read-only for everyone else); otherwise, it is read-only. Only the real user need have  
18300 write permission in the current directory.

18301 The **l-file** contains a table showing which deltas were applied in generating the retrieved text.  
18302 The **l-file** is created in the current directory if the **-l** option is used; it is read-only and it is  
18303 owned by the real user. Only the real user need have write permission in the current directory.

18304 Lines in the **l-file** have the following format:

```
18305 "%c%c%cΔ%s\t%sΔ%s\n", <code1>, <code2>, <code3>,
18306 <SID>, <date-time>, <login>
```

18307 where the entries are:

18308 **<code1>** A <space> character if the delta was applied; ' \* ' otherwise.

18309 **<code2>** A <space> character if the delta was applied or was not applied and ignored; ' \* '  
18310 if the delta was not applied and was not ignored.

18311 **<code3>** A character indicating a special reason why the delta was or was not applied:

18312 **I** Included.

18313 **X** Excluded.

18314 **C** Cut off (by a **-c** option).

18315 **<date-time>** Date and time (using the *date* utility's %y/%m/%d %T format) of creation.

18316 **<login>** Login name of person who created *delta*.

18317 The comments and MR data shall follow on subsequent lines, indented one <tab> character. A  
18318 blank line terminates each entry.

18319 The **p-file** is used to pass information resulting from a *get* with a **-e** option along to *delta*. Its  
18320 contents are also used to prevent a subsequent execution of *get* with a **-e** option for the same SID  
18321 until *delta* is executed or the joint edit flag, **j**, is set in the SCCS file. The **p-file** shall be created in  
18322 the directory containing the SCCS file and the application shall ensure that the effective user has  
18323 write permission in that directory. It is writable by owner only, and it is owned by the effective  
18324 user. Each line in the **p-file** has the following format:

18325 "%sΔ%sΔ%sΔ%s%s%s\n", <g-file SID>,  
 18326 <SID of new delta>, <login-name of real user>,  
 18327 <date-time>, <i-value>, <x-value>

18328 where <i-value> uses the format " " if no **-i** option was specified, and uses the format:

18329 "Δ-i%s", <-i option option-argument>

18330 if a **-i** option was specified and <x-value> uses the format " " if no **-x** option was specified, and  
 18331 uses the format:

18332 "Δ-x%s", <-x option option-argument>

18333 if a **-x** option was specified. There can be an arbitrary number of lines in the **p-file** at any time;  
 18334 no two lines can have the same new delta SID.

18335 The **z-file** serves as a lock-out mechanism against simultaneous updates. Its contents are the  
 18336 binary process ID of the command (that is, *get*) that created it. The **z-file** is created in the  
 18337 directory containing the SCCS file for the duration of *get*. The same protection restrictions as  
 18338 those for the **p-file** apply for the **z-file**. The **z-file** shall be created read-only.

### 18339 EXTENDED DESCRIPTION

| Determination of SCCS Identification String |                       |                                                |                  |                               |
|---------------------------------------------|-----------------------|------------------------------------------------|------------------|-------------------------------|
| SID*<br>Specified                           | -b Keyletter<br>Used† | Other<br>Conditions                            | SID<br>Retrieved | SID of Delta<br>to be Created |
| 18343 none‡                                 | no                    | R defaults to mR                               | mR.mL            | mR.(mL+1)                     |
| 18344 none‡                                 | yes                   | R defaults to mR                               | mR.mL            | mR.mL.(mB+1).1                |
| 18345 R                                     | no                    | R > mR                                         | mR.mL            | R.1***                        |
| 18346 R                                     | no                    | R = mR                                         | mR.mL            | mR.(mL+1)                     |
| 18347 R                                     | yes                   | R > mR                                         | mR.mL            | mR.mL.(mB+1).1                |
| 18348 R                                     | yes                   | R = mR                                         | mR.mL            | mR.mL.(mB+1).1                |
| 18349 R                                     | -                     | R < mR and<br>R does not exist                 | hR.mL**          | hR.mL.(mB+1).1                |
| 18351 R                                     | -                     | Trunk successor in release > R<br>and R exists | R.mL             | R.mL.(mB+1).1                 |
| 18353 R.L                                   | no                    | No trunk successor                             | R.L              | R.(L+1)                       |
| 18354 R.L                                   | yes                   | No trunk successor                             | R.L              | R.L.(mB+1).1                  |
| 18355 R.L                                   | -                     | Trunk successor<br>in release ≥ R              | R.L              | R.L.(mB+1).1                  |
| 18357 R.L.B                                 | no                    | No branch successor                            | R.L.B.mS         | R.L.B.(mS+1)                  |
| 18358 R.L.B                                 | yes                   | No branch successor                            | R.L.B.mS         | R.L.(mB+1).1                  |
| 18359 R.L.B.S                               | no                    | No branch successor                            | R.L.B.S          | R.L.B.(S+1)                   |
| 18360 R.L.B.S                               | yes                   | No branch successor                            | R.L.B.S          | R.L.(mB+1).1                  |
| 18361 R.L.B.S                               | -                     | Branch successor                               | R.L.B.S          | R.L.(mB+1).1                  |

18362 \* R, L, B, and S are the release, level, branch, and sequence components of the SID,  
 18363 respectively; m means maximum. Thus, for example, R.mL means "the maximum level  
 18364 number within release R"; R.L.(mB+1).1 means "the first sequence number on the new  
 18365 branch (that is, maximum branch number plus one) of level L within release R". Note  
 18366 that if the SID specified is of the form R.L, R.L.B, or R.L.B.S, each of the specified  
 18367 components shall exist.

|       |     |                                                                                                           |
|-------|-----|-----------------------------------------------------------------------------------------------------------|
| 18368 | **  | hR is the highest existing release that is lower than the specified, nonexistent, release R.              |
| 18369 | *** | This is used to force creation of the first delta in a new release.                                       |
| 18370 | †   | The <b>-b</b> option is effective only if the <b>b</b> flag is present in the file. An entry of '–' means |
| 18371 |     | “irrelevant”.                                                                                             |
| 18372 | ‡   | This case applies if the <b>d</b> (default SID) flag is not present in the file. If the <b>d</b> flag is  |
| 18373 |     | present in the file, then the SID obtained from the <b>d</b> flag is interpreted as if it had been        |
| 18374 |     | specified on the command line. Thus, one of the other cases in this table applies.                        |

### 18375 Identification Keywords

18376 Identifying information shall be inserted into the text retrieved from the SCCS file by replacing  
 18377 identification keywords with their value wherever they occur. The following keywords may be  
 18378 used in the text stored in an SCCS file:

|       |     |                                                                                               |
|-------|-----|-----------------------------------------------------------------------------------------------|
| 18379 | %M% | Module name: either the value of the <b>m</b> flag in the file, or if absent, the name of the |
| 18380 |     | SCCS file with the leading <b>s.</b> removed.                                                 |
| 18381 | %I% | SCCS identification (SID) (%R%.%L% or %R%.%L%.%B%.%S%) of the retrieved                       |
| 18382 |     | text.                                                                                         |
| 18383 | %R% | Release.                                                                                      |
| 18384 | %L% | Level.                                                                                        |
| 18385 | %B% | Branch.                                                                                       |
| 18386 | %S% | Sequence.                                                                                     |
| 18387 | %D% | Current date (YY/MM/DD).                                                                      |
| 18388 | %H% | Current date (MM/DD/YY).                                                                      |
| 18389 | %T% | Current time (HH:MM:SS).                                                                      |
| 18390 | %E% | Date newest applied delta was created (YY/MM/DD).                                             |
| 18391 | %G% | Date newest applied delta was created (MM/DD/YY).                                             |
| 18392 | %U% | Time newest applied delta was created (HH:MM:SS).                                             |
| 18393 | %Y% | Module type: value of the <b>t</b> flag in the SCCS file.                                     |
| 18394 | %F% | SCCS file name.                                                                               |
| 18395 | %P% | SCCS absolute path name.                                                                      |
| 18396 | %Q% | The value of the <b>q</b> flag in the file.                                                   |
| 18397 | %C% | Current line number. This keyword is intended for identifying messages output by              |
| 18398 |     | the program, such as “this should not have happened” type errors. It is not                   |
| 18399 |     | intended to be used on every line to provide sequence numbers.                                |
| 18400 | %Z% | The four-character string "@(#)" recognizable by <i>what</i> .                                |
| 18401 | %W% | A shorthand notation for constructing <i>what</i> strings:                                    |
| 18402 |     | %W%=%Z%%M%<tab>%I%                                                                            |
| 18403 | %A% | Another shorthand notation for constructing <i>what</i> strings:                              |
| 18404 |     | %A%=%Z%%Y%%M%%I%%Z%                                                                           |



18405 **EXIT STATUS**

18406 The following exit values shall be returned:

18407 0 Successful completion.

18408 >0 An error occurred.

18409 **CONSEQUENCES OF ERRORS**

18410 Default.

18411 **APPLICATION USAGE**

18412 None.

18413 **EXAMPLES**

18414 None.

18415 **RATIONALE**

18416 None.

18417 **FUTURE DIRECTIONS**

18418 The **-lp** option may be withdrawn in a future issue.

18419 **SEE ALSO**

18420 *admin, delta, prs, what*

18421 **CHANGE HISTORY**

18422 First released in Issue 2.

18423 **Issue 4**

18424 Format reorganized.

18425 Exceptions to Utility Syntax Guidelines conformance noted.

18426 Internationalized environment variable support mandated.

18427 **Issue 5**

18428 Correction to the first format string in STDOUT.

18429 The interpretation of the *YY* component of the **-c cutoff** argument is noted.

18430 **Issue 6**

18431 The obsolescent SYNOPSIS is removed, removing the **-lp** option.

18432 The Open Group corrigenda item U025/5 has been applied, correcting text in the OPTIONS section.

18434 The normative text is reworded to avoid use of the term “must” for application requirements. |

18435 The normative text is reworded to emphasise the term “shall” for implementation requirements. |

18436 The Open Group corrigenda item U048/1 has been applied. |

18437 **NAME**18438 `getconf` — get configuration values18439 **SYNOPSIS**18440 `getconf` [ `-v specification` ] `system_var`18441 `getconf` [ `-v specification` ] `path_var pathname`18442 **DESCRIPTION**18443 In the first synopsis form, the `getconf` utility shall write to the standard output the value of the  
18444 variable specified by the `system_var` operand.18445 In the second synopsis form, the `getconf` utility shall write to the standard output the value of the  
18446 variable specified by the `path_var` operand for the path specified by the `pathname` operand.18447 The value of each configuration variable shall be determined as if it were obtained by calling the  
18448 function from which it is defined to be available by this volume of IEEE Std. 1003.1-200x or by  
18449 the System Interfaces volume of IEEE Std. 1003.1-200x (see the OPERANDS section). The value  
18450 shall reflect conditions in the current operating environment.18451 **OPTIONS**18452 The `getconf` utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
18453 12.2, Utility Syntax Guidelines.

18454 The following option shall be supported:

18455 `-v specification`18456 Indicate a specific specification and version for which configuration variables shall  
18457 be determined. If this option is not specified, the values returned correspond to an  
18458 implementation default conforming compilation environment.

18459 If the command:

18460 `getconf _POSIX_V6_ILP32_OFF32`18461 does not write "`-1\n`" or "`undefined\n`" to standard output, then commands of  
18462 the form:18463 `getconf -v POSIX_V6_ILP32_OFF32 ...`18464 determine values for configuration variables corresponding to the  
18465 `POSIX_V6_ILP32_OFF32` compilation environment specified in *c99* (on page 2425),  
18466 EXTENDED DESCRIPTION.

18467 If the command:

18468 `getconf _POSIX_V6_ILP32_OFFBIG`18469 does not write "`-1\n`" or "`undefined\n`" to standard output, then commands of  
18470 the form:18471 `getconf -v POSIX_V6_ILP32_OFFBIG ...`18472 determine values for configuration variables corresponding to the  
18473 `POSIX_V6_ILP32_OFFBIG` compilation environment specified in *c99* (on page  
18474 2425), EXTENDED DESCRIPTION.

18475 If the command:

18476 `getconf _POSIX_V6_LP64_OFF64`18477 does not write "`-1\n`" or "`undefined\n`" to standard output, then commands of  
18478 the form:

18479 `getconf -v POSIX_V6_LP64_OFF64 ...`

18480 determine values for configuration variables corresponding to the  
18481 `POSIX_V6_LP64_OFF64` compilation environment specified in *c99* (on page 2425),  
18482 EXTENDED DESCRIPTION.

18483 If the command:

18484 `getconf _POSIX_V6_LPBIG_OFFBIG`

18485 does not write "-1\n" or "undefined\n" to standard output, then commands of  
18486 the form:

18487 `getconf -v POSIX_V6_LPBIG_OFFBIG ...`

18488 determine values for configuration variables corresponding to the  
18489 `POSIX_V6_LPBIG_OFFBIG` compilation environment specified in *c99* (on page  
18490 2425), EXTENDED DESCRIPTION.

### 18491 OPERANDS

18492 The following operands shall be supported:

18493 *path\_var* A name of a configuration variable. All of the variables in the *pathconf()* function  
18494 defined in the System Interfaces volume of IEEE Std. 1003.1-200x are supported  
18495 and the implementation may add other local variables.

18496 *pathname* A path name for which the variable specified by *path\_var* is to be determined.

18497 *system\_var* A name of a configuration variable. All of the variables in the *confstr()* and  
18498 *sysconf()* functions defined in the System Interfaces volume of  
18499 IEEE Std. 1003.1-200x shall be supported and the implementation may add other  
18500 local values.

18501 When the symbol listed in the first column of the following table is used as the  
18502 *system\_var* operand, *getconf* yields the same value as *confstr()* when called with the  
18503 value in the second column:

| <i>system_var</i>                     | <i>confstr()</i> Name Value         |
|---------------------------------------|-------------------------------------|
| 18506 PATH                            | _CS_PATH                            |
| 18507 POSIX_V6_ILP32_OFF32_CFLAGS     | _CS_POSIX_V6_ILP32_OFF32_CFLAGS     |
| 18508 POSIX_V6_ILP32_OFF32_LDFLAGS    | _CS_POSIX_V6_ILP32_OFF32_LDFLAGS    |
| 18509 POSIX_V6_ILP32_OFF32_LIBS       | _CS_POSIX_V6_ILP32_OFF32_LIBS       |
| 18510 POSIX_V6_ILP32_OFF32_LINTFLAGS  | _CS_POSIX_V6_ILP32_OFF32_LINTFLAGS  |
| 18511 POSIX_V6_ILP32_OFFBIG_CFLAGS    | _CS_POSIX_V6_ILP32_OFFBIG_CFLAGS    |
| 18512 POSIX_V6_ILP32_OFFBIG_LDFLAGS   | _CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS   |
| 18513 POSIX_V6_ILP32_OFFBIG_LIBS      | _CS_POSIX_V6_ILP32_OFFBIG_LIBS      |
| 18514 POSIX_V6_ILP32_OFFBIG_LINTFLAGS | _CS_POSIX_V6_ILP32_OFFBIG_LINTFLAGS |
| 18515 POSIX_V6_LP64_OFF64_CFLAGS      | _CS_POSIX_V6_LP64_OFF64_CFLAGS      |
| 18516 POSIX_V6_LP64_OFF64_LDFLAGS     | _CS_POSIX_V6_LP64_OFF64_LDFLAGS     |
| 18517 POSIX_V6_LP64_OFF64_LIBS        | _CS_POSIX_V6_LP64_OFF64_LIBS        |
| 18518 POSIX_V6_LP64_OFF64_LINTFLAGS   | _CS_POSIX_V6_LP64_OFF64_LINTFLAGS   |
| 18519 POSIX_V6_LPBIG_OFFBIG_CFLAGS    | _CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS    |
| 18520 POSIX_V6_LPBIG_OFFBIG_LDFLAGS   | _CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS   |
| 18521 POSIX_V6_LPBIG_OFFBIG_LIBS      | _CS_POSIX_V6_LPBIG_OFFBIG_LIBS      |

18522

18523

18524

18525 XSI

18526

18527

18528

18529

18530

18531

18532

18533

18534

18535

18536

18537

18538

18539

18540

| <i>system_var</i>                    | <i>confstr() Name Value</i>         |
|--------------------------------------|-------------------------------------|
| POSIX_V6_LPBIG_OFFBIG_LINTFLAGS      | _CS_POSIX_V6_LPBIG_OFFBIG_LINTFLAGS |
| XBS5_ILP32_OFF32_CFLAGS (LEGACY)     | _CS_XBS5_ILP32_OFF32_CFLAGS         |
| XBS5_ILP32_OFF32_LDFLAGS (LEGACY)    | _CS_XBS5_ILP32_OFF32_LDFLAGS        |
| XBS5_ILP32_OFF32_LIBS (LEGACY)       | _CS_XBS5_ILP32_OFF32_LIBS           |
| XBS5_ILP32_OFF32_LINTFLAGS (LEGACY)  | _CS_XBS5_ILP32_OFF32_LINTFLAGS      |
| XBS5_ILP32_OFFBIG_CFLAGS (LEGACY)    | _CS_XBS5_ILP32_OFFBIG_CFLAGS        |
| XBS5_ILP32_OFFBIG_LDFLAGS (LEGACY)   | _CS_XBS5_ILP32_OFFBIG_LDFLAGS       |
| XBS5_ILP32_OFFBIG_LIBS (LEGACY)      | _CS_XBS5_ILP32_OFFBIG_LIBS          |
| XBS5_ILP32_OFFBIG_LINTFLAGS (LEGACY) | _CS_XBS5_ILP32_OFFBIG_LINTFLAGS     |
| XBS5_LP64_OFF64_CFLAGS (LEGACY)      | _CS_XBS5_LP64_OFF64_CFLAGS          |
| XBS5_LP64_OFF64_LDFLAGS (LEGACY)     | _CS_XBS5_LP64_OFF64_LDFLAGS         |
| XBS5_LP64_OFF64_LIBS (LEGACY)        | _CS_XBS5_LP64_OFF64_LIBS            |
| XBS5_LP64_OFF64_LINTFLAGS (LEGACY)   | _CS_XBS5_LP64_OFF64_LINTFLAGS       |
| XBS5_LPBIG_OFFBIG_CFLAGS (LEGACY)    | _CS_XBS5_LPBIG_OFFBIG_CFLAGS        |
| XBS5_LPBIG_OFFBIG_LDFLAGS (LEGACY)   | _CS_XBS5_LPBIG_OFFBIG_LDFLAGS       |
| XBS5_LPBIG_OFFBIG_LIBS (LEGACY)      | _CS_XBS5_LPBIG_OFFBIG_LIBS          |
| XBS5_LPBIG_OFFBIG_LINTFLAGS (LEGACY) | _CS_XBS5_LPBIG_OFFBIG_LINTFLAGS     |

18541 **STDIN**

18542 Not used.

18543 **INPUT FILES**

18544 None.

18545 **ENVIRONMENT VARIABLES**18546 The following environment variables shall affect the execution of *getconf*:

18547 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 18548 If *LANG* is unset or null, the corresponding value from the implementation-  
 18549 defined default locale shall be used. If any of the internationalization variables  
 18550 contains an invalid setting, the utility shall behave as if none of the variables had  
 18551 been defined.

18552 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 18553 internationalization variables.

18554 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 18555 characters (for example, single-byte as opposed to multi-byte characters in  
 18556 arguments).

18557 *LC\_MESSAGES*  
 18558 Determine the locale that should be used to affect the format and contents of  
 18559 diagnostic messages written to standard error.

18560 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

18561 **ASYNCHRONOUS EVENTS**

18562 Default.

18563 **STDOUT**

18564 If the specified variable is defined on the system and its value is described to be available from  
 18565 the *confstr()* function defined in the System Interfaces volume of IEEE Std. 1003.1-200x, its value  
 18566 shall be written in the following format:

18567 "%s\n", &lt;value&gt;

18568 Otherwise, if the specified variable is defined on the system, its value shall be written in the  
18569 following format:

18570 "%d\n", <value>

18571 If the specified variable is valid, but is undefined on the system, *getconf* shall write using the  
18572 following format:

18573 "undefined\n"

18574 If the variable name is invalid or an error occurs, nothing shall be written to standard output.

#### 18575 **STDERR**

18576 Used only for diagnostic messages.

#### 18577 **OUTPUT FILES**

18578 None.

#### 18579 **EXTENDED DESCRIPTION**

18580 None.

#### 18581 **EXIT STATUS**

18582 The following exit values shall be returned:

18583 0 The specified variable is valid and information about its current state was written  
18584 successfully.

18585 >0 An error occurred.

#### 18586 **CONSEQUENCES OF ERRORS**

18587 Default.

#### 18588 **APPLICATION USAGE**

18589 None.

#### 18590 **EXAMPLES**

18591 The following example illustrates the value of {NGROUPS\_MAX}:

18592 `getconf NGROUPS_MAX`

18593 The following example illustrates the value of {NAME\_MAX} for a specific directory:

18594 `getconf NAME_MAX /usr`

18595 The following example shows how to deal more carefully with results that might be unspecified:

```
18596 if value=$(getconf PATH_MAX /usr); then
18597 if ["$value" = "undefined"]; then
18598 echo PATH_MAX in /usr is infinite.
18599 else
18600 echo PATH_MAX in /usr is $value.
18601 fi
18602 else
18603 echo Error in getconf.
18604 fi
```

18605 **Note that:**

18606 `sysconf(_SC_POSIX_C_BIND);`

18607 **and:**

18608 `system("getconf POSIX2_C_BIND");`

18609 in a C program could give different answers. The `sysconf()` call supplies a value that corresponds  
18610 to the conditions when the program was either compiled or executed, depending on the  
18611 implementation; the `system()` call to `getconf` always supplies a value corresponding to conditions  
18612 when the program is executed.

#### 18613 RATIONALE

18614 The original need for this utility, and for the `confstr()` function, was to provide a way of finding  
18615 the configuration-defined default value for the `PATH` environment variable. Since `PATH` can be  
18616 modified by the user to include directories that could contain utilities replacing the standard  
18617 utilities, shell scripts need a way to determine the system-supplied `PATH` environment variable  
18618 value that contains the correct search path for the standard utilities. It was later suggested that  
18619 access to the other variables described in this volume of IEEE Std. 1003.1-200x could also be  
18620 useful to applications.

18621 This functionality of `getconf` would not be adequately subsumed by another command such as:

18622 `grep var /etc/conf`

18623 because such a strategy would provide correct values for neither those variables that can vary at  
18624 runtime, nor those that can vary depending on the path.

18625 Early proposal versions of `getconf` specified exit status 1 when the specified variable was valid,  
18626 but not defined on the system. The output string "undefined" is now used to specify this case  
18627 with exit code 0 because so many things depend on an exit code of zero when an invoked utility  
18628 is successful.

#### 18629 FUTURE DIRECTIONS

18630 None.

#### 18631 SEE ALSO

18632 `c99`, the System Interfaces volume of IEEE Std. 1003.1-200x, `confstr()`, `pathconf()`, `sysconf()`

#### 18633 CHANGE HISTORY

18634 First released in Issue 4.

#### 18635 Issue 4, Version 2

18636 The following changes are made in the table of values for `system_var`:

- 18637 • Names beginning with `POSIX_` are changed to begin with `_POSIX_`.
- 18638 • Names beginning with `XOPEN_` are changed to begin with `_XOPEN_`.
- 18639 • `{MN_NMAX}` is changed to `{NL_MAX}`.
- 18640 • `{NL_SET_MAX}` is changed to `{NL_SETMAX}`.
- 18641 • `{NL_TEXT_MAX}` is changed to `{NL_TEXTMAX}`.
- 18642 • The `_XOPEN_CRYPT`, `_XOPEN_ENH_I18N`, and `_XOPEN_SHM` configuration variables are  
18643 added to the list.

#### 18644 Issue 5

18645 In the OPERANDS section:

- 18646 • `{NL_MAX}` is changed to `{NL_NMAX}`.
- 18647 • Entries beginning `NL_` are deleted from the list of standard configuration variables.
- 18648 • The list of variables previously marked `UX` is merged with the list marked `EX`.

- 18649           • Operands are added to support new Option Groups.
- 18650           • Operands are added so that *getconf* can determine supported programming environments.
- 18651 **Issue 6**
- 18652           The Open Group corrigenda item U029/4 has been applied, correcting the example command in
- 18653           the last paragraph of the OPTIONS section.
- 18654           The following new requirements on POSIX implementations derive from alignment with the
- 18655           Single UNIX Specification:
- 18656           • Operands are added to determine supported programming environments.
- 18657           This reference page is updated for alignment with the ISO/IEC 9899: 1999 standard. Specifically,
- 18658           new macros for *c99* programming environments are introduced.

18659 **NAME**

18660 getopts — parse utility options

18661 **SYNOPSIS**

18662 `getopts optstring name [arg...]`

18663 **DESCRIPTION**

18664 The *getopts* utility can be used to retrieve options and option-arguments from a list of  
 18665 parameters. It shall support the Utility Syntax Guidelines 3 to 10, inclusive, described in the Base  
 18666 Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

18667 Each time it is invoked, the *getopts* utility shall place the value of the next option in the shell  
 18668 variable specified by the *name* operand and the index of the next argument to be processed in the  
 18669 shell variable *OPTIND*. Whenever the shell is invoked, *OPTIND* shall be initialized to 1.

18670 When the option requires an option-argument, the *getopts* utility shall place it in the shell  
 18671 variable *OPTARG*. If no option was found, or if the option that was found does not have an  
 18672 option-argument, *OPTARG* shall be unset.

18673 If an option character not contained in the *optstring* operand is found where an option character  
 18674 is expected, the shell variable specified by *name* shall be set to the question-mark ('?') character.  
 18675 In this case, if the first character in *optstring* is a colon (':'), the shell variable *OPTARG* shall be  
 18676 set to the option character found, but no output shall be written to standard error; otherwise, the  
 18677 shell variable *OPTARG* shall be unset and a diagnostic message shall be written to standard  
 18678 error. This condition shall be considered to be an error detected in the way arguments were  
 18679 presented to the invoking application, but shall be not an error in *getopts* processing.

18680 If an option-argument is missing:

- 18681 • If the first character of *optstring* is a colon, the shell variable specified by *name* shall be set to  
 18682 the colon character and the shell variable *OPTARG* shall be set to the option character found.
- 18683 • Otherwise, the shell variable specified by *name* shall be set to the question-mark character,  
 18684 the shell variable *OPTARG* shall be unset, and a diagnostic message shall be written to  
 18685 standard error. This condition shall be considered to be an error detected in the way  
 18686 arguments were presented to the invoking application, but shall not be an error in *getopts*  
 18687 processing; a diagnostic message shall be written as stated, but the exit status shall be zero.

18688 When the end of options is encountered, the *getopts* utility shall exit with a return value greater  
 18689 than zero; the shell variable *OPTIND* shall be set to the index of the first non-option-argument,  
 18690 where the first "—" argument is considered to be an option-argument if there are no other non-  
 18691 option-arguments appearing before it, or the value "\$#+1" if there are no non-option-  
 18692 arguments; the *name* variable shall be set to the question-mark character. Any of the following  
 18693 shall identify the end of options: the special option "—", finding an argument that does not  
 18694 begin with a '-', or encountering an error.

18695 The shell variables *OPTIND* and *OPTARG* shall be local to the caller of *getopts* and shall not be  
 18696 exported by default.

18697 The shell variable specified by the *name* operand, *OPTIND* and *OPTARG* shall affect the current  
 18698 shell execution environment; see Section 2.13 (on page 2273).

18699 If the application sets *OPTIND* to the value 1, a new set of parameters can be used: either the  
 18700 current positional parameters or new *arg* values. Any other attempt to invoke *getopts* multiple  
 18701 times in a single shell execution environment with parameters (positional parameters or *arg*  
 18702 operands) that are not the same in all invocations, or with an *OPTIND* value modified to be a  
 18703 value other than 1, produces unspecified results.



18704 **OPTIONS**

18705       None.

18706 **OPERANDS**

18707       The following operands shall be supported:

18708       *optstring*     A string containing the option characters recognized by the utility invoking *getopts*.  
 18709                    If a character is followed by a colon, the option shall be expected to have an  
 18710                    argument, which should be supplied as a separate argument. Applications should  
 18711                    specify an option character and its option-argument as separate arguments, but  
 18712                    *getopts* shall interpret the characters following an option character requiring  
 18713                    arguments as an argument whether or not this is done. An explicit null option-  
 18714                    argument need not be recognized if it is not supplied as a separate argument when  
 18715                    *getopts* is invoked. (See also the *getopt()* function defined in the System Interfaces  
 18716                    volume of IEEE Std. 1003.1-200x.) The characters question-mark and colon shall  
 18717                    not be used as option characters by an application. The use of other option  
 18718                    characters that are not alphanumeric produces unspecified results. If the option-  
 18719                    argument is not supplied as a separate argument from the option character, the  
 18720                    value in *OPTARG* shall be stripped of the option character and the '-'. The first  
 18721                    character in *optstring* determines how *getopts* behaves if an option character is not  
 18722                    known or an option-argument is missing.

18723       *name*         The name of a shell variable that shall be set by the *getopts* utility to the option  
 18724                    character that was found.

18725       The *getopts* utility by default shall parse positional parameters passed to the invoking shell  
 18726       procedure. If *args* are given, they shall be parsed instead of the positional parameters.

18727 **STDIN**

18728       Not used.

18729 **INPUT FILES**

18730       None.

18731 **ENVIRONMENT VARIABLES**18732       The following environment variables shall affect the execution of *getopts*:

18733       *LANG*         Provide a default value for the internationalization variables that are unset or null.  
 18734                    If *LANG* is unset or null, the corresponding value from the implementation-  
 18735                    defined default locale shall be used. If any of the internationalization variables  
 18736                    contains an invalid setting, the utility shall behave as if none of the variables had  
 18737                    been defined.

18738       *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 18739                    internationalization variables.

18740       *LC\_CTYPE*     Determine the locale for the interpretation of sequences of bytes of text data as  
 18741                    characters (for example, single-byte as opposed to multi-byte characters in  
 18742                    arguments and input files).

18743       *LC\_MESSAGES*

18744                    Determine the locale that should be used to affect the format and contents of  
 18745                    diagnostic messages written to standard error.

18746 XSI       *NLSPATH*     Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

18747       *OPTIND*       This variable shall be used by the *getopts* utility as the index of the next argument  
 18748                    to be processed.

18749 **ASYNCHRONOUS EVENTS**

18750 Default.

18751 **STDOUT**

18752 Not used.

18753 **STDERR**

18754 Whenever an error is detected and the first character in the *optstring* operand is not a colon  
 18755 (' : '), a diagnostic message shall be written to standard error with the following information in  
 18756 an unspecified format:

18757 • The invoking program name shall be identified in the message. The invoking program name  
 18758 shall be the value of the shell special parameter 0 (see Section 2.5.2 (on page 2241)) at the time  
 18759 the *getopts* utility is invoked. A name equivalent to:

18760 basename "\$0"

18761 may be used.

18762 • If an option is found that was not specified in *optstring*, this error is identified and the invalid  
 18763 option character shall be identified in the message.

18764 • If an option requiring an option-argument is found, but an option-argument is not found,  
 18765 this error shall be identified and the invalid option character shall be identified in the  
 18766 message.

18767 **OUTPUT FILES**

18768 None.

18769 **EXTENDED DESCRIPTION**

18770 None.

18771 **EXIT STATUS**

18772 The following exit values shall be returned:

18773 0 An option, specified or unspecified by *optstring*, was found.

18774 &gt;0 The end of options was encountered or an error occurred.

18775 **CONSEQUENCES OF ERRORS**

18776 Default.

18777 **APPLICATION USAGE**

18778 Since *getopts* affects the current shell execution environment, it is generally provided as a shell  
 18779 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 18780 of the following:

18781 (getopts abc value "\$@")

18782 nohup getopts ...

18783 find . -exec getopts ... \;

18784 it does not affect the shell variables in the caller's environment.

18785 Note that shell functions share *OPTIND* with the calling shell even though the positional  
 18786 parameters are changed. If the calling shell and any of its functions uses *getopts* to parse  
 18787 arguments, the results are unspecified.

18788 **EXAMPLES**

18789 The following example script parses and displays its arguments:

18790 aflag=

18791 bflag=

```

18792 while getopts ab: name
18793 do
18794 case $name in
18795 a) aflag=1;;
18796 b) bflag=1
18797 bval="$OPTARG";;
18798 ?) printf "Usage: %s: [-a] [-b value] args\n" $0
18799 exit 2;;
18800 esac
18801 done
18802 if [! -z "$aflag"]; then
18803 printf "Option -a specified\n"
18804 fi
18805 if [! -z "$bflag"]; then
18806 printf 'Option -b "%s" specified\n' "$bval"
18807 fi
18808 shift $(($OPTIND - 1))
18809 printf "Remaining arguments are: %s\n" "$*"

```

#### 18810 RATIONALE

18811 The *getopts* utility was chosen in preference to the System V *getopt* utility because *getopts* handles  
 18812 option-arguments containing <blank> characters.

18813 The *OPTARG* variable is not mentioned in the ENVIRONMENT VARIABLES section because it  
 18814 does not affect the execution of *getopts*; it is one of the few “output-only” variables used by the  
 18815 standard utilities.

18816 The colon is not allowed as an option character because that is not historical behavior, and it  
 18817 violates the Utility Syntax Guidelines. The colon is now specified to behave as in the KornShell  
 18818 version of the *getopts* utility; when used as the first character in the *optstring* operand, it disables  
 18819 diagnostics concerning missing option-arguments and unexpected option characters. This  
 18820 replaces the use of the *OPTERR* variable that was specified in an early proposal.

18821 The formats of the diagnostic messages produced by the *getopts* utility and the *getopt()* function  
 18822 are not fully specified because implementations with superior (“friendlier”) formats objected to  
 18823 the formats used by some historical implementations. The standard developers considered it  
 18824 important that the information in the messages used be uniform between *getopts* and *getopt()*.  
 18825 Exact duplication of the messages might not be possible, particularly if a utility is built on  
 18826 another system that has a different *getopt()* function, but the messages must have specific  
 18827 information included so that the program name, invalid option character, and type of error can  
 18828 be distinguished by a user.

18829 Only a rare application program intercepts a *getopts* standard error message and wants to parse  
 18830 it. Therefore, implementations are free to choose the most usable messages they can devise. The  
 18831 following formats are used by many historical implementations:

18832 "%s: illegal option — %c\n", <program name>, <option character>

18833 "%s: option requires an argument — %c\n", <program name>, \  
 18834 <option character>

18835 Historical shells with built-in versions of *getopt()* or *getopts* have used different formats,  
 18836 frequently not even indicating the option character found in error.

18837 **FUTURE DIRECTIONS**

18838           None.

18839 **SEE ALSO**18840           The System Interfaces volume of IEEE Std. 1003.1-200x, *getopt()*18841 **CHANGE HISTORY**

18842           First released in Issue 4.

18843 **Issue 6**

18844           The normative text is reworded to avoid use of the term “must” for application requirements.

18845 **NAME**18846 `grep` — search a file for a pattern18847 **SYNOPSIS**18848 `grep [-E | -F][-c | -l | -q][-insvx] -e pattern_list...`  
18849 `[-f pattern_file]...[file...]`18850 `grep [-E | -F][-c | -l | -q][-insvx][-e pattern_list]...`  
18851 `-f pattern_file...[file...]`18852 `grep [-E | -F][-c | -l | -q][-insvx] pattern_list[file...]`18853 **DESCRIPTION**

18854 The *grep* utility shall search the input files, selecting lines matching one or more patterns; the  
 18855 types of patterns are controlled by the options specified. The patterns are specified by the `-e`  
 18856 option, `-f` option, or the *pattern\_list* operand. The *pattern\_list*'s value shall consist of one or more  
 18857 patterns separated by <newline> characters; the *pattern\_file*'s contents shall consist of one or  
 18858 more patterns terminated by <newline> characters. By default, an input line shall be selected if  
 18859 any pattern, treated as an entire basic regular expression (BRE) as described in the Base  
 18860 Definitions volume of IEEE Std. 1003.1-200x, Section 9.3, Basic Regular Expressions, matches any  
 18861 part of the line; a null BRE shall match every line. By default, each selected input line shall be  
 18862 written to the standard output.

18863 Regular expression matching shall be based on text lines. Since a <newline> character separates  
 18864 or terminates patterns (see the `-e` and `-f` options below), regular expressions cannot contain a  
 18865 <newline> character. Similarly, since patterns are matched against individual lines of the input,  
 18866 there is no way for a pattern to match a <newline> character found in the input.

18867 **OPTIONS**

18868 The *grep* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 18869 12.2, Utility Syntax Guidelines.

18870 The following options shall be supported:

18871 **-E** Match using extended regular expressions. Treat each pattern specified as an ERE,  
 18872 as described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.4,  
 18873 Extended Regular Expressions. If any entire ERE pattern matches some part of an  
 18874 input line, the line shall be matched. A null ERE shall match every line.

18875 **-F** Match using fixed strings. Treat each pattern specified as a string instead of a  
 18876 regular expression. If an input line contains any of the patterns as a contiguous  
 18877 sequence of bytes, the line shall be matched. A null string shall match every line.

18878 **-c** Write only a count of selected lines to standard output.

18879 **-e *pattern\_list***

18880 Specify one or more patterns to be used during the search for input. The  
 18881 application shall ensure that patterns in *pattern\_list* are separated by a <newline>  
 18882 character. A null pattern can be specified by two adjacent <newline> characters in  
 18883 *pattern\_list*. Unless the `-E` or `-F` option is also specified, each pattern shall be  
 18884 treated as a BRE, as described in the Base Definitions volume of  
 18885 IEEE Std. 1003.1-200x, Section 9.3, Basic Regular Expressions. Multiple `-e` and `-f`  
 18886 options shall be accepted by the *grep* utility. All of the specified patterns shall be  
 18887 used when matching lines, but the order of evaluation is unspecified.

18888 **-f *pattern\_file***

18889 Read one or more patterns from the file named by the path name *pattern\_file*.  
 18890 Patterns in *pattern\_file* shall be terminated by a <newline> character. A null pattern

- 18891 can be specified by an empty line in *pattern\_file*. Unless the **-E** or **-F** option is also  
 18892 specified, each pattern shall be treated as a BRE, as described in the Base  
 18893 Definitions volume of IEEE Std. 1003.1-200x, Section 9.3, Basic Regular  
 18894 Expressions.
- 18895 **-i** Perform pattern matching in searches without regard to case; see the Base  
 18896 Definitions volume of IEEE Std. 1003.1-200x, Section 9.2, Regular Expression  
 18897 General Requirements.
- 18898 **-l** (The letter ell.) Write only the names of files containing selected lines to standard  
 18899 output. Path names shall be written once per file searched. If the standard input is  
 18900 searched, a path name of "(standard input)" shall be written, in the POSIX  
 18901 locale. In other locales, "standard input" may be replaced by something more  
 18902 appropriate in those locales.
- 18903 **-n** Precede each output line by its relative line number in the file, each file starting at  
 18904 line 1. The line number counter shall be reset for each file processed.
- 18905 **-q** Quiet. Do not write anything to the standard output, regardless of matching lines.  
 18906 Exit with zero status if an input line is selected.
- 18907 **-s** Suppress the error messages ordinarily written for nonexistent or unreadable files.  
 18908 Other error messages shall not be suppressed.
- 18909 **-v** Select lines not matching any of the specified patterns. If the **-v** option is not  
 18910 specified, selected lines shall be those that match any of the specified patterns.
- 18911 **-x** Consider only input lines that use all characters in the line to match an entire fixed  
 18912 string or regular expression to be matching lines.

**18913 OPERANDS**

18914 The following operands shall be supported:

- 18915 *pattern\_list* Specify one or more patterns to be used during the search for input. This operand  
 18916 shall be treated as if it were specified as **-e pattern\_list**.
- 18917 *file* A path name of a file to be searched for the patterns. If no *file* operands are  
 18918 specified, the standard input shall be used.

**18919 STDIN**

18920 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES  
 18921 section.

**18922 INPUT FILES**

18923 The input files shall be text files.

**18924 ENVIRONMENT VARIABLES**

18925 The following environment variables shall affect the execution of *grep*:

- 18926 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 18927 If *LANG* is unset or null, the corresponding value from the implementation-  
 18928 defined default locale shall be used. If any of the internationalization variables  
 18929 contains an invalid setting, the utility shall behave as if none of the variables had  
 18930 been defined.
- 18931 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 18932 internationalization variables.
- 18933 *LC\_COLLATE*  
 18934 Determine the locale for the behavior of ranges, equivalence classes and multi-

- 18935 character collating elements within regular expressions.
- 18936 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
18937 characters (for example, single-byte as opposed to multi-byte characters in  
18938 arguments and input files) and the behavior of character classes within regular  
18939 expressions.
- 18940 **LC\_MESSAGES**  
18941 Determine the locale that should be used to affect the format and contents of  
18942 diagnostic messages written to standard error.
- 18943 **XSI NLS\_PATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 18944 **ASYNCHRONOUS EVENTS**  
18945 Default.
- 18946 **STDOUT**  
18947 If the **-l** option is in effect, and the **-q** option is not, the following shall be written for each file  
18948 containing at least one selected input line:  
18949 `"%s\n", <file>`
- 18950 Otherwise, if more than one *file* argument appears, and **-q** is not specified, the *grep* utility shall  
18951 prefix each output line by:  
18952 `"%s: ", <file>`
- 18953 The remainder of each output line shall depend on the other options specified:
- 18954 • If the **-c** option is in effect, the remainder of each output line shall contain:  
18955 `"%d\n", <count>`
- 18956 • Otherwise, if **-c** is not in effect and the **-n** option is in effect, the following shall be written to  
18957 standard output:  
18958 `"%d: ", <line number>`
- 18959 • Finally, the following shall be written to standard output:  
18960 `"%s", <selected-line contents>`
- 18961 **STDERR**  
18962 Used only for diagnostic messages.
- 18963 **OUTPUT FILES**  
18964 None.
- 18965 **EXTENDED DESCRIPTION**  
18966 None.
- 18967 **EXIT STATUS**  
18968 The following exit values shall be returned:  
18969 **0** One or more lines were selected.  
18970 **1** No lines were selected.  
18971 **>1** An error occurred.

## 18972 CONSEQUENCES OF ERRORS

18973 If the `-q` option is specified, the exit status shall be zero if an input line is selected, even if an  
 18974 error was detected. Otherwise, default actions shall be performed.

## 18975 APPLICATION USAGE

18976 Care should be taken when using characters in *pattern\_list* that may also be meaningful to the  
 18977 command interpreter. It is safest to enclose the entire *pattern\_list* argument in single quotes:

18978 '...'

18979 The `-e pattern_list` option has the same effect as the *pattern\_list* operand, but is useful when  
 18980 *pattern\_list* begins with the hyphen delimiter. It is also useful when it is more convenient to  
 18981 provide multiple patterns as separate arguments.

18982 Multiple `-e` and `-f` options are accepted and *grep* uses all of the patterns it is given while  
 18983 matching input text lines. (Note that the order of evaluation is not specified. If an  
 18984 implementation finds a null string as a pattern, it is allowed to use that pattern first, matching  
 18985 every line, and effectively ignore any other patterns.)

18986 The `-q` option provides a means of easily determining whether or not a pattern (or string) exists  
 18987 in a group of files. When searching several files, it provides a performance improvement  
 18988 (because it can quit as soon as it finds the first match) and requires less care by the user in  
 18989 choosing the set of files to supply as arguments (because it exits zero if it finds a match even if  
 18990 *grep* detected an access or read error on earlier *file* operands).

## 18991 EXAMPLES

18992 1. To find all uses of the word "Posix" (in any case) in file **text.mm** and write with line  
 18993 numbers:

18994 `grep -i -n posix text.mm`

18995 2. To find all empty lines in the standard input:

18996 `grep ^$`

18997 or:

18998 `grep -v .`

18999 3. Both of the following commands print all lines containing strings "abc" or "def" or both:

19000 `grep -E 'abc`

19001 `def'`

19002 `grep -F 'abc`

19003 `def'`

19004 4. Both of the following commands print all lines matching exactly "abc" or "def":

19005 `grep -E '^abc$`

19006 `^def$'`

19007 `grep -F -x 'abc`

19008 `def'`

## 19009 RATIONALE

19010 This *grep* has been enhanced in an upward-compatible way to provide the exact functionality of  
 19011 the historical *egrep* and *fgrep* commands as well. It was the clear intention of the standard  
 19012 developers to consolidate the three *greps* into a single command.



- 19013 The old *egrep* and *fgrep* commands are likely to be supported for many years to come as  
19014 implementation extensions, allowing historical applications to operate unmodified.
- 19015 Historical implementations usually silently ignored all but one of multiply-specified *-e* and *-f*  
19016 options, but were not consistent as to which specification was actually used.
- 19017 The *-b* option was omitted from the `OPTIONS` section because block numbers are  
19018 implementation-defined.
- 19019 The System V restriction on using *-* to mean standard input was omitted.
- 19020 A definition of action taken when given a null BRE or ERE is specified. This is an error condition  
19021 in some historical implementations.
- 19022 The *-I* option previously indicated that its use was undefined when no files were explicitly  
19023 named. This behavior was historical and placed an unnecessary restriction on future  
19024 implementations. It has been removed.
- 19025 The historical BSD *grep -s* option practice is easily duplicated by redirecting standard output to  
19026 `/dev/null`. The *-s* option required here is from System V.
- 19027 The *-x* option, historically available only with *fgrep*, is available here for all of the non-  
19028 obsolescent versions.
- 19029 **FUTURE DIRECTIONS**
- 19030 None.
- 19031 **SEE ALSO**
- 19032 *sed*
- 19033 **CHANGE HISTORY**
- 19034 First released in Issue 2.
- 19035 **Issue 4**
- 19036 Aligned with the ISO/IEC 9945-2:1993 standard.
- 19037 **Issue 6**
- 19038 The Open Group corrigenda item U029/5 has been applied, correcting the SYNOPSIS.
- 19039 The normative text is reworded to avoid use of the term “must” for application requirements.

19040 **NAME**

19041 hash — remember or report utility locations

19042 **SYNOPSIS**19043 xSI hash [*utility...*]

19044 hash -r

19045

19046 **DESCRIPTION**

19047 The *hash* utility shall affect the way the current shell environment remembers the locations of  
 19048 utilities found as described in Section 2.9.1.1 (on page 2257). Depending on the arguments  
 19049 specified, it shall add utility locations to its list of remembered locations or it shall purge the  
 19050 contents of the list. When no arguments are specified, it shall report on the contents of the list.

19051 Utilities provided as built-ins to the shell shall not be reported by *hash*.19052 **OPTIONS**

19053 The *hash* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 19054 12.2, Utility Syntax Guidelines.

19055 The following option shall be supported:

19056 -r Forget all previously remembered utility locations.

19057 **OPERANDS**

19058 The following operand shall be supported:

19059 *utility* The name of a utility to be searched for and added to the list of remembered  
 19060 locations. If *utility* contains one or more slashes, the results are unspecified.

19061 **STDIN**

19062 Not used.

19063 **INPUT FILES**

19064 None.

19065 **ENVIRONMENT VARIABLES**19066 The following environment variables shall affect the execution of *hash*:

19067 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 19068 If *LANG* is unset or null, the corresponding value from the implementation-  
 19069 defined default locale shall be used. If any of the internationalization variables  
 19070 contains an invalid setting, the utility shall behave as if none of the variables had  
 19071 been defined.

19072 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 19073 internationalization variables.

19074 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 19075 characters (for example, single-byte as opposed to multi-byte characters in  
 19076 arguments).

19077 *LC\_MESSAGES*

19078 Determine the locale that should be used to affect the format and contents of  
 19079 diagnostic messages written to standard error.

19080 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

19081 *PATH* Determine the location of *utility*, as described in the Base Definitions volume of  
 19082 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.

19083 **ASYNCHRONOUS EVENTS**

19084 Default.

19085 **STDOUT**

19086 The standard output of *hash* shall be used when no arguments are specified. Its format is  
19087 unspecified, but includes the path name of each utility in the list of remembered locations for the  
19088 current shell environment. This list shall consist of those utilities named in previous *hash*  
19089 invocations that have been invoked, and may contain those invoked and found through the  
19090 normal command search process.

19091 **STDERR**

19092 Used only for diagnostic messages.

19093 **OUTPUT FILES**

19094 None.

19095 **EXTENDED DESCRIPTION**

19096 None.

19097 **EXIT STATUS**

19098 The following exit values shall be returned:

19099 0 Successful completion.

19100 &gt;0 An error occurred.

19101 **CONSEQUENCES OF ERRORS**

19102 Default.

19103 **APPLICATION USAGE**

19104 Since *hash* affects the current shell execution environment, it is always provided as a shell  
19105 regular built-in. If it is called in a separate utility execution environment, such as one of the  
19106 following:

19107 `nohup hash -r`19108 `find . -type f | xargs hash`

19109 it does not affect the command search process of the caller's environment.

19110 The *hash* utility may be implemented as an alias—for example, *alias -t -*, in which case utilities  
19111 found through normal command search are not listed by the *hash* command.

19112 The effects of *hash -r* can also be achieved portably by resetting the value of *PATH*; in the  
19113 simplest form, this can be:

19114 `PATH="$PATH"`

19115 The use of *hash* with *utility* names is unnecessary for most applications, but may provide a  
19116 performance improvement on a few implementations; normally, the hashing process is included  
19117 by default.

19118 **EXAMPLES**

19119 None.

19120 **RATIONALE**

19121 None.

19122 **FUTURE DIRECTIONS**

19123 None.

19124 **SEE ALSO**

19125           Section 2.9.1.1 (on page 2257)

19126 **CHANGE HISTORY**

19127           First released in Issue 2.

19128 **Issue 4**

19129           Relocated from the *sh* description to reflect its status as a regular built-in utility.

19130 **NAME**

19131 head — copy the first part of files

19132 **SYNOPSIS**19133 head [-n *number*][*file...*]19134 **DESCRIPTION**19135 The *head* utility shall copy its input files to the standard output, ending the output for each file at  
19136 a designated point.19137 Copying shall end at the point in each input file indicated by the **-n *number*** option. The option-  
19138 argument *number* shall be counted in units of lines.19139 **OPTIONS**19140 The *head* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
19141 12.2, Utility Syntax Guidelines.

19142 The following option shall be supported:

19143 **-n *number*** The first *number* lines of each input file shall be copied to standard output. The  
19144 application shall ensure that the *number* option-argument is a positive decimal  
19145 integer.19146 If no options are specified, *head* shall act as if **-n 10** had been specified.19147 **OPERANDS**

19148 The following operand shall be supported:

19149 *file* A path name of an input file. If no *file* operands are specified, the standard input  
19150 shall be used.19151 **STDIN**19152 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES  
19153 section.19154 **INPUT FILES**

19155 Input files shall be text files, but the line length is not restricted to {LINE\_MAX} bytes.

19156 **ENVIRONMENT VARIABLES**19157 The following environment variables shall affect the execution of *head*:19158 *LANG* Provide a default value for the internationalization variables that are unset or null.  
19159 If *LANG* is unset or null, the corresponding value from the implementation-  
19160 defined default locale shall be used. If any of the internationalization variables  
19161 contains an invalid setting, the utility shall behave as if none of the variables had  
19162 been defined.19163 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
19164 internationalization variables.19165 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
19166 characters (for example, single-byte as opposed to multi-byte characters in  
19167 arguments and input files).19168 *LC\_MESSAGES*19169 Determine the locale that should be used to affect the format and contents of  
19170 diagnostic messages written to standard error.19171 *XSI* *NLS\_PATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

19172 **ASYNCHRONOUS EVENTS**

19173 Default.

19174 **STDOUT**

19175 The standard output shall contain designated portions of the input files.

19176 If multiple *file* operands are specified, *head* shall precede the output for each with the header:

19177 "\n==&gt; %s &lt;==\n", &lt;pathname&gt;

19178 except that the first header written shall not include the initial &lt;newline&gt; character.

19179 **STDERR**

19180 Used only for diagnostic messages.

19181 **OUTPUT FILES**

19182 None.

19183 **EXTENDED DESCRIPTION**

19184 None.

19185 **EXIT STATUS**

19186 The following exit values shall be returned:

19187 0 Successful completion.

19188 &gt;0 An error occurred.

19189 **CONSEQUENCES OF ERRORS**

19190 Default.

19191 **APPLICATION USAGE**19192 The obsolescent *-number* form is withdrawn in this version. Applications should use the *-n*  
19193 *number* option.19194 **EXAMPLES**

19195 To write the first ten lines of all files (except those with a leading period) in the directory:

19196 head \*

19197 **RATIONALE**19198 Although it is possible to simulate *head* with *sed 10q* for a single file, the standard developers  
19199 decided that the popularity of *head* on historical BSD systems warranted its inclusion alongside  
19200 *tail*.19201 This standard version of *head* follows the Utility Syntax Guidelines. The *-n* option was added to  
19202 this new interface so that *head* and *tail* would be more logically related.19203 There is no *-c* option (as there is in *tail*) because it is not historical practice and because other  
19204 utilities in this volume of IEEE Std. 1003.1-200x provide similar functionality.19205 **FUTURE DIRECTIONS**

19206 None.

19207 **SEE ALSO**19208 *sed*, *tail*19209 **CHANGE HISTORY**

19210 First released in Issue 4.

19211 **Issue 6**

19212 The obsolescent **–number** form is withdrawn.

19213 The normative text is reworded to avoid use of the term “must” for application requirements.

## 19214 NAME

19215 iconv — codeset conversion

## 19216 SYNOPSIS

19217 iconv [-cs] -f *fromcode* -t *tocode* [*file* ...]

19218 iconv -l

## 19219 DESCRIPTION

19220 The *iconv* utility shall convert the encoding of characters in *file* from one codeset to another and  
19221 write the results to standard output.19222 When the options indicate that charmap files are used to specify the codesets (see OPTIONS),  
19223 the codeset conversion shall be accomplished by performing a logical join on the symbolic  
19224 character names in the two charmaps. The implementation need not support the use of charmap  
19225 files for codeset conversion unless the POSIX2\_LOCALEDEF symbol is defined on the system.

## 19226 OPTIONS

19227 The *iconv* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
19228 12.2, Utility Syntax Guidelines.

19229 The following options shall be supported:

19230 **-c** Omit any invalid characters from the output. When **-c** is not used, the results of  
19231 encountering invalid characters in the input stream (either those that are not valid  
19232 members of the *fromcode* or those that have no corresponding value in *tocode*) shall  
19233 be specified in the system documentation. The presence or absence of **-c** shall not  
19234 affect the exit status of *iconv*.19235 **-f *fromcode*** Identify the codeset of the input file. If the option-argument contains a slash  
19236 character, *iconv* shall attempt to use it as the path name of a charmap file, as  
19237 defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 6.4,  
19238 Character Set Description File. If the path name does not represent a valid,  
19239 readable charmap file, the results are undefined. If the option-argument does not  
19240 contain a slash, it shall be considered the name of one of the codeset descriptions  
19241 provided by the system, in an unspecified format. The valid values of the option-  
19242 argument without a slash are implementation-defined. If this option is omitted, the  
19243 codeset of the current locale shall be used.19244 **-l** Write all supported *fromcode* and *tocode* values to standard output in an unspecified  
19245 format.19246 **-s** Suppress any messages written to standard error concerning invalid characters.  
19247 When **-s** is not used, the results of encountering invalid characters in the input  
19248 stream (either those that are not valid members of the *fromcode* or those that have  
19249 no corresponding value in *tocode*) shall be specified in the system documentation.  
19250 The presence or absence of **-s** shall not affect the exit status of *iconv*.19251 **-t *tocode*** Identify the codeset to be used for the output file. The semantics are equivalent to  
19252 the **-f *fromcode*** option.19253 If either **-f** or **-t** represents a charmap file, but the other does not (or is omitted), or both **-f** and  
19254 **-t** are omitted, the results are undefined.

## 19255 OPERANDS

19256 The following operand shall be supported:

19257 ***file*** A path name of an input file. If no *file* operands are specified, or if a *file* operand is  
19258 '-', the standard input shall be used.



19259 **STDIN**

19260 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'. |

19261 **INPUT FILES**

19262 The input file shall be a text file.

19263 **ENVIRONMENT VARIABLES**

19264 The following environment variables shall affect the execution of *iconv*:

19265 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 19266 If *LANG* is unset or null, the corresponding value from the implementation-  
 19267 defined default locale shall be used. If any of the internationalization variables  
 19268 contains an invalid setting, the utility shall behave as if none of the variables had  
 19269 been defined.

19270 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 19271 internationalization variables.

19272 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 19273 characters (for example, single-byte as opposed to multi-byte characters in  
 19274 arguments). During translation of the file, this variable is superseded by the use of  
 19275 the *fromcode* option-argument.

19276 *LC\_MESSAGES*

19277 Determine the locale that should be used to affect the format and contents of  
 19278 diagnostic messages written to standard error.

19279 *XSI* *NLS\_PATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

19280 **ASYNCHRONOUS EVENTS**

19281 Default.

19282 **STDOUT**

19283 When the *-I* option is used, the standard output shall contain all supported *fromcode* and *to*  
 19284 *code* values, written in an unspecified format.

19285 When the *-I* option is not used, the standard output shall contain the sequence of characters  
 19286 read from the input files, translated to the specified codeset. Nothing else shall be written to the  
 19287 standard output.

19288 **STDERR**

19289 Used only for diagnostic messages.

19290 **OUTPUT FILES**

19291 None.

19292 **EXTENDED DESCRIPTION**

19293 None.

19294 **EXIT STATUS**

19295 The following exit values shall be returned:

19296 0 Successful completion.

19297 >0 An error occurred.

19298 **CONSEQUENCES OF ERRORS**

19299 Default.

19300 **APPLICATION USAGE**

19301           The user must ensure that both charmap files use the same symbolic names for characters the  
19302           two codesets have in common.

19303 **EXAMPLES**

19304           The following example converts the contents of file **mail.x400** from the ISO/IEC 6937:1994  
19305           standard codeset to the ISO/IEC 8859-1:1998 standard codeset, and stores the results in file  
19306           **mail.local**:

```
19307 iconv -f IS6937 -t IS8859 mail.x400 > mail.local
```

19308 **RATIONALE**

19309           The *iconv* utility can be used portably only when the user provides two charmap files as option-  
19310           arguments. This is because a single charmap provided by the user cannot reliably be joined with  
19311           the names in a system-provided character set description. The valid values for *fromcode* and  
19312           *tocode* are implementation-defined and do not have to have any relation to the charmap  
19313           mechanisms. As an aid to interactive users, the **-I** option was adopted from the Plan 9 operating  
19314           system. It writes information concerning these implementation-defined values. The format is  
19315           unspecified because there are many possible useful formats that could be chosen, such as a  
19316           matrix of valid combinations of *fromcode* and *tocode*. The **-I** option is not intended for shell script  
19317           usage; portable applications will have to use charmaps.

19318 **FUTURE DIRECTIONS**

19319           None.

19320 **SEE ALSO**

19321           *gencat*

19322 **CHANGE HISTORY**

19323           First released in Issue 3.

19324 **Issue 4**

19325           Format reorganized.

19326           Utility Syntax Guidelines support mandated.

19327           Internationalized environment variable support mandated.

19328 **Issue 6**

19329           This utility has been rewritten to align with the IEEE P1003.2b draft standard. Specifically, the  
19330           ability to use charmap files for conversion has been added.

19331 **NAME**19332 `id` — return user identity19333 **SYNOPSIS**19334 `id` [*user*]19335 `id -G[-n]` [*user*]19336 `id -g[-nr]` [*user*]19337 `id -u[-nr]` [*user*]19338 **DESCRIPTION**

19339 If no *user* operand is provided, the *id* utility shall write the user and group IDs and the  
 19340 corresponding user and group names of the invoking process to standard output. If the effective  
 19341 and real IDs do not match, both shall be written. If multiple groups are supported by the  
 19342 underlying system (see the description of {NGROUPS\_MAX} in the System Interfaces volume of  
 19343 IEEE Std. 1003.1-200x), the supplementary group affiliations of the invoking process shall also be  
 19344 written.

19345 If a *user* operand is provided and the process has the appropriate privileges, the user and group  
 19346 IDs of the selected user shall be written. In this case, effective IDs shall be assumed to be  
 19347 identical to real IDs. If the selected user has more than one allowable group membership listed  
 19348 in the group database, these shall be written in the same manner as the supplementary groups  
 19349 described in the preceding paragraph.

19350 **OPTIONS**

19351 The *id* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 19352 Utility Syntax Guidelines.

19353 The following options shall be supported:

19354 **-G** Output all different group IDs (effective, real, and supplementary) only, using the  
 19355 format "`%u\n`". If there is more than one distinct group affiliation, output each  
 19356 such affiliation, using the format "`%u`", before the <newline> character is output.

19357 **-g** Output only the effective group ID, using the format "`%u\n`".

19358 **-n** Output the name in the format `%s` instead of the numeric ID using the format `%u`.

19359 **-r** Output the real ID instead of the effective ID.

19360 **-u** Output only the effective user ID, using the format "`%u\n`".

19361 **OPERANDS**

19362 The following operand shall be supported:

19363 *user* The login name for which information is to be written.

19364 **STDIN**

19365 Not used.

19366 **INPUT FILES**

19367 None.

19368 **ENVIRONMENT VARIABLES**

19369 The following environment variables shall affect the execution of *id*:

19370 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 19371 If *LANG* is unset or null, the corresponding value from the implementation-  
 19372 defined default locale shall be used. If any of the internationalization variables  
 19373 contains an invalid setting, the utility shall behave as if none of the variables had

- 19374 been defined.
- 19375 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
19376 internationalization variables.
- 19377 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
19378 characters (for example, single-byte as opposed to multi-byte characters in  
19379 arguments).
- 19380 *LC\_MESSAGES*  
19381 Determine the locale that should be used to affect the format and contents of  
19382 diagnostic messages written to standard error and informative messages written to  
19383 standard output.
- 19384 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 19385 **ASYNCHRONOUS EVENTS**  
19386 Default.
- 19387 **STDOUT**  
19388 The following formats shall be used when the *LC\_MESSAGES* locale category specifies the  
19389 POSIX locale. In other locales, the strings *uid*, *gid*, *eid*, *egid*, and *groups* may be replaced with  
19390 more appropriate strings corresponding to the locale.
- 19391 "uid=%u(%s) gid=%u(%s)\n", <real user ID>, <user-name>,  
19392 <real group ID>, <group-name>
- 19393 If the effective and real user IDs do not match, the following shall be inserted immediately  
19394 before the '\n' character in the previous format:
- 19395 " eid=%u(%s) "
- 19396 with the following arguments added at the end of the argument list:
- 19397 "effective user ID", <effective user-name>
- 19398 If the effective and real group IDs do not match, the following shall be inserted directly before  
19399 the '\n' character in the format string (and after any addition resulting from the effective and  
19400 real user IDs not matching):
- 19401 " egid=%u(%s) "
- 19402 with the following arguments added at the end of the argument list:
- 19403 <effective group-ID>, <effective group name>
- 19404 If the process has supplementary group affiliations or the selected user is allowed to belong to  
19405 multiple groups, the first shall be added directly before the <newline> character in the format  
19406 string:
- 19407 " groups=%u(%s) "
- 19408 with the following arguments added at the end of the argument list:
- 19409 <supplementary group ID>, <supplementary group name>
- 19410 and the necessary number of the following added after that for any remaining supplementary  
19411 group IDs:
- 19412 " ,%u(%s) "
- 19413 and the necessary number of the following arguments added at the end of the argument list:

19414 <supplementary group ID>, <supplementary group name>

19415 If any of the user ID, group ID, effective user ID, effective group ID, or supplementary/multiple  
 19416 group IDs cannot be mapped by the system into printable user or group names, the  
 19417 corresponding (%s) and name argument is omitted from the corresponding format string.

19418 When any of the options are specified, the output format shall be as described in the OPTIONS  
 19419 section.

19420 **STDERR**

19421 Used only for diagnostic messages.

19422 **OUTPUT FILES**

19423 None.

19424 **EXTENDED DESCRIPTION**

19425 None.

19426 **EXIT STATUS**

19427 The following exit values shall be returned:

19428 0 Successful completion.

19429 >0 An error occurred.

19430 **CONSEQUENCES OF ERRORS**

19431 Default.

19432 **APPLICATION USAGE**

19433 Output produced by the **-G** option and by the default case could potentially produce very long  
 19434 lines on systems that support large numbers of supplementary groups. (On systems with user  
 19435 and group IDs that are 32-bit integers and with group names with a maximum of 8 bytes per  
 19436 name, 93 supplementary groups plus distinct effective and real group and user IDs could  
 19437 theoretically overflow the 2 048-byte {LINE\_MAX} text file line limit on the default output case.  
 19438 It would take about 186 supplementary groups to overflow the 2 048-byte barrier using *id -G*).  
 19439 This is not expected to be a problem in practice, but in cases where it is a concern, applications  
 19440 should consider using *fold -s* before postprocessing the output of *id*.

19441 **EXAMPLES**

19442 None.

19443 **RATIONALE**

19444 The functionality provided by the 4 BSD *groups* utility can be simulated using:

19445 `id -Gn [ user ]`

19446 The 4 BSD command *groups* was considered, but it was not included because it did not provide  
 19447 the functionality of the *id* utility of the SVID. Also, it was thought that it would be easier to  
 19448 modify *id* to provide the additional functionality necessary to systems with multiple groups  
 19449 than to invent another command.

19450 The options **-u**, **-g**, **-n**, and **-r** were added to ease the use of *id* with shell commands  
 19451 substitution. Without these options it is necessary to use some preprocessor such as *sed* to select  
 19452 the desired piece of information. Since output such as that produced by:

19453 `id -u -n`

19454 is frequently wanted, it seemed desirable to add the options.

19455 **FUTURE DIRECTIONS**

19456 None.

19457 **SEE ALSO**19458 *fold, logname, who*, the System Interfaces volume of IEEE Std. 1003.1-200x, *getgid()*, *getgroups()*,  
19459 *getuid()*19460 **CHANGE HISTORY**

19461 First released in Issue 2.

19462 **Issue 4**

19463 Aligned with the ISO/IEC 9945-2:1993 standard.

19464 **NAME**

19465 ipcrm — remove an XSI message queue, semaphore set, or shared memory segment identifier

19466 **SYNOPSIS**

```
19467 xsi ipcrm [-q msgid | -Q msgkey | -s semid | -S semkey |
19468 -m shmid | -M shmkey] ...
```

19469

19470 **DESCRIPTION**

19471 The *ipcrm* utility shall remove zero or more message queues, semaphore sets, or shared memory  
 19472 segments. The interprocess communication facilities to be removed are specified by the options.

19473 Only a user with appropriate privilege shall be allowed to remove an interprocess  
 19474 communication facility that was not created by or owned by the user invoking *ipcrm*.

19475 **OPTIONS**

19476 The *ipcrm* facility supports the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 19477 Utility Syntax Guidelines.

19478 The following options shall be supported:

19479 **-q msgid** Remove the message queue identifier *msgid* from the system and destroy the  
 19480 message queue and data structure associated with it.

19481 **-m shmid** Remove the shared memory identifier *shmid* from the system. The shared memory  
 19482 segment and data structure associated with it shall be destroyed after the last  
 19483 detach.

19484 **-s semid** Remove the semaphore identifier *semid* from the system and destroy the set of  
 19485 semaphores and data structure associated with it.

19486 **-Q msgkey** Remove the message queue identifier, created with key *msgkey*, from the system  
 19487 and destroy the message queue and data structure associated with it.

19488 **-M shmkey** Remove the shared memory identifier, created with key *shmkey*, from the system.  
 19489 The shared memory segment and data structure associated with it shall be  
 19490 destroyed after the last detach.

19491 **-S semkey** Remove the semaphore identifier, created with key *semkey*, from the system and  
 19492 destroy the set of semaphores and data structure associated with it.

19493 **OPERANDS**

19494 None.

19495 **STDIN**

19496 Not used.

19497 **INPUT FILES**

19498 None.

19499 **ENVIRONMENT VARIABLES**

19500 The following environment variables shall affect the execution of *ipcrm*:

19501 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 19502 If *LANG* is unset or null, the corresponding value from the implementation-  
 19503 defined default locale shall be used. If any of the internationalization variables  
 19504 contain an invalid setting, the utility behaves as if none of the variables had been  
 19505 set.

19506 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 19507 internationalization variables.

- 19508            *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
19509            characters (for example, single-byte as opposed to multi-byte characters in  
19510            arguments).
- 19511            *LC\_MESSAGES*  
19512                    Determine the locale that should be used to affect the format and contents of  
19513            diagnostic messages written to standard error.
- 19514            *NLSPATH*  
19515                    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 19516 **ASYNCHRONOUS EVENTS**
- 19517            Default.
- 19518 **STDOUT**
- 19519            Not used.
- 19520 **STDERR**
- 19521            Used only for diagnostic messages.
- 19522 **OUTPUT FILES**
- 19523            None.
- 19524 **EXTENDED DESCRIPTION**
- 19525            None.
- 19526 **EXIT STATUS**
- 19527            The following exit values shall be returned:
- 19528            0   Successful completion.
- 19529            >0   An error occurred.
- 19530 **CONSEQUENCES OF ERRORS**
- 19531            Default.
- 19532 **APPLICATION USAGE**
- 19533            None.
- 19534 **EXAMPLES**
- 19535            None.
- 19536 **RATIONALE**
- 19537            None.
- 19538 **FUTURE DIRECTIONS**
- 19539            None.
- 19540 **SEE ALSO**
- 19541            *ipcs*, the System Interfaces volume of IEEE Std. 1003.1-200x, *msgctl()*, *semctl()*, *shmctl()*
- 19542 **CHANGE HISTORY**
- 19543            First released in Issue 5.



19544 **NAME**19545 `ipcs` — report XSI interprocess communication facilities status19546 **SYNOPSIS**19547 XSI `ipcs [-qms] [-a | -bcopt]`

19548

19549 **DESCRIPTION**19550 The *ipcs* utility shall write information about active interprocess communication facilities.

19551 Without options, information shall be written in short format for message queues, shared  
 19552 memory segments, and semaphores sets that are currently active in the system. Otherwise, the  
 19553 information that is displayed is controlled by the options specified.

19554 **OPTIONS**

19555 The *ipcs* facility supports the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 19556 Utility Syntax Guidelines.

19557 The *ipcs* utility accepts the following options:19558 **-q** Write information about active message queues.19559 **-m** Write information about active shared memory segments.19560 **-s** Write information about active semaphores sets.

19561 If **-q**, **-m**, or **-s** are specified, only information about those facilities shall be written. If none of  
 19562 these three are specified, information about all three shall be written subject to the following  
 19563 options:

19564 **-a** Use all print options. (This is a shorthand notation for **-b**, **-c**, **-o**, **-p**, and **-t**.)

19565 **-b** Write information on maximum allowable size. (Maximum number of bytes in  
 19566 messages on queue for message queues, size of segments for shared memory, and  
 19567 number of semaphores in each set for semaphores.)

19568 **-c** Write creator's user name and group name; see below.

19569 **-o** Write information on outstanding usage. (Number of messages on queue and total  
 19570 number of bytes in messages on queue for message queues, and number of  
 19571 processes attached to shared memory segments.)

19572 **-p** Write process number information. (Process ID of last process to send a message  
 19573 and process ID of last process to receive a message on message queues, process ID  
 19574 of creating process, and process ID of last process to attach or detach on shared  
 19575 memory segments.)

19576 **-t** Write time information. (Time of the last control operation that changed the access  
 19577 permissions for all facilities, time of last *msgsnd()* and *msgrcv()* operations on  
 19578 message queues, time of last *shmat()* and *shmdt()* operations on shared memory,  
 19579 and time of last *semop()* operation on semaphores.)

19580 **OPERANDS**

19581 None.

19582 **STDIN**

19583 Not used.

19584 **INPUT FILES**

- 19585           • The group database
- 19586           • The user database

19587 **ENVIRONMENT VARIABLES**

19588           The following environment variables shall affect the execution of *ipcs*:

- 19589           *LANG*       Provide a default value for the internationalization variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contain an invalid setting, the utility behaves as if none of the variables had been set.
- 19590
- 19591
- 19592
- 19593
- 19594           *LC\_ALL*       If set to a non-empty string value, override the values of all the other internationalization variables.
- 19595
- 19596           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).
- 19597
- 19598
- 19599           *LC\_MESSAGES*   Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
- 19600
- 19601
- 19602           *NLSPATH*     Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 19603           *TZ*           Determine the timezone for the time strings written by *ipcs*.

19604 **ASYNCHRONOUS EVENTS**

19605           Default.

19606 **STDOUT**

19607           An introductory line shall be written with the format:

19608           "IPC status from %s as of %s\n", <source>, <date>

19609           where <source> indicates the source used to gather the statistics and <date> is the information that would be produced by the *date* command when invoked in the POSIX locale.

19610

19611           The *ipcs* utility then shall create up to three reports depending upon the *-q*, *-m*, and *-s* options. The first report shall indicate the status of message queues, the second report shall indicate the status of shared memory segments, and the third report shall indicate the status of semaphore sets.

19612

19613

19614

19615           If the corresponding facility is not installed or has not been used since the last reboot, then the report shall be written out in the format:

19616

19617           "%s facility not in system.\n", <facility>

19618           where <facility> is *Message Queue*, *Shared Memory*, or *Semaphore*, as appropriate. If the facility has been installed and has been used since the last reboot, column headings separated by one or more spaces and followed by a <newline> shall be written as indicated below followed by the facility name written out using the format:

19619

19620

19621

19622           "%s:\n", <facility>

19623           where <facility> is *Message Queues*, *Shared Memory*, or *Semaphores*, as appropriate. On the second and third reports the column headings need not be written if the last column headings written already provide column headings for all information in that report.

19624

19625

19626 The column headings provided in the first column below and the meaning of the information in  
 19627 those columns shall be given in order below; the letters in parentheses indicate the options that  
 19628 shall cause the corresponding column to appear; “all” means that the column shall always  
 19629 appear. Each column is separated by one or more <space> characters. Note that these options  
 19630 only determine what information is provided for each report; they do not determine which  
 19631 reports are written.

|       |      |              |                                                                                                     |
|-------|------|--------------|-----------------------------------------------------------------------------------------------------|
| 19632 | T    | (all)        | Type of facility:                                                                                   |
| 19633 |      | <b>q</b>     | Message queue.                                                                                      |
| 19634 |      | <b>m</b>     | Shared memory segment.                                                                              |
| 19635 |      | <b>s</b>     | Semaphore.                                                                                          |
| 19636 |      |              | This field is a single character written using the format %c.                                       |
| 19637 | ID   | (all)        | The identifier for the facility entry. This field shall be written using the format                 |
| 19638 |      |              | %d.                                                                                                 |
| 19639 | KEY  | (all)        | The key used as an argument to <i>msgget()</i> , <i>semget()</i> , or <i>shmget()</i> to create the |
| 19640 |      |              | facility entry.                                                                                     |
| 19641 |      | <b>Note:</b> | The key of a shared memory segment is changed to IPC_PRIVATE                                        |
| 19642 |      |              | when the segment has been removed until all processes attached                                      |
| 19643 |      |              | to the segment detach it.                                                                           |
| 19644 |      |              | This field shall be written using the format 0x%x.                                                  |
| 19645 | MODE | (all)        | The facility access modes and flags. The mode shall consist of 11 characters                        |
| 19646 |      |              | that are interpreted as follows.                                                                    |
| 19647 |      |              | The first character shall be:                                                                       |
| 19648 |      | S            | If a process is waiting on a <i>msgsnd()</i> operation.                                             |
| 19649 |      | –            | If the above is not true.                                                                           |
| 19650 |      |              | The second character shall be:                                                                      |
| 19651 |      | R            | If a process is waiting on a <i>msgrcv()</i> operation.                                             |
| 19652 |      | C or –       | If the associated shared memory segment is to be cleared when the                                   |
| 19653 |      |              | first attach operation is executed.                                                                 |
| 19654 |      | –            | If none of the above is true.                                                                       |
| 19655 |      |              | The next nine characters shall be interpreted as three sets of three bits each.                     |
| 19656 |      |              | The first set refers to the owner’s permissions; the next to permissions of                         |
| 19657 |      |              | others in the usergroup of the facility entry; and the last to all others. Within                   |
| 19658 |      |              | each set, the first character indicates permission to read, the second character                    |
| 19659 |      |              | indicates permission to write or alter the facility entry, and the last character is                |
| 19660 |      |              | a minus sign (‘-’).                                                                                 |
| 19661 |      |              | The permissions shall be indicated as follows:                                                      |
| 19662 |      | r            | If read permission is granted.                                                                      |
| 19663 |      | w            | If write permission is granted.                                                                     |
| 19664 |      | a            | If alter permission is granted.                                                                     |
| 19665 |      | –            | If the indicated permission is not granted.                                                         |

|       |         |       |                                                                                                                                                                                                                                                                                                                                               |
|-------|---------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19666 |         |       | The first character following the permissions specifies if there is an alternate or additional access control method associated with the facility. If there is no alternate or additional access control method associated with the facility, a single <space> character shall be written; otherwise, another printable character is written. |
| 19667 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19668 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19669 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19670 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19671 | OWNER   | (all) | The user name of the owner of the facility entry. If the user name of the owner is found in the user database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the user ID of the owner shall be written using the format %d.                                                          |
| 19672 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19673 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19674 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19675 | GROUP   | (all) | The group name of the owner of the facility entry. If the group name of the owner is found in the group database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the group ID of the owner shall be written using the format %d.                                                      |
| 19676 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19677 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19678 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19679 |         |       | The following nine columns shall be only written out for message queues:                                                                                                                                                                                                                                                                      |
| 19680 | CREATOR | (a,c) | The user name of the creator of the facility entry. If the user name of the creator is found in the user database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the user ID of the creator shall be written using the format %d.                                                    |
| 19681 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19682 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19683 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19684 | CGROUP  | (a,c) | The group name of the creator of the facility entry. If the group name of the creator is found in the group database, at least the first eight column positions of the name shall be written using the format %s. Otherwise, the group ID of the creator shall be written using the format %d.                                                |
| 19685 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19686 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19687 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19688 | CBYTES  | (a,o) | The number of bytes in messages currently outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                                                       |
| 19689 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19690 | QNUM    | (a,o) | The number of messages currently outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                                                                |
| 19691 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19692 | QBYTES  | (a,b) | The maximum number of bytes allowed in messages outstanding on the associated message queue. This field shall be written using the format %d.                                                                                                                                                                                                 |
| 19693 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19694 | LSPID   | (a,p) | The process ID of the last process to send a message to the associated queue. This field shall be written using the format:                                                                                                                                                                                                                   |
| 19695 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19696 |         |       | "%d", <pid>                                                                                                                                                                                                                                                                                                                                   |
| 19697 |         |       | where <pid> is 0 if no message has been sent to the corresponding message queue; otherwise, <pid> shall be the process ID of the last process to send a message to the queue.                                                                                                                                                                 |
| 19698 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19699 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19700 | LRPID   | (a,p) | The process ID of the last process to receive a message from the associated queue. This field shall be written using the format:                                                                                                                                                                                                              |
| 19701 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19702 |         |       | "%d", <pid>                                                                                                                                                                                                                                                                                                                                   |
| 19703 |         |       | where <pid> is 0 if no message has been received from the corresponding message queue; otherwise, <pid> shall be the process ID of the last process to receive a message from the queue.                                                                                                                                                      |
| 19704 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19705 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19706 | STIME   | (a,t) | The time the last message was sent to the associated queue. If a message has been sent to the corresponding message queue, the hour, minute, and second of the last time a message was sent to the queue shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format " no-entry" shall be written.                                |
| 19707 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19708 |         |       |                                                                                                                                                                                                                                                                                                                                               |
| 19709 |         |       |                                                                                                                                                                                                                                                                                                                                               |

|       |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
|-------|-----------------------------------------------------------------------------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19710 | RTIME                                                                             | (a,t) | The time the last message was received from the associated queue. If a message has been received from the corresponding message queue, the hour, minute, and second of the last time a message was received from the queue shall be written using the format <code>%d:%2.2d:%2.2d</code> . Otherwise, the format <code>" no-entry"</code> shall be written. |
| 19711 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19712 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19713 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19714 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19715 | The following eight columns shall be only written out for shared memory segments. |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19716 | CREATOR                                                                           | (a,c) | The user of the creator of the facility entry. If the user name of the creator is found in the user database, at least the first eight column positions of the name shall be written using the format <code>%s</code> . Otherwise, the user ID of the creator shall be written using the format <code>%d</code> .                                           |
| 19717 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19718 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19719 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19720 | CGROUP                                                                            | (a,c) | The group name of the creator of the facility entry. If the group name of the creator is found in the group database, at least the first eight column positions of the name shall be written using the format <code>%s</code> . Otherwise, the group ID of the creator shall be written using the format <code>%d</code> .                                  |
| 19721 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19722 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19723 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19724 | NATTCH                                                                            | (a,o) | The number of processes attached to the associated shared memory segment. This field shall be written using the format <code>%d</code> .                                                                                                                                                                                                                    |
| 19725 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19726 | SEGSZ                                                                             | (a,b) | The size of the associated shared memory segment. This field shall be written using the format <code>%d</code> .                                                                                                                                                                                                                                            |
| 19727 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19728 | CPID                                                                              | (a,p) | The process ID of the creator of the shared memory entry. This field shall be written using the format <code>%d</code> .                                                                                                                                                                                                                                    |
| 19729 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19730 | LPID                                                                              | (a,p) | The process ID of the last process to attach or detach the shared memory segment. This field shall be written using the format:                                                                                                                                                                                                                             |
| 19731 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19732 |                                                                                   |       | <code>"%d", &lt;pid&gt;</code>                                                                                                                                                                                                                                                                                                                              |
| 19733 |                                                                                   |       | where <code>&lt;pid&gt;</code> is 0 if no process has attached the corresponding shared memory segment; otherwise, <code>&lt;pid&gt;</code> shall be the process ID of the last process to attach or detach the segment.                                                                                                                                    |
| 19734 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19735 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19736 | ATIME                                                                             | (a,t) | The time the last attach on the associated shared memory segment was completed. If the corresponding shared memory segment has ever been attached, the hour, minute, and second of the last time the segment was attached shall be written using the format <code>%d:%2.2d:%2.2d</code> . Otherwise, the format <code>" no-entry"</code> shall be written.  |
| 19737 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19738 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19739 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19740 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19741 | DTIME                                                                             | (a,t) | The time the last detach on the associated shared memory segment was completed. If the corresponding shared memory segment has ever been detached, the hour, minute, and second of the last time the segment was detached shall be written using the format <code>%d:%2.2d:%2.2d</code> . Otherwise, the format <code>" no-entry"</code> shall be written.  |
| 19742 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19743 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19744 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19745 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19746 | The following four columns shall be only written out for semaphore sets:          |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19747 | CREATOR                                                                           | (a,c) | The user of the creator of the facility entry. If the user name of the creator is found in the user database, at least the first eight column positions of the name shall be written using the format <code>%s</code> . Otherwise, the user ID of the creator shall be written using the format <code>%d</code> .                                           |
| 19748 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19749 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19750 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19751 | CGROUP                                                                            | (a,c) | The group name of the creator of the facility entry. If the group name of the creator is found in the group database, at least the first eight column positions of the name shall be written using the format <code>%s</code> . Otherwise, the group ID of the creator shall be written using the format <code>%d</code> .                                  |
| 19752 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19753 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |
| 19754 |                                                                                   |       |                                                                                                                                                                                                                                                                                                                                                             |

- 19755 NSEMS (a,b) The number of semaphores in the set associated with the semaphore entry.  
19756 This field shall be written using the format %d.
- 19757 OTIME (a,t) The time the last semaphore operation on the set associated with the  
19758 semaphore entry was completed. If a semaphore operation has ever been  
19759 performed on the corresponding semaphore set, the hour, minute, and second  
19760 of the last semaphore operation on the semaphore set shall be written using  
19761 the format %d:%2.2d:%2.2d. Otherwise, the format " no-entry" shall be  
19762 written.
- 19763 The following column shall be written for all three reports when it is requested:
- 19764 CTIME (a,t) The time the associated entry was created or changed. The hour, minute, and  
19765 second of the time when the associated entry was created shall be written  
19766 using the format %d:%2.2d:%2.2d.
- 19767 **STDERR**  
19768 Used only for diagnostic messages.
- 19769 **OUTPUT FILES**  
19770 None.
- 19771 **EXTENDED DESCRIPTION**  
19772 None.
- 19773 **EXIT STATUS**  
19774 The following exit values shall be returned:  
19775 0 Successful completion.  
19776 >0 An error occurred.
- 19777 **CONSEQUENCES OF ERRORS**  
19778 Default.
- 19779 **APPLICATION USAGE**  
19780 Things can change while *ipcs* is running; the information it gives is guaranteed to be accurate  
19781 only when it was retrieved.
- 19782 **EXAMPLES**  
19783 None.
- 19784 **RATIONALE**  
19785 None.
- 19786 **FUTURE DIRECTIONS**  
19787 None.
- 19788 **SEE ALSO**  
19789 The System Interfaces volume of IEEE Std. 1003.1-200x, *msgop()*, *msgrcv()*, *msgsnd()*, *semget()*,  
19790 *semop()*, *shmat()*, *shmdt()*, *shmget()*, *shmop()*
- 19791 **CHANGE HISTORY**  
19792 First released in Issue 5.
- 19793 **Issue 6**  
19794 The Open Group corrigenda item U020/1 has been applied, correcting the SYNOPSIS.  
19795 The Open Group corrigenda items U032/1 and U032/2 have been applied, clarifying the output  
19796 format.

19797

The Open Group Base Resolution bwg98-004 is applied.

19798 **NAME**

19799        jobs — display status of jobs in the current session

19800 **SYNOPSIS**19801 UP        jobs [-l | -p][*job\_id*...]

19802

19803 **DESCRIPTION**19804        The *jobs* utility shall display the status of jobs that were started in the current shell environment; see Section 2.13 (on page 2273).19806        When *jobs* reports the termination status of a job, the shell shall remove its process ID from the list of those “known in the current shell execution environment”; see Section 2.9.3.1 (on page 2259).19809 **OPTIONS**19810        The *jobs* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

19812        The following options shall be supported:

19813        -**l**           (The letter ell.) Provide more information about each job listed. This information shall include the job number, current job, process group ID, state, and the command that formed the job.19816        -**p**           Display only the process IDs for the process group leaders of the selected jobs.19817        By default, the *jobs* utility shall display the status of all stopped jobs, running background jobs and all jobs whose status has changed and have not been reported by the shell.19819 **OPERANDS**

19820        The following operand shall be supported:

19821        *job\_id*       Specifies the jobs for which the status is to be displayed. If no *job\_id* is given, the status information for all jobs shall be displayed. The format of *job\_id* is described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.205, Job Control Job ID.19825 **STDIN**

19826        Not used.

19827 **INPUT FILES**

19828        None.

19829 **ENVIRONMENT VARIABLES**19830        The following environment variables shall affect the execution of *jobs*:19831        **LANG**        Provide a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined.19836        **LC\_ALL**       If set to a non-empty string value, override the values of all the other internationalization variables.19838        **LC\_CTYPE**     Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).



- 19841 *LC\_MESSAGES*
- 19842 Determine the locale that should be used to affect the format and contents of
- 19843 diagnostic messages written to standard error and informative messages written to
- 19844 standard output.
- 19845 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 19846 **ASYNCHRONOUS EVENTS**
- 19847 Default.
- 19848 **STDOUT**
- 19849 If the **-p** option is specified, the output shall consist of one line for each process ID:
- 19850 "%d\n", <process ID>
- 19851 Otherwise, if the **-l** option is not specified, the output shall be a series of lines of the form:
- 19852 "[%d] %c %s %s\n", <job-number>, <current>, <state>, <command>
- 19853 where the fields shall be as follows:
- 19854 <current> The character '+' identifies the job that would be used as a default for the *fg* or *bg*
- 19855 utilities; this job can also be specified using the *job\_id* %+ or "%%". The character
- 19856 '-' identifies the job that would become the default if the current default job were to
- 19857 exit; this job can also be specified using the *job\_id* %-. For other jobs, this field is
- 19858 a <space> character. At most one job can be identified with '+' and at most one
- 19859 job can be identified with '-'. If there is any suspended job, then the current job
- 19860 shall be a suspended job. If there are at least two suspended jobs, then the previous
- 19861 job also shall be a suspended job.
- 19862 <job-number> A number that can be used to identify the process group to the *wait*, *fg*, *bg*, and *kill*
- 19863 utilities. Using these utilities, the job can be identified by prefixing the job number
- 19864 with '% '.
- 19865 <state> One of the following strings (in the POSIX locale):
- 19866 **Running** Indicates that the job has not been suspended by a signal and has not
- 19867 exited.
- 19868 **Done** Indicates that the job completed and returned exit status zero.
- 19869 **Done(code)** Indicates that the job completed normally and that it exited with the
- 19870 specified non-zero exit status, *code*, expressed as a decimal number.
- 19871 **Stopped** Indicates that the job was suspended by the SIGTSTP signal.
- 19872 **Stopped (SIGTSTP)**
- 19873 Indicates that the job was suspended by the SIGTSTP signal.
- 19874 **Stopped (SIGSTOP)**
- 19875 Indicates that the job was suspended by the SIGSTOP signal.
- 19876 **Stopped (SIGTTIN)**
- 19877 Indicates that the job was suspended by the SIGTTIN signal.
- 19878 **Stopped (SIGTTOU)**
- 19879 Indicates that the job was suspended by the SIGTTOU signal.
- 19880 The implementation may substitute the string **Suspended** in place of **Stopped**. If
- 19881 the job was terminated by a signal, the format of <state> is unspecified, but it shall
- 19882 be visibly distinct from all of the other <state> formats shown here and shall
- 19883 indicate the name or description of the signal causing the termination.

- 19884           <*command*> The associated command that was given to the shell.
- 19885           If the **-I** option is specified, a field containing the process group ID shall be inserted before the
- 19886           <*state*> field. Also, more processes in a process group may be output on separate lines, using
- 19887           only the process ID and <*command*> fields.
- 19888 **STDERR**
- 19889           Used only for diagnostic messages.
- 19890 **OUTPUT FILES**
- 19891           None.
- 19892 **EXTENDED DESCRIPTION**
- 19893           None.
- 19894 **EXIT STATUS**
- 19895           The following exit values shall be returned:
- 19896           0 Successful completion.
- 19897           >0 An error occurred.
- 19898 **CONSEQUENCES OF ERRORS**
- 19899           Default.
- 19900 **APPLICATION USAGE**
- 19901           The **-p** option is the only portable way to find out the process group of a job because different
- 19902           implementations have different strategies for defining the process group of the job. Usage such
- 19903           as  $\$(jobs -p)$  provides a way of referring to the process group of the job in an implementation-
- 19904           independent way.
- 19905           The *jobs* utility does not work as expected when it is operating in its own utility execution
- 19906           environment because that environment has no applicable jobs to manipulate. See the
- 19907           APPLICATION USAGE section for *bg* (on page 2422). For this reason, *jobs* is generally
- 19908           implemented as a shell regular built-in.
- 19909 **EXAMPLES**
- 19910           None.
- 19911 **RATIONALE**
- 19912           Both "%%" and "%+" are used to refer to the current job. Both forms are of equal validity—the
- 19913           "%%" mirroring "\$\$" and "%+" mirroring the output of *jobs*. Both forms reflect historical
- 19914           practice of the KornShell and the C shell with job control.
- 19915           The job control features provided by *bg*, *fg*, and *jobs* are based on the KornShell. The standard
- 19916           developers examined the characteristics of the C shell versions of these utilities and found that
- 19917           differences exist. Despite widespread use of the C shell, the KornShell versions were selected for
- 19918           this volume of IEEE Std. 1003.1-200x to maintain a degree of uniformity with the rest of the
- 19919           KornShell features selected (such as the very popular command line editing features).
- 19920           The *jobs* utility is not dependent on the job control option, as are the seemingly related *bg* and *fg*
- 19921           utilities because *jobs* is useful for examining background jobs, regardless of the condition of job
- 19922           control. When the user has invoked a *set +m* command and job control has been turned off, *jobs*
- 19923           can still be used to examine the background jobs associated with that current session. Similarly,
- 19924           *kill* can then be used to kill background jobs with *kill% <background job number>*.
- 19925           The output for terminated jobs is left unspecified to accommodate various historical systems.
- 19926           The following formats have been witnessed:

- 19927           1. **Killed**(*signal name*)
- 19928           2. *signal name*
- 19929           3. *signal name*(**coredump**)
- 19930           4. *signal description*– **core dumped**
- 19931           Most users should be able to understand these formats, although it means that applications have  
19932           trouble parsing them.
- 19933           The calculation of job IDs was not described since this would suggest an implementation, which  
19934           may impose unnecessary restrictions.
- 19935           In an early proposal, a **-n** option was included to “Display the status of jobs that have changed,  
19936           exited, or stopped since the last status report”. It was removed because the shell always writes  
19937           any changed status of jobs before each prompt.
- 19938 **FUTURE DIRECTIONS**
- 19939           None.
- 19940 **SEE ALSO**
- 19941           *bg, fg, kill, wait*
- 19942 **CHANGE HISTORY**
- 19943           First released in Issue 4.
- 19944 **Issue 6**
- 19945           This utility is now marked as part of the User Portability Utilities option.
- 19946           The JC shading is removed as job control is mandatory in this issue.

19947 **NAME**

19948 join — relational database operator

19949 **SYNOPSIS**

```
19950 join [-a file_number | -v file_number][-e string][-o list][-t char]
19951 [-1 field][-2 field] file1 file2
```

19952 **DESCRIPTION**

19953 The *join* utility shall perform an equality join on the files *file1* and *file2*. The joined files shall be  
 19954 written to the standard output.

19955 The join field is a field in each file on which the files are compared. By default, *join* shall write  
 19956 one line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The  
 19957 output line by default shall consist of the join field, then the remaining fields from *file1*, then the  
 19958 remaining fields from *file2*. This format can be changed by using the **-o** option (see below). The  
 19959 **-a** option can be used to add unmatched lines to the output. The **-v** option can be used to output  
 19960 only unmatched lines.

19961 **Notes to Reviewers**19962 *This section with side shading will not appear in the final copy. - Ed.*

19963 D1, XCU, ERN 265 proposes to add the following text here: "If the same key appears more than  
 19964 once in either file, all possible pairwise combinations are output, in unspecified order".

19965 By default, the files *file1* and *file2* should be ordered in the collating sequence of *sort -b* on the  
 19966 fields on which they shall be joined, by default the first in each line. All selected output shall be  
 19967 written in the same collating sequence.

19968 The default input field separators shall be <blank> characters. In this case, multiple separators  
 19969 shall count as one field separator, and leading separators shall be ignored. The default output  
 19970 field separator shall be a <space> character.

19971 The field separator and collating sequence can be changed by using the **-t** option (see below).

19972 If the input files are not in the appropriate collating sequence, the results are unspecified.

19973 **OPTIONS**

19974 The *join* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 19975 12.2, Utility Syntax Guidelines.

19976 The following options shall be supported:

19977 **-a file\_number**

19978 Produce a line for each unpairable line in file *file\_number*, where *file\_number* is 1 or  
 19979 2, in addition to the default output. If both **-a1** and **-a2** are specified, all unpairable  
 19980 lines shall be output.

19981 **-e string** Replace empty output fields in the list selected by **-o** with the string *string*.

19982 **-o list** Construct the output line to comprise the fields specified in *list*, each element of  
 19983 which shall have one of the following two forms:

- 19984 1. *file\_number.field*, where *file\_number* is a file number and *field* is a decimal  
 19985 integer field number
- 19986 2. 0 (zero), representing the join field

19987 The elements of *list* shall be either comma-separated or <blank>-separated, as  
 19988 specified in Guideline 8 of the Base Definitions volume of IEEE Std. 1003.1-200x,  
 19989 Section 12.2, Utility Syntax Guidelines. The fields specified by *list* shall be written

- 19990 for all selected output lines. Fields selected by *list* that do not appear in the input  
 19991 shall be treated as empty output fields. (See the `-e` option.) Only specifically  
 19992 requested fields shall be written. The application shall ensure that *list* is a single  
 19993 command line argument.
- 19994 `-t char` Use character *char* as a separator, for both input and output. Every appearance of  
 19995 *char* in a line shall be significant. When this option is specified, the collating  
 19996 sequence should be the same as *sort* without the `-b` option.
- 19997 `-v file_number`  
 19998 Instead of the default output, produce a line only for each unpairable line in  
 19999 *file\_number*, where *file\_number* is 1 or 2. If both `-v1` and `-v2` are specified, all  
 20000 unpairable lines shall be output.
- 20001 `-1 field` Join on the *field*th field of file 1. Fields are decimal integers starting with 1.  
 20002 `-2 field` Join on the *field*th field of file 2. Fields are decimal integers starting with 1.
- 20003 **OPERANDS**  
 20004 The following operands shall be supported:
- 20005 *file1, file2*  
 20006 A path name of a file to be joined. If either of the *file1* or *file2* operands is '-', the  
 20007 standard input shall be used in its place.
- 20008 **STDIN**  
 20009 The standard input shall be used only if the *file1* or *file2* operand is '-'. See the INPUT FILES  
 20010 section.
- 20011 **INPUT FILES**  
 20012 The input files shall be text files.
- 20013 **ENVIRONMENT VARIABLES**  
 20014 The following environment variables shall affect the execution of *join*:
- 20015 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 20016 If *LANG* is unset or null, the corresponding value from the implementation-  
 20017 defined default locale shall be used. If any of the internationalization variables  
 20018 contains an invalid setting, the utility shall behave as if none of the variables had  
 20019 been defined.
- 20020 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 20021 internationalization variables.
- 20022 *LC\_COLLATE*  
 20023 Determine the locale of the collating sequence *join* expects to have been used when  
 20024 the input files were sorted.
- 20025 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 20026 characters (for example, single-byte as opposed to multi-byte characters in  
 20027 arguments and input files).
- 20028 *LC\_MESSAGES*  
 20029 Determine the locale that should be used to affect the format and contents of  
 20030 diagnostic messages written to standard error.
- 20031 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

20032 **ASYNCHRONOUS EVENTS**

20033 Default.

20034 **STDOUT**20035 The *join* utility output shall be a concatenation of selected character fields. When the **-o** option  
20036 is not specified, the output shall be:20037 "%s%s%s\n", <join field>, <other file1 fields>,  
20038 <other file2 fields>

20039 If the join field is not the first field in a file, the &lt;other file fields&gt; for that file shall be:

20040 &lt;fields preceding join field&gt;, &lt;fields following join field&gt;

20041 When the **-o** option is specified, the output format shall be:

20042 "%s\n", &lt;concatenation of fields&gt;

20043 where the concatenation of fields is described by the **-o** option, above.20044 For either format, each field (except the last) shall be written with its trailing separator character.  
20045 If the separator is the default (<blank> characters), a single <space> character shall be written  
20046 after each field (except the last).20047 **STDERR**

20048 Used only for diagnostic messages.

20049 **OUTPUT FILES**

20050 None.

20051 **EXTENDED DESCRIPTION**

20052 None.

20053 **EXIT STATUS**

20054 The following exit values shall be returned:

20055 0 All input files were output successfully.

20056 &gt;0 An error occurred.

20057 **CONSEQUENCES OF ERRORS**

20058 Default.

20059 **APPLICATION USAGE**20060 Path names consisting of numeric digits or of the form *string.string* should not be specified  
20061 directly following the **-o** list.20062 **EXAMPLES**20063 The **-o 0** field essentially selects the union of the join fields. For example, given file **phone**:

| 20064 | !Name   | Phone Number    |
|-------|---------|-----------------|
| 20065 | Don     | +1 123-456-7890 |
| 20066 | Hal     | +1 234-567-8901 |
| 20067 | Yasushi | +2 345-678-9012 |

20068 and file **fax**:

| 20069 | !Name   | Fax Number      |
|-------|---------|-----------------|
| 20070 | Don     | +1 123-456-7899 |
| 20071 | Keith   | +1 456-789-0122 |
| 20072 | Yasushi | +2 345-678-9011 |

20073 (where the large expanses of white space are meant to each represent a single <tab> character),  
20074 the command:

```
20075 join -t "<tab>" -a 1 -a 2 -e '(unknown)' -o 0,1.2,2.2 phone fax
```

20076 would produce:

| 20077 | !Name   | Phone Number    | Fax Number      |
|-------|---------|-----------------|-----------------|
| 20078 | Don     | +1 123-456-7890 | +1 123-456-7899 |
| 20079 | Hal     | +1 234-567-8901 | (unknown)       |
| 20080 | Keith   | (unknown)       | +1 456-789-0122 |
| 20081 | Yasushi | +2 345-678-9012 | +2 345-678-9011 |

## 20082 **Notes to Reviewers**

20083 *This section with side shading will not appear in the final copy. - Ed.*

20084 D1, XCU, ERN 265 proposes to add the following example.

20085 The following:

20086 fa:

20087       a x

20088       a y

20089       a z

20090 fb:

20091       a p

20092       a q

20093 would produce:

20094       a x p

20095       a x q

20096       a y p

20097       a y q

20098       a z p

20099       a z q

## 20100 **RATIONALE**

20101 The `-e` option is only effective when used with `-o` because, unless specific fields are identified  
20102 using `-o`, *join* is not aware of what fields might be empty. The exception to this is the join field,  
20103 but identifying an empty join field with the `-e` string is not historical practice and some scripts  
20104 might break if this were changed.

20105 The 0 field in the `-o` list was adopted from the Tenth Edition version of *join* to satisfy  
20106 international objections that the *join* in the base documents do not support the “full join” or  
20107 “outer join” described in relational database literature. Although it has been possible to include  
20108 a join field in the output (by default, or by field number using `-o`), the join field could not be  
20109 included for an unpaired line selected by `-a`. The `-o 0` field essentially selects the union of the  
20110 join fields.

20111 This sort of outer join was not possible with the *join* commands in the base documents. The `-o 0`  
20112 field was chosen because it is an upward-compatible change for applications. An alternative was  
20113 considered: have the join field represent the union of the fields in the files (where they are  
20114 identical for matched lines, and one or both are null for unmatched lines). This was not adopted  
20115 because it would break some historical applications.

20116 The ability to specify *file2* as `-` is not historical practice; it was added for completeness.

- 20117 The `-v` option is not historical practice, but was considered necessary because it permitted the  
20118 writing of *only* those lines that do not match on the join field, as opposed to the `-a` option, which  
20119 prints both lines that do and do not match. This additional facility is parallel with the `-v` option  
20120 of *grep*.
- 20121 Some historical implementations have been encountered where a blank line in one of the input  
20122 files was considered to be the end of the file; the description in this volume of  
20123 IEEE Std. 1003.1-200x does not cite this as an allowable case.
- 20124 **FUTURE DIRECTIONS**
- 20125 None.
- 20126 **SEE ALSO**
- 20127 *awk, comm, sort, uniq*
- 20128 **CHANGE HISTORY**
- 20129 First released in Issue 2.
- 20130 **Issue 4**
- 20131 Aligned with the ISO/IEC 9945-2:1993 standard.
- 20132 **Issue 6**
- 20133 The obsolescent `-j` options and the multi-argument `-o` option are withdrawn in this issue.
- 20134 The normative text is reworded to avoid use of the term “must” for application requirements.



## 20135 NAME

20136 kill — terminate or signal processes

## 20137 SYNOPSIS

20138 kill -s *signal\_name* *pid*...

20139 kill -l [*exit\_status*]

## 20140 DESCRIPTION

20141 The *kill* utility shall send a signal to the process or processes specified by each *pid* operand.

20142 For each *pid* operand, the *kill* utility shall perform actions equivalent to the *kill()* function  
20143 defined in the System Interfaces volume of IEEE Std. 1003.1-200x called with the following  
20144 arguments:

- 20145 • The value of the *pid* operand shall be used as the *pid* argument.
- 20146 • The *sig* argument is the value specified by the *-s* option, *-signal\_number* option, or the  
20147 *-signal\_name* option, or by SIGTERM, if none of these options is specified.

## 20148 OPTIONS

20149 The *kill* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
20150 12.2, Utility Syntax Guidelines.

20151 The following options shall be supported:

20152 **-l** (The letter ell.) Write all values of *signal\_name* supported by the implementation, if  
20153 no operand is given. If an *exit\_status* operand is given and it is a value of the ' ? '   
20154 shell special parameter (see Section 2.5.2 (on page 2241) and *wait* (on page 3254))  
20155 corresponding to a process that was terminated by a signal, the *signal\_name*  
20156 corresponding to the signal that terminated the process shall be written. If an  
20157 *exit\_status* operand is given and it is the unsigned decimal integer value of a signal  
20158 number, the *signal\_name* (the symbolic constant name without the **SIG** prefix  
20159 defined in the Base Definitions volume of IEEE Std. 1003.1-200x) corresponding to  
20160 that signal shall be written. Otherwise, the results are unspecified.

20161 **-s *signal\_name***

20162 Specify the signal to send, using one of the symbolic names defined in the  
20163 <**signal.h**> header defined in the Base Definitions volume of IEEE Std. 1003.1-200x,  
20164 Chapter 13, Headers. Values of *signal\_name* shall be recognized in a case-  
20165 independent fashion, without the **SIG** prefix. In addition, the symbolic name 0  
20166 shall be recognized, representing the signal value zero. The corresponding signal  
20167 shall be sent instead of SIGTERM.

## 20168 OPERANDS

20169 The following operands shall be supported:

20170 *pid* One of the following:

- 20171 1. A decimal integer specifying a process or process group to be signaled. The  
20172 process or processes selected by positive, negative and zero values of the *pid*  
20173 operand shall be as described for the *kill()* function defined in the System  
20174 Interfaces volume of IEEE Std. 1003.1-200x. If process number 0 is specified,  
20175 all processes in the current process group are signaled. For the effects of  
20176 negative *pid* numbers, see the *kill()* function defined in the System Interfaces  
20177 volume of IEEE Std. 1003.1-200x. If the first *pid* operand is negative, it should  
20178 be preceded by "—" to keep it from being interpreted as an option.

- 20179                   2. A job control job ID (see the Base Definitions volume of  
20180                   IEEE Std. 1003.1-200x, Section 3.205, Job Control Job ID) that identifies a  
20181                   background process group to be signaled. The job control job ID notation is  
20182                   applicable only for invocations of *kill* in the current shell execution  
20183                   environment; see Section 2.13 (on page 2273).
- 20184           *exit\_status*   A decimal integer specifying a signal number or the exit status of a process  
20185                   terminated by a signal.
- 20186 **STDIN**  
20187           Not used.
- 20188 **INPUT FILES**  
20189           None.
- 20190 **ENVIRONMENT VARIABLES**  
20191           The following environment variables shall affect the execution of *kill*:
- 20192           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
20193                   If *LANG* is unset or null, the corresponding value from the implementation-  
20194                   defined default locale shall be used. If any of the internationalization variables  
20195                   contains an invalid setting, the utility shall behave as if none of the variables had  
20196                   been defined.
- 20197           *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
20198                   internationalization variables.
- 20199           *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
20200                   characters (for example, single-byte as opposed to multi-byte characters in  
20201                   arguments).
- 20202           *LC\_MESSAGES*  
20203                   Determine the locale that should be used to affect the format and contents of  
20204                   diagnostic messages written to standard error.
- 20205 XSI        *NLSPATH*       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 20206 **ASYNCHRONOUS EVENTS**  
20207           Default.
- 20208 **STDOUT**  
20209           When the **-I** option is not specified, the standard output shall not be used.
- 20210           When the **-I** option is specified, the symbolic name of each signal shall be written in the  
20211           following format:
- 20212           "%s%c", <signal\_name>, <separator>
- 20213           where the <signal\_name> is in uppercase, without the **SIG** prefix, and the <separator> shall be  
20214           either a <newline> character or a <space> character. For the last signal written, <separator> shall  
20215           be a <newline> character.
- 20216           When both the **-I** option and *exit\_status* operand are specified, the symbolic name of the  
20217           corresponding signal shall be written in the following format:
- 20218           "%s\n", <signal\_name>

20219 **STDERR**

20220           Used only for diagnostic messages.

20221 **OUTPUT FILES**

20222           None.

20223 **EXTENDED DESCRIPTION**

20224           None.

20225 **EXIT STATUS**

20226           The following exit values shall be returned:

20227            0   At least one matching process was found for each *pid* operand, and the specified signal was  
20228                successfully processed for at least one matching process.

20229           >0   An error occurred.

20230 **CONSEQUENCES OF ERRORS**

20231           Default.

20232 **APPLICATION USAGE**

20233           Process numbers can be found by using *ps*.

20234           The job control job ID notation is not required to work as expected when *kill* is operating in its  
20235           own utility execution environment. In either of the following examples:

```
20236 nohup kill %1 &
20237 system("kill %1");
```

20238           the *kill* operates in a different environment and does not share the shell's understanding of job  
20239           numbers.

20240 **EXAMPLES**

20241           Any of the commands:

```
20242 kill -s kill 100 -165
20243 kill -s KILL 100 -165
```

20244           sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose  
20245           process group ID is 165, assuming the sending process has permission to send that signal to the  
20246           specified processes, and that they exist.

20247           The System Interfaces volume of IEEE Std. 1003.1-200x and this volume of IEEE Std. 1003.1-200x  
20248           do not require specific signal numbers for any *signal\_names*. Even the *-signal\_number* option  
20249           provides symbolic (although numeric) names for signals. If a process is terminated by a signal,  
20250           its exit status indicates the signal that killed it, but the exact values are not specified. The *kill -l*  
20251           option, however, can be used to map decimal signal numbers and exit status values into the  
20252           name of a signal. The following example reports the status of a terminated job:

```
20253 job
20254 stat=$?
20255 if [$stat -eq 0]
20256 then
20257 echo job completed successfully.
20258 elif [$stat -gt 128]
20259 then
20260 echo job terminated by signal SIG$(kill -l $stat).
20261 else
20262 echo job terminated with error code $stat.
20263 fi
```

20264 To avoid an ambiguity of an initial negative number argument specifying either a signal number  
20265 or a process group, the ISO/IEC 9945-2: 1993 standard mandates that it always be considered the  
20266 former. Therefore, to send the default signal to a process group (say 123), an application should  
20267 use a command similar to one of the following:

20268 `kill -TERM -123`

20269 `kill -- -123`

#### 20270 RATIONALE

20271 The `-l` option originated from the C shell, and is also implemented in the KornShell. The C shell  
20272 output can consist of multiple output lines because the signal names do not always fit on a  
20273 single line on some terminal screens. The KornShell output also included the implementation-  
20274 defined signal numbers and was considered by the standard developers to be too difficult for  
20275 scripts to parse conveniently. The specified output format is intended not only to accommodate  
20276 the historical C shell output, but also to permit an entirely vertical or entirely horizontal listing  
20277 on systems for which this is appropriate.

20278 An early proposal invented the name `SIGNULL` as a *signal\_name* for signal 0 (used by the System  
20279 Interfaces volume of IEEE Std. 1003.1-200x to test for the existence of a process without sending  
20280 it a signal). Since the *signal\_name* 0 can be used in this case unambiguously, `SIGNULL` has been  
20281 removed.

20282 An early proposal also required symbolic *signal\_names* to be recognized with or without the **SIG**  
20283 prefix. Historical versions of *kill* have not written the **SIG** prefix for the `-l` option and have not  
20284 recognized the **SIG** prefix on *signal\_names*. Since neither applications portability nor ease-of-use  
20285 would be improved by requiring this extension, it is no longer required.

20286 This volume of IEEE Std. 1003.1-200x contains no utility that browses for process IDs. Values for  
20287 *pid* are available via the `'!'` and `'$'` parameters of the shell command language.

20288 The `-s` option was added in response to international interest in providing some form of *kill* that  
20289 meets the Utility Syntax Guidelines.

20290 The job control job ID notation is not required to work as expected when *kill* is operating in its  
20291 own utility execution environment. In either of the following examples:

```
20292 nohup kill %1 &
20293 system("kill %1");
```

20294 the *kill* operates in a different environment and does not understand how the shell has managed  
20295 its job numbers.

#### 20296 FUTURE DIRECTIONS

20297 None.

#### 20298 SEE ALSO

20299 *ps*, *wait*, the System Interfaces volume of IEEE Std. 1003.1-200x, *kill()*, `<signal.h>`

#### 20300 CHANGE HISTORY

20301 First released in Issue 2.

#### 20302 Issue 4

20303 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 20304 Issue 6

20305 The obsolescent versions of the SYNOPSIS are withdrawn in this issue.

## 20306 NAME

20307 `lex` — generate programs for lexical tasks (**DEVELOPMENT**)

## 20308 SYNOPSIS

20309 CD `lex -c [-t][-n] -v[file ...]`

20310

## 20311 DESCRIPTION

20312 The *lex* utility shall generate C programs to be used in lexical processing of character input, and  
 20313 that can be used as an interface to *yacc*. The C programs shall be generated from *lex* source code  
 20314 and conform to the ISO C standard. Usually, the *lex* utility shall write the program it generates to  
 20315 the file `lex.yy.c`; the state of this file is unspecified if *lex* exits with a non-zero exit status. See the  
 20316 EXTENDED DESCRIPTION section for a complete description of the *lex* input language.

## 20317 OPTIONS

20318 The *lex* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 20319 12.2, Utility Syntax Guidelines.

20320 The following options shall be supported:

20321 `-n` Suppress the summary of statistics usually written with the `-v` option. If no table  
 20322 sizes are specified in the *lex* source code and the `-v` option is not specified, then `-n`  
 20323 is implied.

20324 `-t` Write the resulting program to standard output instead of `lex.yy.c`.

20325 `-v` Write a summary of *lex* statistics to the standard output. (See the discussion of *lex*  
 20326 table sizes in **Definitions in lex** (on page 2745).) If the `-t` option is specified and  
 20327 `-n` is not specified, this report shall be written to standard error. If table sizes are  
 20328 specified in the *lex* source code, and if the `-n` option is not specified, the `-v` option  
 20329 may be enabled.

## 20330 OPERANDS

20331 The following operand shall be supported:

20332 *file* A path name of an input file. If more than one such *file* is specified, all files shall be  
 20333 concatenated to produce a single *lex* program. If no *file* operands are specified, or if  
 20334 a *file* operand is `'-'`, the standard input shall be used.

## 20335 STDIN

20336 The standard input shall be used if no *file* operands are specified, or if a *file* operand is `'-'`. See  
 20337 INPUT FILES.

## 20338 INPUT FILES

20339 The input files shall be text files containing *lex* source code, as described in the EXTENDED  
 20340 DESCRIPTION section.

## 20341 ENVIRONMENT VARIABLES

20342 If this variable is not set to the POSIX locale, the results are unspecified.

20343 The following environment variables shall affect the execution of *lex*:

20344 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 20345 If *LANG* is unset or null, the corresponding value from the implementation-  
 20346 defined default locale shall be used. If any of the internationalization variables  
 20347 contains an invalid setting, the utility shall behave as if none of the variables had  
 20348 been defined.

20349 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 20350 internationalization variables.

- 20351 *LC\_COLLATE*  
 20352 Determine the locale for the behavior of ranges, equivalence classes and multi-  
 20353 character collating elements within regular expressions. If this variable is not set to  
 20354 the POSIX locale, the results are unspecified.
- 20355 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 20356 characters (for example, single-byte as opposed to multi-byte characters in  
 20357 arguments and input files), and the behavior of character classes within regular  
 20358 expressions. If this variable is not set to the POSIX locale, the results are  
 20359 unspecified.
- 20360 *LC\_MESSAGES*  
 20361 Determine the locale that should be used to affect the format and contents of  
 20362 diagnostic messages written to standard error.
- 20363 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 20364 **ASYNCHRONOUS EVENTS**  
 20365 Default.
- 20366 **STDOUT**  
 20367 If the `-t` option is specified, the text file of C source code output of *lex* shall be written to  
 20368 standard output.
- 20369 If the `-t` option is not specified:
- 20370 • Implementation-defined informational, error, and warning messages concerning the contents  
 20371 of *lex* source code input shall be written to either the standard output or standard error.
  - 20372 • If the `-v` option is specified and the `-n` option is not specified, *lex* statistics shall also be  
 20373 written to either the standard output or standard error, in an implementation-defined format.  
 20374 These statistics may also be generated if table sizes are specified with a '%' operator in the  
 20375 *Definitions* section, as long as the `-n` option is not specified.
- 20376 **STDERR**  
 20377 If the `-t` option is specified, implementation-defined informational, error, and warning messages  
 20378 concerning the contents of *lex* source code input shall be written to the standard error.
- 20379 If the `-t` option is not specified:
- 20380 1. Implementation-defined informational, error, and warning messages concerning the  
 20381 contents of *lex* source code input shall be written to either the standard output or standard  
 20382 error.
  - 20383 2. If the `-v` option is specified and the `-n` option is not specified, *lex* statistics shall also be  
 20384 written to either the standard output or standard error, in an implementation-defined  
 20385 format. These statistics may also be generated if table sizes are specified with a '%' operator  
 20386 in the *Definitions* section, as long as the `-n` option is not specified.
- 20387 **OUTPUT FILES**  
 20388 A text file containing C source code shall be written to `lex.yy.c`, or to the standard output if the  
 20389 `-t` option is present.
- 20390 **EXTENDED DESCRIPTION**  
 20391 Each input file contains *lex* source code, which is a table of regular expressions with  
 20392 corresponding actions in the form of C program fragments.
- 20393 When `lex.yy.c` is compiled and linked with the *lex* library (using the `-ll` operand with *c99* or *cc*),  
 20394 the resulting program reads character input from the standard input and partitions it into strings  
 20395 that match the given expressions.

20396 When an expression is matched, these actions shall occur:

- 20397 • The input string that was matched is left in *yytext* as a null-terminated string; *yytext* is either
- 20398 an external character array or a pointer to a character string. As explained in **Definitions in**
- 20399 **lex**, the type can be explicitly selected using the **%array** or **%pointer** declarations, but the
- 20400 default is implementation-defined.
- 20401 • The external **int** *yylen* is set to the length of the matching string.
- 20402 • The expression's corresponding program fragment, or action, is executed.

20403 During pattern matching, *lex* shall search the set of patterns for the single longest possible

20404 match. Among rules that match the same number of characters, the rule given first shall be

20405 chosen.

20406 The general format of *lex* source shall be:

20407       *Definitions*

20408       %%

20409       *Rules*

20410       %%

20411       *UserSubroutines*

20412 The first "%%" is required to mark the beginning of the rules (regular expressions and actions);

20413 the second "%%" is required only if user subroutines follow.

20414 Any line in the *Definitions* section beginning with a <blank> character shall be assumed to be a C

20415 program fragment and shall be copied to the external definition area of the **lex.yy.c** file.

20416 Similarly, anything in the *Definitions* section included between delimiter lines containing only

20417 "%{" and "%}" shall also be copied unchanged to the external definition area of the **lex.yy.c** file.

20418 Any such input (beginning with a <blank> character or within "%{" and "%}" delimiter lines)

20419 appearing at the beginning of the *Rules* section before any rules are specified shall be written to

20420 **lex.yy.c** after the declarations of variables for the *yylex* function and before the first line of code

20421 in *yylex*. Thus, user variables local to *yylex* can be declared here, as well as application code to

20422 execute upon entry to *yylex*.

20423 The action taken by *lex* when encountering any input beginning with a <blank> character or

20424 within "%{" and "%}" delimiter lines appearing in the *Rules* section but coming after one or

20425 more rules is undefined. The presence of such input may result in an erroneous definition of the

20426 *yylex* function.

20427 **Definitions in lex**

20428 *Definitions* appear before the first "%%" delimiter. Any line in this section not contained between

20429 "%{" and "%}" lines and not beginning with a <blank> character shall be assumed to define a

20430 *lex* substitution string. The format of these lines shall be:

20431       *name substitute*

20432 If a *name* does not meet the requirements for identifiers in the ISO C standard, the result is

20433 undefined. The string *substitute* shall replace the string *{name}* when it is used in a rule. The *name*

20434 string shall be recognized in this context only when the braces are provided and when it does

20435 not appear within a bracket expression or within double-quotes.

20436 In the *Definitions* section, any line beginning with a '%' (percent sign) character and followed by

20437 an alphanumeric word beginning with either 's' or 'S' shall define a set of start conditions.

20438 Any line beginning with a '%' followed by a word beginning with either 'x' or 'X' shall define

20439 a set of exclusive start conditions. When the generated scanner is in a "%s" state, patterns with

20440 no state specified shall be also active; in a "%x" state, such patterns shall not be active. The rest  
 20441 of the line, after the first word, shall be considered to be one or more <blank> character-  
 20442 separated names of start conditions. Start condition names shall be constructed in the same way  
 20443 as definition names. Start conditions can be used to restrict the matching of regular expressions  
 20444 to one or more states as described in **Regular Expressions in lex** (on page 2747).

20445 Implementations shall accept either of the following two mutually exclusive declarations in the  
 20446 *Definitions* section:

20447 **%array**     Declare the type of *yytext* to be a null-terminated character array.

20448 **%pointer**   Declare the type of *yytext* to be a pointer to a null-terminated character string.

20449 The default type of *yytext* is implementation-defined. If an application refers to *yytext* outside of  
 20450 the scanner source file (that is, via an **extern**), the application shall include the appropriate  
 20451 **%array** or **%pointer** declaration in the scanner source file.

20452 Implementations shall accept declarations in the *Definitions* section for setting certain internal  
 20453 table sizes. The declarations are shown in the following table.

20454 **Table 4-9** Table Size Declarations in *lex*

| Declaration        | Description                        | Minimum Value |
|--------------------|------------------------------------|---------------|
| <b>%p</b> <i>n</i> | Number of positions                | 2 500         |
| <b>%n</b> <i>n</i> | Number of states                   | 500           |
| <b>%a</b> <i>n</i> | Number of transitions              | 2 000         |
| <b>%e</b> <i>n</i> | Number of parse tree nodes         | 1 000         |
| <b>%k</b> <i>n</i> | Number of packed character classes | 1 000         |
| <b>%o</b> <i>n</i> | Size of the output array           | 3 000         |

20462 In the table, *n* represents a positive decimal integer, preceded by one or more <blank>  
 20463 characters. The exact meaning of these table size numbers is implementation-defined. The  
 20464 implementation shall document how these numbers affect the *lex* utility and how they are  
 20465 related to any output that may be generated by the implementation should space limitations be  
 20466 encountered during the execution of *lex*. It shall be possible to determine from this output which  
 20467 of the table size values needs to be modified to permit *lex* to successfully generate tables for the  
 20468 input language. The values in the column Minimum Value represent the lowest values  
 20469 conforming implementations shall provide.

## 20470 **Rules in lex**

20471 The rules in *lex* source files are a table in which the left column contains regular expressions and  
 20472 the right column contains actions (C program fragments) to be executed when the expressions  
 20473 are recognized.

20474 *ERE action*

20475 *ERE action*

20476 ...

20477 The extended regular expression (*ERE*) portion of a row shall be separated from *action* by one or  
 20478 more <blank> characters. A regular expression containing <blank> characters shall be  
 20479 recognized under one of the following conditions:

- 20480 • The entire expression appears within double-quotes.
- 20481 • The <blank> characters appear within double-quotes or square brackets.



- 20482           • Each <blank> character is preceded by a backslash character.

20483           **User Subroutines in lex**

20484           Anything in the user subroutines section shall be copied to **lex.yy.c** following *yylex*.

20485           **Regular Expressions in lex**

20486           The *lex* utility shall support the set of extended regular expressions (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.4, Extended Regular Expressions), with the following additions and exceptions to the syntax:

20489           ". . ."       Any string enclosed in double-quotes shall represent the characters within the double-quotes as themselves, except that backslash escapes (which appear in the following table) shall be recognized. Any backslash-escape sequence shall be terminated by the closing quote. For example, "\01" "1" represents a single string: the octal value 1 followed by the character '1'.

20494           <state>*r*, <state1, state2, . . .>*r*

20495           The regular expression *r* shall be matched only when the program is in one of the start conditions indicated by *state*, *state1*, and so on; see **Actions in lex** (on page 2749). (As an exception to the typographical conventions of the rest of this volume of IEEE Std. 1003.1-200x, in this case <state> does not represent a metavariable, but the literal angle-bracket characters surrounding a symbol.) The start condition shall be recognized as such only at the beginning of a regular expression.

20501           *r/x*

20502           The regular expression *r* shall be matched only if it is followed by an occurrence of regular expression *x* (*x* is the instance of trailing context, further defined below). The token returned in *yytext* shall only match *r*. If the trailing portion of *r* matches the beginning of *x*, the result is unspecified. The *r* expression cannot include further trailing context or the '\$' (match-end-of-line) operator; *x* cannot include the '^' (match-beginning-of-line) operator, nor trailing context, nor the '\$' operator. That is, only one occurrence of trailing context is allowed in a *lex* regular expression, and the '^' operator only can be used at the beginning of such an expression.

20510           {*name*}

20511           When *name* is one of the substitution symbols from the *Definitions* section, the string, including the enclosing braces, shall be replaced by the *substitute* value. The *substitute* value shall be treated in the extended regular expression as if it were enclosed in parentheses. No substitution shall occur if {*name*} occurs within a bracket expression or within double-quotes.

20515           Within an ERE, a backslash character shall be considered to begin an escape sequence as specified in the table in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'). In addition, the escape sequences in the following table shall be recognized.

20519           A literal <newline> character cannot occur within an ERE; the escape sequence '\n' can be used to represent a <newline> character. A <newline> character shall not be matched by a period operator.

20522

Table 4-10 Escape Sequences in *lex*

20523

20524

20525

20526

20527

20528

20529

20530

20531

20532

20533

20534

20535

20536

20537

20538

20539

20540

20541

20542

20543

20544

20545

20546

20547

20548

20549

20550

20551

20552

| Escape Sequence       | Description                                                                                                                                                                                                                                                                                                                                                   | Meaning                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\digits</code>  | A backslash character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.                                                                                                                                         | The character whose encoding is represented by the one, two, or three-digit octal integer. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation-defined. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading ' <code>\</code> ' for each byte. |
| <code>\xdigits</code> | A backslash character followed by the longest sequence of hexadecimal-digit characters (01234567abcdefABCDEF). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.                                                                                                                                          | The character whose encoding is represented by the hexadecimal integer.                                                                                                                                                                                                                                                                                                             |
| <code>\c</code>       | A backslash character followed by any character not described in this table or in the table in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation (' <code>\</code> ', ' <code>\a</code> ', ' <code>\b</code> ', ' <code>\f</code> ', ' <code>\n</code> ', ' <code>\r</code> ', ' <code>\t</code> ', ' <code>\v</code> '). | The character ' <code>c</code> ', unchanged.                                                                                                                                                                                                                                                                                                                                        |

20553 **Notes to Reviewers**

20554

*This section with side shading will not appear in the final copy. - Ed.*

20555

20556

20557

D3, XCU, ERN 120, re length limitation for hex, suggests adding a note: "Note: If a hexadecimal escape sequence which is followed by a hexadecimal digit is required, either the character in hex or the following character may be parenthesized using `\(` and `\)`."

20558

20559

20560

20561

The order of precedence given to extended regular expressions for *lex* differs from that specified in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.4, Extended Regular Expressions. The order of precedence for *lex* shall be as shown in the following table, from high to low.

20562

20563

20564

20565

20566

**Note:** The escaped characters entry is not meant to imply that these are operators, but they are included in the table to show their relationships to the true operators. The start condition, trailing context, and anchoring notations have been omitted from the table because of the placement restrictions described in this section; they can only appear at the beginning or ending of an ERE.

20567

Table 4-11 ERE Precedence in *lex*

20568

20569

20570

20571

20572

20573

20574

20575

20576

20577

20578

| Extended Regular Expression              | Precedence            |
|------------------------------------------|-----------------------|
| <i>collation-related bracket symbols</i> | [ = ] [ : : ] [ . . ] |
| <i>escaped characters</i>                | \<special character>  |
| <i>bracket expression</i>                | [ ]                   |
| <i>quoting</i>                           | " . . . "             |
| <i>grouping</i>                          | ( )                   |
| <i>definition</i>                        | { name }              |
| <i>single-character RE duplication</i>   | * + ?                 |
| <i>concatenation</i>                     |                       |
| <i>interval expression</i>               | { m, n }              |
| <i>alternation</i>                       |                       |

20579

20580

20581

20582

20583

20584

20585

20586

The ERE anchoring operators '*^*' and '*\$*' do not appear in the table. With *lex* regular expressions, these operators are restricted in their use: the '*^*' operator can only be used at the beginning of an entire regular expression, and the '*\$*' operator only at the end. The operators apply to the entire regular expression. Thus, for example, the pattern "*(^abc)|(def\$)*" is undefined; it can instead be written as two separate rules, one with the regular expression "*^abc*" and one with "*def\$*", which share a common action via the special '*|*' action (see below). If the pattern were written "*^abc|def\$*", it would match either "*abc*" or "*def*" on a line by itself.

20587

20588

20589

20590

Unlike the general ERE rules, embedded anchoring is not allowed by most historical *lex* implementations. An example of embedded anchoring would be for patterns such as "*(^| )foo( |\$)*" to match "*foo*" when it exists as a complete word. This functionality can be obtained using existing *lex* features:

20591

20592

```
^foo/[\n] |
" foo"/[\n] /* Found foo as a separate word. */
```

20593

20594

20595

Note also that '*\$*' is a form of trailing context (it is equivalent to "*/\n*") and as such cannot be used with regular expressions containing another instance of the operator (see the preceding discussion of trailing context).

20596

20597

20598

20599

20600

The additional regular expressions trailing-context operator '*/'*' can be used as an ordinary character if presented within double-quotes, "*/"*"; preceded by a backslash, "*\"*"; or within a bracket expression, "*[ / ]*". The start-condition '*<*' and '*>*' operators shall be special only in a start condition at the beginning of a regular expression; elsewhere in the regular expression they shall be treated as ordinary characters.

20601

### Actions in *lex*

20602

20603

20604

20605

20606

20607

The action to be taken when an ERE is matched can be a C program fragment or the special actions described below; the program fragment can contain one or more C statements, and can also include special actions. The empty C statement '*;*' shall be a valid action; any string in the *lex.yy.c* input that matches the pattern portion of such a rule is effectively ignored or skipped. However, the absence of an action shall not be valid, and the action *lex* takes in such a condition is undefined.

20608

20609

The specification for an action, including C statements and special actions, can extend across several lines if enclosed in braces:

20610

20611

```
ERE <one or more blanks> { program statement
 program statement }
```

20612 The default action when a string in the input to a **lex.yy.c** program is not matched by any  
 20613 expression shall be to copy the string to the output. Because the default behavior of a program  
 20614 generated by *lex* is to read the input and copy it to the output, a minimal *lex* source program that  
 20615 has just "%%" shall generate a C program that simply copies the input to the output unchanged.

20616 Four special actions shall be available:

20617 | ECHO; REJECT; BEGIN

20618 | The action ' | ' means that the action for the next rule is the action for this rule.  
 20619 Unlike the other three actions, ' | ' cannot be enclosed in braces or be semicolon-  
 20620 terminated; the application shall ensure that it is specified alone, with no other  
 20621 actions.

20622 **ECHO;** Write the contents of the string *yytext* on the output.

20623 **REJECT;** Usually only a single expression is matched by a given string in the input. **REJECT**  
 20624 means "continue to the next expression that matches the current input", and shall  
 20625 cause whatever rule was the second choice after the current rule to be executed for  
 20626 the same input. Thus, multiple rules can be matched and executed for one input  
 20627 string or overlapping input strings. For example, given the regular expressions  
 20628 "xyz" and "xy" and the input "xyz", usually only the regular expression "xyz"  
 20629 would match. The next attempted match would start after z. If the last action in the  
 20630 "xyz" rule is **REJECT**, both this rule and the "xy" rule would be executed. The  
 20631 **REJECT** action may be implemented in such a fashion that flow of control does not  
 20632 continue after it, as if it were equivalent to a **goto** to another part of *yylex*. The use  
 20633 of **REJECT** may result in somewhat larger and slower scanners.

20634 **BEGIN** The action:

20635 BEGIN *newstate*;

20636 switches the state (start condition) to *newstate*. If the string *newstate* has not been  
 20637 declared previously as a start condition in the *Definitions* section, the results are  
 20638 unspecified. The initial state is indicated by the digit '0' or the token **INITIAL**.

20639 The functions or macros described below are accessible to user code included in the *lex* input. It  
 20640 is unspecified whether they appear in the C code output of *lex*, or are accessible only through the  
 20641 **-ll** operand to *c99* or *cc* (the *lex* library).

20642 **int yylex(void)**

20643 Performs lexical analysis on the input; this is the primary function generated by the *lex*  
 20644 utility. The function shall return zero when the end of input is reached; otherwise, it shall  
 20645 return non-zero values (tokens) determined by the actions that are selected.

20646 **int yymore(void)**

20647 When called, indicates that when the next input string is recognized, it is to be appended to  
 20648 the current value of *yytext* rather than replacing it; the value in *yyleng* shall be adjusted  
 20649 accordingly.

20650 **int yless(int n)**

20651 Retains *n* initial characters in *yytext*, NUL-terminated, and treats the remaining characters  
 20652 as if they had not been read; the value in *yyleng* shall be adjusted accordingly.

20653 **int input(void)**

20654 Returns the next character from the input, or zero on end-of-file. It shall obtain input from  
 20655 the stream pointer *yyin*, although possibly via an intermediate buffer. Thus, once scanning  
 20656 has begun, the effect of altering the value of *yyin* is undefined. The character read is  
 20657 removed from the input stream of the scanner without any processing by the scanner.

20658 **int unput(int c)**  
 20659 Returns the character 'c' to the input; *yytext* and *yyleng* are undefined until the next  
 20660 expression is matched. The result of using *unput* for more characters than have been input is  
 20661 unspecified.

20662 The following functions appear only in the *lex* library accessible through the `-ll` operand; they  
 20663 can therefore be redefined by a portable application:

20664 **int yywrap(void)**  
 20665 Called by *yylex* at end-of-file; the default *yywrap* always shall return 1. If the application  
 20666 requires *yylex* to continue processing with another source of input, then the application can  
 20667 include a function *yywrap*, which associates another file with the external variable **FILE\****yyin*  
 20668 and shall return a value of zero.

20669 **int main(int argc, char \*argv[ ])**  
 20670 Calls *yylex* to perform lexical analysis, then exits. The user code can contain *main* to perform  
 20671 application-specific operations, calling *yylex* as applicable.

20672 Except for *input*, *unput*, and *main*, all external and static names generated by *lex* shall begin with  
 20673 the prefix **yy** or **YY**.

#### 20674 EXIT STATUS

20675 The following exit values shall be returned:

20676 0 Successful completion.

20677 >0 An error occurred.

#### 20678 CONSEQUENCES OF ERRORS

20679 Default.

#### 20680 APPLICATION USAGE

20681 Portable applications are warned that in the *Rules* section, an *ERE* without an action is not  
 20682 acceptable, but need not be detected as erroneous by *lex*. This may result in compilation or  
 20683 runtime errors.

20684 The purpose of *input* is to take characters off the input stream and discard them as far as the  
 20685 lexical analysis is concerned. A common use is to discard the body of a comment once the  
 20686 beginning of a comment is recognized.

20687 The *lex* utility is not fully internationalized in its treatment of regular expressions in the *lex*  
 20688 source code or generated lexical analyzer. It would seem desirable to have the lexical analyzer  
 20689 interpret the regular expressions given in the *lex* source according to the environment specified  
 20690 when the lexical analyzer is executed, but this is not possible with the current *lex* technology.  
 20691 Furthermore, the very nature of the lexical analyzers produced by *lex* must be closely tied to the  
 20692 lexical requirements of the input language being described, which is frequently locale-specific  
 20693 anyway. (For example, writing an analyzer that is used for French text is not automatically  
 20694 useful for processing other languages.)

#### 20695 EXAMPLES

20696 The following is an example of a *lex* program that implements a rudimentary scanner for a  
 20697 Pascal-like syntax:

```
20698 % {
20699 /* Need this for the call to atof() below. */
20700 #include <math.h>
20701 /* Need this for printf(), fopen(), and stdin below. */
20702 #include <stdio.h>
20703 % }
```

```

20704 DIGIT [0-9]
20705 ID [a-z][a-z0-9]*
20706 %%
20707 {DIGIT}+ {
20708 printf("An integer: %s (%d)\n", yytext,
20709 atoi(yytext));
20710 }
20711 {DIGIT}+"."{DIGIT}* {
20712 printf("A float: %s (%g)\n", yytext,
20713 atof(yytext));
20714 }
20715 if|then|begin|end|procedure|function {
20716 printf("A keyword: %s\n", yytext);
20717 }
20718 {ID} printf("An identifier: %s\n", yytext);
20719 "+"|"-"|"*"|"|" /" printf("An operator: %s\n", yytext);
20720 "{ "[^]\n]*" /* Eat up one-line comments. */
20721 [\t\n]+ /* Eat up white space. */
20722 . printf("Unrecognized character: %s\n", yytext);
20723 %%
20724 int main(int argc, char *argv[])
20725 {
20726 ++argv, --argc; /* Skip over program name. */
20727 if (argc > 0)
20728 yyin = fopen(argv[0], "r");
20729 else
20730 yyin = stdin;
20731 yylex();
20732 }

```

### 20733 RATIONALE

20734 Even though the `-c` option and references to the C language are retained in this description, *lex*  
 20735 may be generalized to other languages, as was done at one time for EFL, the Extended  
 20736 FORTRAN Language. Since the *lex* input specification is essentially language-independent,  
 20737 versions of this utility could be written to produce Ada, Modula-2, or Pascal code, and there are  
 20738 known historical implementations that do so.

20739 The current description of *lex* bypasses the issue of dealing with internationalized EREs in the *lex*  
 20740 source code or generated lexical analyzer. If it follows the model used by *awk* (the source code is  
 20741 assumed to be presented in the POSIX locale, but input and output are in the locale specified by  
 20742 the environment variables), then the tables in the lexical analyzer produced by *lex* would  
 20743 interpret EREs specified in the *lex* source in terms of the environment variables specified when  
 20744 *lex* was executed. The desired effect would be to have the lexical analyzer interpret the EREs  
 20745 given in the *lex* source according to the environment specified when the lexical analyzer is  
 20746 executed, but this is not possible with the current *lex* technology.

20747 The description of octal and hexadecimal-digit escape sequences agrees with the ISO C standard  
 20748 use of escape sequences. See the RATIONALE for *ed* (on page 2546) for a discussion of bytes

- 20749 larger than 9 bits being represented by octal values. Hexadecimal values can represent larger  
20750 bytes and multi-byte characters directly, using as many digits as required.
- 20751 There is no detailed output format specification. The observed behavior of *lex* under four  
20752 different historical implementations was that none of these implementations consistently  
20753 reported the line numbers for error and warning messages. Furthermore, there was a desire that  
20754 *lex* be allowed to output additional diagnostic messages. Leaving message formats unspecified  
20755 avoids these formatting questions and problems with internationalization.
- 20756 Although the `%x` specifier for *exclusive* start conditions is not historical practice, it is believed to  
20757 be a minor change to historical implementations and greatly enhances the usability of *lex*  
20758 programs since it permits an application to obtain the expected functionality with fewer  
20759 statements.
- 20760 The `%array` and `%pointer` declarations were added as a compromise between historical systems.  
20761 The System V-based *lex* copies the matched text to a *yytext* array. The *flex* program, supported in  
20762 BSD and GNU systems, uses a pointer. In the latter case, significant performance improvements  
20763 are available for some scanners. Most historical programs should require no change in porting  
20764 from one system to another because the string being referenced is null-terminated in both cases.  
20765 (The method used by *flex* in its case is to null-terminate the token in place by remembering the  
20766 character that used to come right after the token and replacing it before continuing on to the next  
20767 scan.) Multi-file programs with external references to *yytext* outside the scanner source file  
20768 should continue to operate on their historical systems, but would require one of the new  
20769 declarations to be considered strictly portable.
- 20770 The description of EREs avoids unnecessary duplication of ERE details because their meanings  
20771 within a *lex* ERE are the same as that for the ERE in this volume of IEEE Std. 1003.1-200x.
- 20772 The reason for the undefined condition associated with text beginning with a <blank> or within  
20773 "%{ " and "%}" delimiter lines appearing in the *Rules* section is historical practice. Both the BSD  
20774 and System V *lex* copy the indented (or enclosed) input in the *Rules* section (except at the  
20775 beginning) to unreachable areas of the *yylex* function (the code is written directly after a *break*  
20776 statement). In some cases, the System V *lex* generates an error message or a syntax error,  
20777 depending on the form of indented input.
- 20778 The intention in breaking the list of functions into those that may appear in *lex.yy.c* versus those  
20779 that only appear in *libl.a* is that only those functions in *libl.a* can be reliably redefined by a  
20780 portable application.
- 20781 The descriptions of standard output and standard error are somewhat complicated because  
20782 historical *lex* implementations chose to issue diagnostic messages to standard output (unless `-t`  
20783 was given). This standard allows this behavior, but leaves an opening for the more expected  
20784 behavior of using standard error for diagnostics. Also, the System V behavior of writing the  
20785 statistics when any table sizes are given is allowed, while BSD-derived systems can avoid it. The  
20786 programmer can always precisely obtain the desired results by using either the `-t` or `-n` options.
- 20787 The OPERANDS section does not mention the use of `-` as a synonym for standard input; not all  
20788 historical implementations support such usage for any of the *file* operands.
- 20789 A description of the *translation table* was deleted from early proposals because of its relatively  
20790 low usage in historical applications.
- 20791 The change to the definition of the *input* function that allows buffering of input presents the  
20792 opportunity for major performance gains in some applications.
- 20793 The following examples clarify the differences between *lex* regular expressions and regular  
20794 expressions appearing elsewhere in this volume of IEEE Std. 1003.1-200x. For regular  
20795 expressions of the form `"r/x"`, the string matching *r* is always returned; confusion may arise

20796 when the beginning of *x* matches the trailing portion of *r*. For example, given the regular  
20797 expression "a\*b/cc" and the input "aaabcc", *yytext* would contain the string "aaab" on this  
20798 match. But given the regular expression "x\*/xy" and the input "xxxxy", the token **xxx**, not **xx**,  
20799 is returned by some implementations because **xxx** matches "x\*".

20800 In the rule "ab\*/bc", the "b\*" at the end of *r* extends *r*'s match into the beginning of the  
20801 trailing context, so the result is unspecified. If this rule were "ab/bc", however, the rule  
20802 matches the text "ab" when it is followed by the text "bc". In this latter case, the matching of *r*  
20803 cannot extend into the beginning of *x*, so the result is specified.

20804 **FUTURE DIRECTIONS**

20805 None.

20806 **SEE ALSO**

20807 *c99*, *yacc*

20808 **CHANGE HISTORY**

20809 First released in Issue 2.

20810 **Issue 4**

20811 Aligned with the ISO/IEC 9945-2:1993 standard.

20812 **Issue 6**

20813 This utility is now marked as part of the C-Language Development Utilities option.

20814 The obsolescent `-c` option is withdrawn in this issue.

20815 The normative text is reworded to avoid use of the term "must" for application requirements.



20816 **NAME**20817 link — call *link()* function20818 **SYNOPSIS**20819 XSI link *file1 file2*

20820

20821 **DESCRIPTION**20822 The *link* utility shall perform the function call:20823 link(*file1*, *file2*);20824 A user may need appropriate privilege to invoke the *link* utility.20825 **OPTIONS**

20826 None.

20827 **OPERANDS**

20828 The following operands shall be supported:

20829 *file1* The path name of an existing file.20830 *file2* The path name of the new directory entry to be created.20831 **STDIN**

20832 Not used.

20833 **INPUT FILES**

20834 Not used.

20835 **ENVIRONMENT VARIABLES**20836 The following environment variables shall affect the execution of *link*:

20837 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 20838 If *LANG* is unset or null, the corresponding value from the implementation-  
 20839 defined default locale shall be used. If any of the internationalization variables  
 20840 contain an invalid setting, the utility behaves as if none of the variables had been  
 20841 set.

20842 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 20843 internationalization variables.

20844 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 20845 characters (for example, single-byte as opposed to multi-byte characters in  
 20846 arguments).

20847 *LC\_MESSAGES*

20848 Determine the locale that should be used to affect the format and contents of  
 20849 diagnostic messages written to standard error.

20850 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

20851 **ASYNCHRONOUS EVENTS**

20852 Default.

20853 **STDOUT**

20854 None.

20855 **STDERR**

20856           Used only for diagnostic messages.

20857 **OUTPUT FILES**

20858           None.

20859 **EXTENDED DESCRIPTION**

20860           None.

20861 **EXIT STATUS**

20862           The following exit values shall be returned:

20863           0   Successful completion.

20864           &gt;0  An error occurred.

20865 **CONSEQUENCES OF ERRORS**

20866           Default.

20867 **APPLICATION USAGE**

20868           None.

20869 **EXAMPLES**

20870           None.

20871 **RATIONALE**

20872           None.

20873 **FUTURE DIRECTIONS**

20874           None.

20875 **SEE ALSO**20876           *In, unlink*, the System Interfaces volume of IEEE Std. 1003.1-200x, *link()*20877 **CHANGE HISTORY**

20878           First released in Issue 5.

20879 **NAME**20880 **ln** — link files20881 **SYNOPSIS**20882 **ln** [-fs] *source\_file target\_file*20883 **ln** [-fs] *source\_file ... target\_dir*20884 **DESCRIPTION**

20885 In the first synopsis form, the *ln* utility shall create a new directory entry (link), or if the **-s**  
 20886 option is specified a symbolic link, for the file specified by the *source\_file* operand, at the  
 20887 *destination* path specified by the *target\_file* operand. This first synopsis form shall be assumed  
 20888 when the final operand does not name an existing directory; if more than two operands are  
 20889 specified and the final is not an existing directory, an error shall result.

20890 In the second synopsis form, the *ln* utility shall create a new directory entry (link), or if the **-s**  
 20891 option is specified a symbolic link, for each file specified by a *source\_file* operand, at a *destination*  
 20892 path in the existing directory named by *target\_dir*.

20893 If the last operand specifies an existing file of a type not specified by the System Interfaces  
 20894 volume of IEEE Std. 1003.1-200x, the behavior is implementation-defined.

20895 The corresponding *destination* path for each *source\_file* shall be the concatenation of the target  
 20896 directory path name, a slash character, and the last path name component of the *source\_file*. The  
 20897 second synopsis form shall be assumed when the final operand names an existing directory.

20898 For each *source\_file*:

- 20899 1. If the *destination* path exists:
- 20900 a. If the **-f** option is not specified, *ln* shall write a diagnostic message to standard error,  
 20901 do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
  - 20902 b. Actions shall be performed equivalent to the *unlink()* function defined in the System  
 20903 Interfaces volume of IEEE Std. 1003.1-200x, called using *destination* as the *path*  
 20904 argument. If this fails for any reason, *ln* shall write a diagnostic message to standard  
 20905 error, do nothing more with the current *source\_file*, and go on to any remaining  
 20906 *source\_files*.
- 20907 2. If the **-s** option is specified, *ln* shall create a symbolic link named by the *destination* path  
 20908 and containing as its path name *source\_file*. The *ln* utility shall do nothing more with  
 20909 *source\_file* and shall go on to any remaining files.
- 20910 3. If *source\_file* is a symbolic link, actions shall be performed equivalent to the *link()* function  
 20911 using the object that *source\_file* references as the *path1* argument and the destination path  
 20912 as the *path2* argument. The *ln* utility shall do nothing more with *source\_file* and shall go on  
 20913 to any remaining files.
- 20914 4. Actions shall be performed equivalent to the *link()* function defined in the System  
 20915 Interfaces volume of IEEE Std. 1003.1-200x using *source\_file* as the *path1* argument, and the  
 20916 *destination* path as the *path2* argument.

20917 **OPTIONS**

20918 The *ln* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 20919 Utility Syntax Guidelines.

20920 The following option shall be supported:

20921 **-f** Force existing *destination* path names to be removed to allow the link.

- 20922            **-s**            Create symbolic links instead of hard links.
- 20923 **OPERANDS**
- 20924            The following operands shall be supported:
- 20925            *source\_file*    A path name of a file to be linked. If the **-s** option is specified, no restrictions on the type of file or on its existence shall be made. If the **-s** option is not specified, whether a directory can be linked is implementation-defined.
- 20926
- 20927
- 20928            *target\_file*    The path name of the new directory entry to be created.
- 20929            *target\_dir*    A path name of an existing directory in which the new directory entries are created.
- 20930
- 20931 **STDIN**
- 20932            Not used.
- 20933 **INPUT FILES**
- 20934            None.
- 20935 **ENVIRONMENT VARIABLES**
- 20936            The following environment variables shall affect the execution of *ln*:
- 20937            *LANG*            Provide a default value for the internationalization variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined.
- 20938
- 20939
- 20940
- 20941
- 20942            *LC\_ALL*            If set to a non-empty string value, override the values of all the other internationalization variables.
- 20943
- 20944            *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).
- 20945
- 20946
- 20947            *LC\_MESSAGES*
- 20948                            Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
- 20949
- 20950 XSI        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 20951 **ASYNCHRONOUS EVENTS**
- 20952            Default.
- 20953 **STDOUT**
- 20954            Not used.
- 20955 **STDERR**
- 20956            Used only for diagnostic messages.
- 20957 **OUTPUT FILES**
- 20958            None.
- 20959 **EXTENDED DESCRIPTION**
- 20960            None.
- 20961 **EXIT STATUS**
- 20962            The following exit values shall be returned:
- 20963            0    All the specified files were linked successfully.

20964 >0 An error occurred.

20965 **CONSEQUENCES OF ERRORS**

20966 Default.

20967 **APPLICATION USAGE**

20968 None.

20969 **EXAMPLES**

20970 None.

20971 **RATIONALE**

20972 Some historic versions of *ln* (including the one specified by the SVID, unlink the destination file,  
20973 if it exists, by default. If the mode does not permit writing, these versions prompt for  
20974 confirmation before attempting the unlink. In these versions the `-f` option causes *ln* not to  
20975 attempt to prompt for confirmation.

20976 This allows *ln* to succeed in creating links when the target file already exists, even if the file itself  
20977 is not writable (although the directory must be). Early proposals specified this functionality.

20978 This volume of IEEE Std. 1003.1-200x does not allow the *ln* utility to unlink existing destination  
20979 paths by default for the following reasons:

- 20980 • The *ln* utility has historically been used to provide locking for shell applications, a usage that  
20981 is incompatible with *ln* unlinking the destination path by default. There was no  
20982 corresponding technical advantage to adding this functionality.
- 20983 • This functionality gave *ln* the ability to destroy the link structure of files, which changes the  
20984 historical behavior of *ln*.
- 20985 • This functionality is easily replicated with a combination of *rm* and *ln*.
- 20986 • It is not historical practice in many systems; BSD and BSD-derived systems do not support  
20987 this behavior. Unfortunately, whichever behavior is selected can cause scripts written  
20988 expecting the other behavior to fail.
- 20989 • It is preferable that *ln* perform in the same manner as the *link()* function, which does not  
20990 permit the target to exist already.

20991 This volume of IEEE Std. 1003.1-200x retains the `-f` option to provide support for shell scripts  
20992 depending on the SVID semantics. It seems likely that shell scripts would not be written to  
20993 handle prompting by *ln* and would therefore have specified the `-f` option.

20994 The `-f` option is an undocumented feature of many historical versions of the *ln* utility, allowing  
20995 linking to directories. These versions require modification.

20996 Early proposals of this volume of IEEE Std. 1003.1-200x also required an `-i` option, which  
20997 behaved like the `-i` options in *cp* and *mv*, prompting for confirmation before unlinking existing  
20998 files. This was not historical practice for the *ln* utility and has been omitted.

20999 **FUTURE DIRECTIONS**

21000 None.

21001 **SEE ALSO**

21002 *chmod*, *find*, *pax*, *rm*, the System Interfaces volume of IEEE Std. 1003.1-200x, *link()*

21003 **CHANGE HISTORY**

21004 First released in Issue 2.

21005 **Issue 4**

21006 Aligned with the ISO/IEC 9945-2:1993 standard.

21007 **Issue 6**21008 The *ln* utility is updated to include symbolic link processing as defined in the IEEE P1003.2b  
21009 draft standard.

21010 **NAME**

21011 locale — get locale-specific information

21012 **SYNOPSIS**

21013 locale [-a | -m]

21014 locale [-ck] *name*...21015 **DESCRIPTION**

21016 The *locale* utility shall write information about the current locale environment, or all public  
 21017 locales, to the standard output. For the purposes of this section, a *public locale* is one provided by  
 21018 the implementation that is accessible to the application.

21019 When *locale* is invoked without any arguments, it shall summarize the current locale  
 21020 environment for each locale category as determined by the settings of the environment variables  
 21021 defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 7, Locale.

21022 When invoked with operands, it shall write values that have been assigned to the keywords in  
 21023 the locale categories, as follows:

- 21024 • Specifying a keyword name shall select the named keyword and the category containing that  
 21025 keyword.
- 21026 • Specifying a category name shall select the named category and all keywords in that  
 21027 category.

21028 **OPTIONS**

21029 The *locale* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 21030 12.2, Utility Syntax Guidelines.

21031 The following options shall be supported:

- 21032 **-a** Write information about all available public locales. The available locales shall  
 21033 include **POSIX**, representing the POSIX locale. The manner in which the  
 21034 implementation determines what other locales are available is implementation-  
 21035 defined.
- 21036 **-c** Write the names of selected locale categories; see the **STDOUT** section. The **-c**  
 21037 option increases readability when more than one category is selected (for example,  
 21038 via more than one keyword name or via a category name). It is valid both with  
 21039 and without the **-k** option.
- 21040 **-k** Write the names and values of selected keywords. The implementation may omit  
 21041 values for some keywords; see the **OPERANDS** section.
- 21042 **-m** Write names of available charmaps; see the Base Definitions volume of  
 21043 IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set.

21044 **OPERANDS**

21045 The following operand shall be supported:

- 21046 *name* The name of a locale category as defined in the Base Definitions volume of  
 21047 IEEE Std. 1003.1-200x, Chapter 7, Locale, the name of a keyword in a locale  
 21048 category, or the reserved name **charmap**. The named category or keyword shall be  
 21049 selected for output. If a single *name* represents both a locale category name and a  
 21050 keyword name in the current locale, the results are unspecified. Otherwise, both  
 21051 category and keyword names can be specified as *name* operands, in any sequence.  
 21052 It is implementation-defined whether any keyword values are written for the  
 21053 categories *LC\_CTYPE* and *LC\_COLLATE*.

21054 **STDIN**

21055 Not used.

21056 **INPUT FILES**

21057 None.

21058 **ENVIRONMENT VARIABLES**21059 The following environment variables shall affect the execution of *locale*:

21060 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 21061 If *LANG* is unset or null, the corresponding value from the implementation-  
 21062 defined default locale shall be used. If any of the internationalization variables  
 21063 contains an invalid setting, the utility shall behave as if none of the variables had  
 21064 been defined.

21065 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 21066 internationalization variables.

21067 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 21068 characters (for example, single-byte as opposed to multi-byte characters in  
 21069 arguments and input files).

21070 *LC\_MESSAGES*

21071 Determine the locale that should be used to affect the format and contents of  
 21072 diagnostic messages written to standard error.

21073 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

21074 XSI The application shall ensure that the *LANG*, *LC\_\**, and *NLSPATH* environment variables specify  
 21075 the current locale environment to be written out; they shall be used if the *-a* option is not  
 21076 specified.

21077 **ASYNCHRONOUS EVENTS**

21078 Default.

21079 **STDOUT**

21080 If *locale* is invoked without any options or operands, the names and values of the *LANG* and  
 21081 *LC\_\** environment variables described in this volume of IEEE Std. 1003.1-200x shall be written to  
 21082 the standard output, one variable per line, with *LANG* first, and each line using the following  
 21083 format. Only those variables set in the environment and not overridden by *LC\_ALL* shall be  
 21084 written using this format:

21085 "%s=%s\n", &lt;variable\_name&gt;, &lt;value&gt;

21086 The names of those *LC\_\** variables associated with locale categories defined in this volume of  
 21087 IEEE Std. 1003.1-200x that are not set in the environment or are overridden by *LC\_ALL* shall be  
 21088 written in the following format:

21089 "%s=\"\"%s\"\"\n", &lt;variable\_name&gt;, &lt;implied value&gt;

21090 The <implied value> shall be the name of the locale that has been selected for that category by the  
 21091 implementation, based on the values in *LANG* and *LC\_ALL*, as described in the Base Definitions  
 21092 volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.

21093 The <value> and <implied value> shown above shall be properly quoted for possible later reentry  
 21094 to the shell. The <value> shall not be quoted using double-quotes (so that it can be distinguished  
 21095 by the user from the <implied value> case, which always requires double-quotes).

21096 The *LC\_ALL* variable shall be written last, using the first format shown above. If it is not set, it  
 21097 shall be written as:



21098 "LC\_ALL=\n"

21099 If any arguments are specified:

21100 1. If the **-a** option is specified, the names of all the public locales shall be written, each in the  
21101 following format:

21102 "%s\n", <locale name>

21103 2. If the **-c** option is specified, the names of all selected categories shall be written, each in the  
21104 following format:

21105 "%s\n", <category name>

21106 If keywords are also selected for writing (see following items), the category name output  
21107 shall precede the keyword output for that category.

21108 If the **-c** option is not specified, the names of the categories shall not be written; only the  
21109 keywords, as selected by the <name> operand, shall be written.

21110 3. If the **-k** option is specified, the names and values of selected keywords shall be written. If  
21111 a value is non-numeric, it shall be written in the following format:

21112 "%s=\"%s\"\\n", <keyword name>, <keyword value>

21113 If the keyword was **charmap**, the name of the charmap (if any) that was specified via the  
21114 *localedef* **-f** option when the locale was created shall be written, with the word **charmap** as  
21115 <keyword name>.

21116 If a value is numeric, it shall be written in one of the following formats:

21117 "%s=%d\n", <keyword name>, <keyword value>

21118 "%s=%c%o\n", <keyword name>, <escape character>, <keyword value>

21119 "%s=%cx%x\n", <keyword name>, <escape character>, <keyword value>

21120 where the <escape character> is that identified by the **escape\_char** keyword in the current  
21121 locale; see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3, Locale  
21122 Definition.

21123 Compound keyword values (list entries) shall be separated in the output by semicolons.  
21124 When included in keyword values, the semicolon, the double-quote, the backslash, and  
21125 any control character shall be preceded (escaped) with the escape character.

21126 4. If the **-k** option is not specified, selected keyword values shall be written, each in the  
21127 following format:

21128 "%s\n", <keyword value>

21129 If the keyword was **charmap**, the name of the charmap (if any) that was specified via the  
21130 *localedef* **-f** option when the locale was created shall be written.

21131 5. If the **-m** option is specified, then a list of all available charmaps shall be written, each in  
21132 the format:

21133 "%s\n", <charmap>

21134 where <charmap> is in a format suitable for use as the option-argument to the *localedef* **-f**  
21135 option.

21136 **STDERR**

21137           Used only for diagnostic messages.

21138 **OUTPUT FILES**

21139           None.

21140 **EXTENDED DESCRIPTION**

21141           None.

21142 **EXIT STATUS**

21143           The following exit values shall be returned:

21144           0   All the requested information was found and output successfully.

21145           >0  An error occurred.

21146 **CONSEQUENCES OF ERRORS**

21147           Default.

21148 **APPLICATION USAGE**

21149           If the *LANG* environment variable is not set or set to an empty value, or one of the *LC\_\**  
21150           environment variables is set to an unrecognized value, the actual locales assumed (if any) are  
21151           implementation-defined as described in the Base Definitions volume of IEEE Std. 1003.1-200x,  
21152           Chapter 8, Environment Variables.

21153           Implementations are not required to write out the actual values for keywords in the categories  
21154           *LC\_CTYPE* and *LC\_COLLATE*; however, they must write out the categories (allowing an  
21155           application to determine, for example, which character classes are available).

21156 **EXAMPLES**

21157           In the following examples, the assumption is that locale environment variables are set as  
21158           follows:

21159           LANG=locale\_x

21160           LC\_COLLATE=locale\_y

21161           The command *locale* would result in the following output:

21162           LANG=locale\_x

21163           LC\_CTYPE="locale\_x"

21164           LC\_COLLATE=locale\_y

21165           LC\_TIME="locale\_x"

21166           LC\_NUMERIC="locale\_x"

21167           LC\_MONETARY="locale\_x"

21168           LC\_MESSAGES="locale\_x"

21169           LC\_ALL=

21170           The order of presentation of the categories is not specified by this volume of  
21171           IEEE Std. 1003.1-200x.

21172           The command:

21173           LC\_ALL=POSIX locale -ck decimal\_point

21174           would produce:

21175           LC\_NUMERIC

21176           decimal\_point="."

21177           The following command shows an application of *locale* to determine whether a user-supplied  
21178           response is affirmative:

```
21179 if printf "%s\n" "$response" | grep -Eq "$(locale yesexpr)"
21180 then
21181 affirmative processing goes here
21182 else
21183 non-affirmative processing goes here
21184 fi
```

**21185 RATIONALE**

21186 The output for categories *LC\_CTYPE* and *LC\_COLLATE* has been made implementation-defined |  
21187 because there is a questionable value in having a shell script receive an entire array of characters. |  
21188 It is also difficult to return a logical collation description, short of returning a complete *localedef*  
21189 source.

21190 The **-m** option was included to allow applications to query for the existence of charmaps. The  
21191 output is a list of the charmaps (implementation-supplied and user-supplied, if any) on the  
21192 system.

21193 The **-c** option was included for readability when more than one category is selected (for  
21194 example, via more than one keyword name or via a category name). It is valid both with and  
21195 without the **-k** option.

21196 The **charmap** keyword, which returns the name of the charmap (if any) that was used when the  
21197 current locale was created, was included to allow applications needing the information to  
21198 retrieve it.

**21199 FUTURE DIRECTIONS**

21200 None.

**21201 SEE ALSO**

21202 *localedef*, the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3, Locale Definition |

**21203 CHANGE HISTORY**

21204 First released in Issue 4.

**21205 Issue 5**

21206 FUTURE DIRECTIONS section added.

**21207 Issue 6**

21208 The normative text is reworded to avoid use of the term “must” for application requirements.

## 21209 NAME

21210 localedef — define locale environment

## 21211 SYNOPSIS

21212 localedef [-c][-f *charmap*][-i *sourcefile*][-u *code\_set\_name*] *name*

## 21213 DESCRIPTION

21214 The *localedef* utility shall convert source definitions for locale categories into a format usable by  
 21215 the functions and utilities whose operational behavior is determined by the setting of the locale  
 21216 environment variables defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter  
 21217 7, Locale. It is implementation-defined whether users have the capability to create new locales,  
 21218 in addition to those supplied by the implementation. If the symbolic constant  
 21219 `POSIX2_LOCALEDEF` is defined, the system supports the creation of new locales. On XSI-  
 21220 conformant systems, the symbolic constant `POSIX2_LOCALEDEF` shall be defined.

21221 The utility shall read source definitions for one or more locale categories belonging to the same  
 21222 locale from the file named in the `-i` option (if specified) or from standard input.

21223 The *name* operand identifies the target locale. The utility shall support the creation of *public*, or  
 21224 generally accessible locales, as well as *private*, or restricted-access locales. Implementations may  
 21225 restrict the capability to create or modify public locales to users with the appropriate privileges.

21226 Each category source definition shall be identified by the corresponding environment variable  
 21227 name and terminated by an `END category-name` statement. The following categories shall be  
 21228 supported. In addition, the input may contain source for implementation-defined categories.

21229 `LC_CTYPE` Defines character classification and case conversion.

21230 `LC_COLLATE`

21231 Defines collation rules.

21232 `LC_MONETARY`

21233 Defines the format and symbols used in formatting of monetary information.

21234 `LC_NUMERIC`

21235 Defines the decimal delimiter, grouping, and grouping symbol for non-monetary  
 21236 numeric editing.

21237 `LC_TIME` Defines the format and content of date and time information.

21238 `LC_MESSAGES`

21239 Defines the format and values of affirmative and negative responses.

## 21240 OPTIONS

21241 The *localedef* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,  
 21242 Section 12.2, Utility Syntax Guidelines.

21243 The following options shall be supported:

21244 `-c` Create permanent output even if warning messages have been issued.

21245 `-f charmap` Specify the path name of a file containing a mapping of character symbols and  
 21246 collating element symbols to actual character encodings. The format of the  
 21247 *charmap* is described under the Base Definitions volume of IEEE Std. 1003.1-200x,  
 21248 Section 6.4, Character Set Description File. The application shall ensure that this  
 21249 option is specified if symbolic names (other than collating symbols defined in a  
 21250 **collating-symbol** keyword) are used. If the `-f` option is not present, an  
 21251 implementation-defined character mapping shall be used.

21252        **-i *inputfile***   The path name of a file containing the source definitions. If this option is not  
 21253                            present, source definitions shall be read from standard input. The format of the  
 21254                            *inputfile* is described in the Base Definitions volume of IEEE Std. 1003.1-200x,  
 21255                            Section 7.3, Locale Definition.

21256        **-u *code\_set\_name*** Specify the name of a codeset used as the target mapping of character symbols  
 21257                            and collating element symbols whose encoding values are defined in terms of the  
 21258                            ISO/IEC 10646-1: 1993 standard position constant values.

#### 21259 OPERANDS

21260        The following operand shall be supported:

21261        ***name***           Identifies the locale; see the Base Definitions volume of IEEE Std. 1003.1-200x,  
 21262                            Chapter 7, Locale for a description of the use of this name. If the name contains one  
 21263                            or more slash characters, *name* shall be interpreted as a path name where the  
 21264                            created locale definitions shall be stored. If *name* does not contain any slash  
 21265                            characters, the interpretation of the name is implementation-defined and the locale  
 21266                            shall be public. This capability may be restricted to users with appropriate  
 21267                            privileges. (As a consequence of specifying one *name*, although several categories  
 21268                            can be processed in one execution, only categories belonging to the same locale can  
 21269                            be processed.)

#### 21270 STDIN

21271        Unless the **-i** option is specified, the standard input shall be a text file containing one or more  
 21272        locale category source definitions, as described in the Base Definitions volume of  
 21273        IEEE Std. 1003.1-200x, Section 7.3, Locale Definition. When lines are continued using the escape  
 21274        character mechanism, there is no limit to the length of the accumulated continued line.

#### 21275 INPUT FILES

21276        The character set mapping file specified as the *charmap* option-argument is described under the  
 21277        Base Definitions volume of IEEE Std. 1003.1-200x, Section 6.4, Character Set Description File. If a  
 21278        locale category source definition contains a **copy** statement, as defined in the Base Definitions  
 21279        volume of IEEE Std. 1003.1-200x, Chapter 7, Locale, and the **copy** statement names a valid,  
 21280        existing locale, then *localedef* shall behave as if the source definition had contained a valid  
 21281        category source definition for the named locale.

#### 21282 ENVIRONMENT VARIABLES

21283        The following environment variables shall affect the execution of *localedef*:

21284        ***LANG***           Provide a default value for the internationalization variables that are unset or null.  
 21285                            If *LANG* is unset or null, the corresponding value from the implementation-  
 21286                            defined default locale shall be used. If any of the internationalization variables  
 21287                            contains an invalid setting, the utility shall behave as if none of the variables had  
 21288                            been defined.

21289        ***LC\_ALL***        If set to a non-empty string value, override the values of all the other  
 21290                            internationalization variables.

21291        ***LC\_COLLATE***

21292                            (This variable has no affect on *localedef*; the POSIX locale is used for this category.)

21293        ***LC\_CTYPE***   Determine the locale for the interpretation of sequences of bytes of text data as  
 21294                            characters (for example, single-byte as opposed to multi-byte characters in  
 21295                            arguments and input files). This variable has no affect on the processing of *localedef*  
 21296                            input data; the POSIX locale is used for this purpose, regardless of the value of this  
 21297                            variable.

- 21298 *LC\_MESSAGES*  
 21299 Determine the locale that should be used to affect the format and contents of  
 21300 diagnostic messages written to standard error.
- 21301 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 21302 **ASYNCHRONOUS EVENTS**  
 21303 Default.
- 21304 **STDOUT**  
 21305 The utility shall report all categories successfully processed, in an unspecified format.
- 21306 **STDERR**  
 21307 Used only for diagnostic messages.
- 21308 **OUTPUT FILES**  
 21309 The format of the created output is unspecified. If the *name* operand does not contain a slash, the  
 21310 existence of an output file for the locale is unspecified.
- 21311 **EXTENDED DESCRIPTION**  
 21312 When the *-u* option is used, the *code\_set\_name* option-argument shall be interpreted as an  
 21313 implementation-defined name of a codeset to which the ISO/IEC 10646-1:1993 standard  
 21314 position constant values shall be converted via an implementation-defined method. Both the  
 21315 ISO/IEC 10646-1:1993 standard position constant values and other formats (decimal,  
 21316 hexadecimal, or octal) shall be valid as encoding values within the *charmap* file. The codeset  
 21317 represented by the implementation-defined name can be any codeset that is supported by the  
 21318 implementation.
- 21319 When conflicts occur between the *charmap* specification of *<code\_set\_name>*, *<mb\_cur\_max>*, or  
 21320 *<mb\_cur\_min>* and the implementation-defined interpretation of these respective items for the  
 21321 codeset represented by the *-u* option-argument *code\_set\_name*, the result is unspecified.
- 21322 When conflicts occur between the *charmap* encoding values specified for symbolic names of  
 21323 characters of the portable character set and the implementation-defined assignment of character  
 21324 encoding values, the result is unspecified.
- 21325 If a non-printable character in the *charmap* has a width specified that is not *-1*, *localedef* shall  
 21326 generate a warning.
- 21327 **EXIT STATUS**  
 21328 The following exit values shall be returned:
- 21329 0 No errors occurred and the locales were successfully created.  
 21330 1 Warnings occurred and the locales were successfully created.  
 21331 2 The locale specification exceeded implementation limits or the coded character set or sets  
 21332 used were not supported by the implementation, and no locale was created.  
 21333 3 The capability to create new locales is not supported by the implementation.  
 21334 >3 Warnings or errors occurred and no output was created.
- 21335 **CONSEQUENCES OF ERRORS**  
 21336 If an error is detected, no permanent output shall be created.
- 21337 If warnings occur, permanent output shall be created if the *-c* option was specified. The  
 21338 following conditions shall cause warning messages to be issued:
- 21339 • If a symbolic name not found in the *charmap* file is used for the descriptions of the *LC\_CTYPE*  
 21340 or *LC\_COLLATE* categories (for other categories, this shall be an error condition).

- 21341 • If the number of operands to the **order** keyword exceeds the {COLL\_WEIGHTS\_MAX} limit.
- 21342 • If optional keywords not supported by the implementation are present in the source.
- 21343 • If a non-printable character has a width specified other than -1.
- 21344 Other implementation-defined conditions may also cause warnings.

**21345 APPLICATION USAGE**

21346 The *charmap* definition is optional, and is contained outside the locale definition. This allows  
21347 both completely self-defined source files, and generic sources (applicable to more than one  
21348 codeset). To aid portability, all *charmap* definitions must use the same symbolic names for the  
21349 portable character set. As explained in the Base Definitions volume of IEEE Std. 1003.1-200x,  
21350 Section 6.4, Character Set Description File, it is implementation-defined whether or not users or  
21351 applications can provide additional character set description files. Therefore, the **-f** option might  
21352 be operable only when an implementation-defined *charmap* is named.

**21353 EXAMPLES**

21354 None.

**21355 RATIONALE**

21356 The output produced by the *localedef* utility is implementation-defined. The *name* operand is  
21357 used to identify the specific locale. (As a consequence, although several categories can be  
21358 processed in one execution, only categories belonging to the same locale can be processed.)

**21359 FUTURE DIRECTIONS**

21360 None.

**21361 SEE ALSO**

21362 *locale*, the Base Definitions volume of IEEE Std. 1003.1-200x, Section 7.3, Locale Definition

**21363 CHANGE HISTORY**

21364 First released in Issue 4.

**21365 Issue 6**

21366 The **-u** option is added, as specified in the IEEE P1003.2b draft standard.

21367 The normative text is reworded to avoid use of the term “must” for application requirements.

21368 **NAME**

21369           logger — log messages

21370 **SYNOPSIS**21371           logger *string* ...21372 **DESCRIPTION**

21373           The *logger* utility saves a message, in an unspecified manner and format, containing the *string*  
 21374           operands provided by the user. The messages are expected to be evaluated later by personnel  
 21375           performing system administration tasks.

21376           It is implementation-defined whether messages written in locales other than the POSIX locale  
 21377           are effective.

21378 **OPTIONS**

21379           None.

21380 **OPERANDS**

21381           The following operand shall be supported:

21382           *string*       One of the string arguments whose contents are concatenated together, in the  
 21383           order specified, separated by single <space> characters.

21384 **STDIN**

21385           Not used.

21386 **INPUT FILES**

21387           None.

21388 **ENVIRONMENT VARIABLES**21389           The following environment variables shall affect the execution of *logger*:

21390           *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 21391           If *LANG* is unset or null, the corresponding value from the implementation-  
 21392           defined default locale shall be used. If any of the internationalization variables  
 21393           contains an invalid setting, the utility shall behave as if none of the variables had  
 21394           been defined.

21395           *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 21396           internationalization variables.

21397           *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 21398           characters (for example, single-byte as opposed to multi-byte characters in  
 21399           arguments).

21400           *LC\_MESSAGES*

21401           Determine the locale that should be used to affect the format and contents of  
 21402           diagnostic messages written to standard error. (This means diagnostics from *logger*  
 21403           to the user or application, not diagnostic messages that the user is sending to the  
 21404           system administrator.)

21405 XSI           *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

21406 **ASYNCHRONOUS EVENTS**

21407           Default.

21408 **STDOUT**

21409           Not used.



21410 **STDERR**

21411           Used only for diagnostic messages.

21412 **OUTPUT FILES**

21413           Unspecified.

21414 **EXTENDED DESCRIPTION**

21415           None.

21416 **EXIT STATUS**

21417           The following exit values shall be returned:

21418           0   Successful completion.

21419           >0  An error occurred.

21420 **CONSEQUENCES OF ERRORS**

21421           Default.

21422 **APPLICATION USAGE**

21423           This utility allows logging of information for later use by a system administrator or programmer  
21424           in determining why non-interactive utilities have failed. The locations of the saved messages,  
21425           their format, and retention period are all unspecified. There is no method for a portable  
21426           application to read messages, once written.

21427 **EXAMPLES**

21428           A batch application, running non-interactively, tries to read a configuration file and fails; it may  
21429           attempt to notify the system administrator with:

```
21430 logger myname: unable to read file foo. [timestamp]
```

21431 **RATIONALE**

21432           The standard developers believed strongly that some method of alerting administrators to errors  
21433           was necessary. The obvious example is a batch utility, running non-interactively, that is unable  
21434           to read its configuration files or that is unable to create or write its results file. However, the  
21435           standard developers did not wish to define the format or delivery mechanisms as they have  
21436           historically been (and will probably continue to be) very system-specific, as well as involving  
21437           functionality clearly outside of the scope of this volume of IEEE Std. 1003.1-200x.

21438           The text with *LC\_MESSAGES* about diagnostic messages means diagnostics from *logger* to the  
21439           user or application, not diagnostic messages that the user is sending to the system administrator.

21440           Multiple *string* arguments are allowed, similar to *echo*, for ease-of-use.

21441           Like the utilities *mailx* and *lp*, *logger* is admittedly difficult to test. This was not deemed sufficient  
21442           justification to exclude these utilities from this volume of IEEE Std. 1003.1-200x. It is also  
21443           arguable that they are, in fact, testable, but that the tests themselves are not portable.

21444 **FUTURE DIRECTIONS**

21445           None.

21446 **SEE ALSO**

21447           *mailx*, *write*

21448 **CHANGE HISTORY**

21449           First released in Issue 4.

21450 **NAME**

21451 logname — return the user's login name

21452 **SYNOPSIS**

21453 logname

21454 **DESCRIPTION**

21455 The *logname* utility shall write the user's login name to standard output. The login name shall be  
21456 the string that would be returned by the *getlogin()* function defined in the System Interfaces  
21457 volume of IEEE Std. 1003.1-200x. Under the conditions where the *getlogin()* function would fail,  
21458 the *logname* utility shall write a diagnostic message to standard error and exit with a non-zero  
21459 exit status.

21460 **OPTIONS**

21461 None.

21462 **OPERANDS**

21463 None.

21464 **STDIN**

21465 Not used.

21466 **INPUT FILES**

21467 None.

21468 **ENVIRONMENT VARIABLES**

21469 The following environment variables shall affect the execution of *logname*:

21470 *LANG* Provide a default value for the internationalization variables that are unset or null.  
21471 If *LANG* is unset or null, the corresponding value from the implementation-  
21472 defined default locale shall be used. If any of the internationalization variables  
21473 contains an invalid setting, the utility shall behave as if none of the variables had  
21474 been defined.

21475 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
21476 internationalization variables.

21477 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
21478 characters (for example, single-byte as opposed to multi-byte characters in  
21479 arguments).

21480 *LC\_MESSAGES*

21481 Determine the locale that should be used to affect the format and contents of  
21482 diagnostic messages written to standard error.

21483 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

21484 **ASYNCHRONOUS EVENTS**

21485 Default.

21486 **STDOUT**

21487 The *logname* utility output shall be a single line consisting of the user's login name:

21488 "%s\n", <login name>

21489 **STDERR**

21490 Used only for diagnostic messages.

21491 **OUTPUT FILES**

21492           None.

21493 **EXTENDED DESCRIPTION**

21494           None.

21495 **EXIT STATUS**

21496           The following exit values shall be returned:

21497           0   Successful completion.

21498           &gt;0  An error occurred.

21499 **CONSEQUENCES OF ERRORS**

21500           Default.

21501 **APPLICATION USAGE**21502           The *logname* utility explicitly ignores the *LOGNAME* environment variable because environment changes could produce erroneous results.21504 **EXAMPLES**

21505           None.

21506 **RATIONALE**21507           The **passwd** file is not listed as required because the implementation may have other means of mapping login names.21509 **FUTURE DIRECTIONS**

21510           None.

21511 **SEE ALSO**21512           *id, who*21513 **CHANGE HISTORY**

21514           First released in Issue 2.

21515 **Issue 4**

21516           Aligned with the ISO/IEC 9945-2:1993 standard.

## 21517 NAME

21518 lp — send files to a printer

## 21519 SYNOPSIS

21520 lp [-c][-d *dest*][-n *copies*][-msw][-o *option*]... [-t *title*][*file*...]

## 21521 DESCRIPTION

21522 The *lp* utility shall copy the input files to an output destination in an unspecified manner. The  
 21523 default output destination should be to a hardcopy device, such as a printer or microfilm  
 21524 recorder, that produces non-volatile, human-readable documents. If such a device is not  
 21525 available to the application, or if the system provides no such device, the *lp* utility shall exit with  
 21526 a non-zero exit status.

21527 The actual writing to the output device may occur some time after the *lp* utility successfully  
 21528 exits. During the portion of the writing that corresponds to each input file, the implementation  
 21529 shall guarantee exclusive access to the device.

21530 The *lp* utility shall associate a unique *request ID* with each request.

21531 Normally, a banner page is produced to separate and identify each print job. This page may be  
 21532 suppressed by implementation-defined conditions, such as an operator command or one of the  
 21533 **-o** *option* values.

## 21534 OPTIONS

21535 The *lp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 21536 Utility Syntax Guidelines.

21537 The following options shall be supported:

21538 **-c** Exit only after further access to any of the input files is no longer required. The  
 21539 application can then safely delete or modify the files without affecting the output  
 21540 operation. Normally, files are not copied, but are linked whenever possible. If the  
 21541 **-c** option is not given, then the user should be careful not to remove any of the  
 21542 files before the request has been printed in its entirety. It should also be noted that  
 21543 in the absence of the **-c** option, any changes made to the named files after the  
 21544 request is made but before it is printed are reflected in the printed output. On some  
 21545 systems, **-c** may be on by default.

21546 **-d** *dest* Specify a string that names the destination (*dest*). If *dest* is a printer, the request  
 21547 shall be printed only on that specific printer. If *dest* is a class of printers, the request  
 21548 shall be printed on the first available printer that is a member of the class. Under  
 21549 certain conditions (printer unavailability, file space limitation, and so on), requests  
 21550 for specific destinations need not be accepted. Destination names vary between  
 21551 systems.

21552 If **-d** is not specified, and neither the *LPDEST* nor *PRINTER* environment variable  
 21553 is set, an unspecified destination is used. The **-d** *dest* option shall take precedence  
 21554 over *LPDEST*, which in turn shall take precedence over *PRINTER*. Results are  
 21555 undefined when *dest* contains a value that is not a valid destination name.

21556 **-m** Send mail (see *mailx* (on page 2794)) after the files have been printed. By default,  
 21557 no mail is sent upon normal completion of the print request.

21558 **-n** *copies* Write *copies* number of copies of the files, where *copies* is a positive decimal integer.  
 21559 The methods for producing multiple copies and for arranging the multiple copies  
 21560 when multiple *file* operands are used are unspecified, except that each file shall be  
 21561 output as an integral whole, not interleaved with portions of other files.

|       |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
|-------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 21562 | <b>-o option</b>             | Specify printer-dependent or class-dependent <i>options</i> . Several such <i>options</i> may be collected by specifying the <b>-o</b> option more than once.                                                                                                                                                                                                                          |
| 21563 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21564 | <b>-s</b>                    | Suppress messages from <i>lp</i> .                                                                                                                                                                                                                                                                                                                                                     |
| 21565 | <b>-t title</b>              | Write <i>title</i> on the banner page of the output.                                                                                                                                                                                                                                                                                                                                   |
| 21566 | <b>-w</b>                    | Write a message on the user's terminal after the files have been printed. If the user is not logged in, then mail shall be sent instead.                                                                                                                                                                                                                                               |
| 21567 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21568 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21569 |                              | The following operand shall be supported:                                                                                                                                                                                                                                                                                                                                              |
| 21570 | <b>file</b>                  | A path name of a file to be output. If no <i>file</i> operands are specified, or if a <i>file</i> operand is '-', the standard input shall be used. If a <i>file</i> operand is used, but the <b>-c</b> option is not specified, the process performing the writing to the output device may have user and group permissions that differ from that of the process invoking <i>lp</i> . |
| 21571 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21572 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21573 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21574 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21575 | <b>STDIN</b>                 |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21576 |                              | The standard input is used only if no <i>file</i> operands are specified, or if a <i>file</i> operand is '-'. See the INPUT FILES section.                                                                                                                                                                                                                                             |
| 21577 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21578 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21579 |                              | The input files shall be text files.                                                                                                                                                                                                                                                                                                                                                   |
| 21580 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21581 |                              | The following environment variables shall affect the execution of <i>lp</i> :                                                                                                                                                                                                                                                                                                          |
| 21582 | <b>LANG</b>                  | Provide a default value for the internationalization variables that are unset or null. If <i>LANG</i> is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined.                         |
| 21583 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21584 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21585 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21586 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21587 | <b>LC_ALL</b>                | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                               |
| 21588 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21589 | <b>LC_CTYPE</b>              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).                                                                                                                                                                                              |
| 21590 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21591 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21592 | <b>LC_MESSAGES</b>           |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21593 |                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.                                                                                                                                                                                                       |
| 21594 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21595 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21596 | <b>LC_TIME</b>               | Determine the format and contents of date and time strings displayed in the <i>lp</i> banner page, if any.                                                                                                                                                                                                                                                                             |
| 21597 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21598 | <b>LPDEST</b>                | Determine the destination. If the <i>LPDEST</i> environment variable is not set, the <i>PRINTER</i> environment variable shall be used. The <b>-d dest</b> option takes precedence over <i>LPDEST</i> . Results are undefined when <b>-d</b> is not specified and <i>LPDEST</i> contains a value that is not a valid destination name.                                                 |
| 21599 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21600 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21601 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21602 | <b>XLSPATH</b>               | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                  |
| 21603 | <b>PRINTER</b>               | Determine the output device or destination. If the <i>LPDEST</i> and <i>PRINTER</i> environment variables are not set, an unspecified output device is used. The <b>-d dest</b> option and the <i>LPDEST</i> environment variable shall take precedence over                                                                                                                           |
| 21604 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |
| 21605 |                              |                                                                                                                                                                                                                                                                                                                                                                                        |

21606                    *PRINTER*. Results are undefined when `-d` is not specified, *LPDEST* is unset, and  
21607                    *PRINTER* contains a value that is not a valid device or destination name.

#### 21608 ASYNCHRONOUS EVENTS

21609            Default.

#### 21610 STDOUT

21611            The *lp* utility shall write a *request ID* to the standard output, unless `-s` is specified. The format of  
21612            the message is unspecified. The request ID can be used on systems supporting the historical  
21613            *cancel* and *lpstat* utilities.

#### 21614 STDERR

21615            Used only for diagnostic messages.

#### 21616 OUTPUT FILES

21617            None.

#### 21618 EXTENDED DESCRIPTION

21619            None.

#### 21620 EXIT STATUS

21621            The following exit values shall be returned:

21622            0 All input files were processed successfully.

21623            >0 No output device was available, or an error occurred.

#### 21624 CONSEQUENCES OF ERRORS

21625            Default.

#### 21626 APPLICATION USAGE

21627            The *pr* and *fold* utilities can be used to achieve reasonable formatting for the implementation's  
21628            default page size.

21629            A portable application can use one of the *file* operands only with the `-c` option or if the file is  
21630            publicly readable and guaranteed to be available at the time of printing. This is because the  
21631            standard gives the implementation the freedom to queue up the request for printing at some  
21632            later time by a different process that might not be able to access the file.

#### 21633 EXAMPLES

21634            1. To print file *file*:

21635                    `lp -c file`

21636            2. To print multiple files with headers:

21637                    `pr file1 file2 | lp`

#### 21638 RATIONALE

21639            The *lp* utility was designed to be a basic version of a utility that is already available in many  
21640            historical implementations. The standard developers considered that it should be implementable  
21641            simply as:

21642                    `cat "$@" > /dev/lp`

21643            after appropriate processing of options, if that is how the implementation chose to do it and if  
21644            exclusive access could be granted (so that two users did not write to the device simultaneously).  
21645            Although in the future the standard developers may add other options to this utility, it should  
21646            always be able to execute with no options or operands and send the standard input to an  
21647            unspecified output device.

21648 This volume of IEEE Std. 1003.1-200x makes no representations concerning the format of the  
 21649 printed output, except that it must be “human-readable” and “non-volatile”. Thus, writing by  
 21650 default to a disk or tape drive or a display terminal would not qualify. (Such destinations are not  
 21651 prohibited when `-d dest`, `LPDEST`, or `PRINTER` are used, however.)

21652 This volume of IEEE Std. 1003.1-200x is worded such that a “print job” consisting of multiple  
 21653 input files, possibly in multiple copies, is guaranteed to print so that any one file is not  
 21654 intermixed with another, but there is no statement that all the files or copies have to print out  
 21655 together.

21656 The `-c` option may imply a spooling operation, but this is not required. The utility can be  
 21657 implemented to wait until the printer is ready and then wait until it is finished. Because of that,  
 21658 there is no attempt to define a queuing mechanism (priorities, classes of output, and so on).

21659 On some historical systems, the request ID reported on the `STDOUT` can be used to later cancel  
 21660 or find the status of a request using utilities not defined in this volume of IEEE Std. 1003.1-200x.

21661 Although the historical System V `lp` and BSD `lpr` utilities have provided similar functionality,  
 21662 they used different names for the environment variable specifying the destination printer. Since  
 21663 the name of the utility here is `lp`, `LPDEST` (used by the System V `lp` utility) was given precedence  
 21664 over `PRINTER` (used by the BSD `lpr` utility). Since environments of users frequently contain one  
 21665 or the other environment variable, the `lp` utility is required to recognize both. If this was not  
 21666 done, many applications would send output to unexpected output devices when users moved  
 21667 from system to system.

21668 Some have commented that `lp` has far too little functionality to make it worthwhile. Requests  
 21669 have proposed additional options or operands or both that added functionality. The requests  
 21670 included:

- 21671 • Wording *requiring* the output to be “hardcopy”
- 21672 • A requirement for multiple printers
- 21673 • Options for supporting various page-description languages

21674 Given that a compliant system is not required to even have a printer, placing further restrictions  
 21675 upon the behavior of the printer is not useful. Since hardcopy format is so application-  
 21676 dependent, it is difficult, if not impossible, to select a reasonable subset of functionality that  
 21677 should be required on all compliant systems.

21678 The term “unspecified” is used in this section in lieu of “implementation-defined” as most  
 21679 known implementations would not be able to make definitive statements in their conformance  
 21680 documents: the existence and usage of printers is very dependent on how the system  
 21681 administrator configures each individual system.

21682 Since the default destination, device type, queuing mechanisms, and acceptable forms of input  
 21683 are all unspecified, usage guidelines for what a portable application can do are as follows:

- 21684 • Use the command in a pipeline, or with `-c`, so that there are no permission problems and the  
 21685 files can be safely deleted or modified.
- 21686 • Limit output to text files of reasonable line lengths and printable characters and include no  
 21687 device-specific formatting information, such as a page description language. The meaning of  
 21688 “reasonable” in this context can only be answered as a quality-of-implementation issue, but  
 21689 it should be apparent from historical usage patterns in the industry and the locale. The `pr` and  
 21690 `fold` utilities can be used to achieve reasonable formatting for the default page size of the  
 21691 implementation.

21692 Alternatively, the application can arrange its installation in such a way that it requires the  
 21693 system administrator or operator to provide the appropriate information on *lp* options and  
 21694 environment variable values.

21695 At a minimum, having this utility in this volume of IEEE Std. 1003.1-200x tells the industry that  
 21696 portable applications require a means to print output and provides at least a command name  
 21697 and *LPDEST* routing mechanism that can be used for discussions between vendors, application  
 21698 writers, and users. The use of “should” in the DESCRIPTION of *lp* clearly shows the intent of  
 21699 the standard developers, even if they cannot mandate that all systems (such as laptops) have  
 21700 printers.

21701 This volume of IEEE Std. 1003.1-200x does not specify what the ownership of the process  
 21702 performing the writing to the output device may be. If *-c* is not used, it is unspecified whether  
 21703 the process performing the writing to the output device has permission to read *file* if there are  
 21704 any restrictions in place on who may read *file* until after it is printed. Also, if *-c* is not used, the  
 21705 results of deleting *file* before it is printed are unspecified.

#### 21706 FUTURE DIRECTIONS

21707 None.

#### 21708 SEE ALSO

21709 *mailx*

#### 21710 CHANGE HISTORY

21711 First released in Issue 2.

#### 21712 Issue 4

21713 Aligned with the ISO/IEC 9945-2: 1993 standard.

#### 21714 Issue 6

21715 The following new requirements on POSIX implementations derive from alignment with the  
 21716 Single UNIX Specification:

- 21717 • In the DESCRIPTION, the requirement to associate a unique request ID, and the normal  
 21718 generation of a banner page is added.
- 21719 • In the OPTIONS section:
  - 21720 — The *-d dest* description is expanded, but references to *lpstat* are removed.
  - 21721 — The *-m*, *-o*, *-s*, *-t*, and *-w* options are added.
- 21722 • In the ENVIRONMENT VARIABLES section, *LC\_TIME* may now affect the execution.
- 21723 • The STDOUT section is added.

21724 The normative text is reworded to avoid use of the term “must” for application requirements.



21725 **NAME**21726        **ls** — list directory contents21727 **SYNOPSIS**21728 xSI        **ls** [-CFRacdilqrutl][-H | -L ][-fgmno~~psx~~][file...]21729 **DESCRIPTION**

21730        For each operand that names a file of a type other than directory or symbolic link to a directory,  
 21731        *ls* shall write the name of the file as well as any requested, associated information. For each  
 21732        operand that names a file of type directory, *ls* shall write the names of files contained within the  
 21733        directory as well as any requested, associated information. If one of the **-d**, **-F**, or **-l** options are  
 21734        specified, and one of the **-H** or **-L** options are not specified, for each operand that names a file of  
 21735        type symbolic link to a directory, *ls* shall write the name of the file as well as any requested,  
 21736        associated information. If none of the **-d**, **-F**, or **-l** options are specified, or the **-H** or **-L** options  
 21737        are specified, for each operand that names a file of type symbolic link to a directory, *ls* shall write  
 21738        the names of files contained within the directory as well as any requested, associated  
 21739        information.

21740        If no operands are specified, *ls* shall write the contents of the current directory. If more than one  
 21741        operand is specified, *ls* shall write non-directory operands first; it shall sort directory and non-  
 21742        directory operands separately according to the collating sequence in the current locale.

21743        The *ls* utility shall detect infinite loops; that is, entering a previously visited directory that is an  
 21744        ancestor of the last file encountered. When it detects an infinite loop, *ls* shall write a diagnostic  
 21745        message to standard error and shall either recover its position in the hierarchy or terminate.

21746 **OPTIONS**

21747        The *ls* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 21748        Utility Syntax Guidelines.

21749        The following options shall be supported:

21750        **-C**        Write multi-text-column output with entries sorted down the columns, according  
 21751        to the collating sequence. The number of text columns and the column separator  
 21752        characters are unspecified, but should be adapted to the nature of the output  
 21753        device.

21754        **-F**        Do not follow symbolic links named as operands unless the **-H** or **-L** options are  
 21755        specified. Write a slash ( '/' ) immediately after each path name that is a directory,  
 21756        an asterisk ( '\*' ) after each that is executable, a vertical bar ( '|' ) after each that is  
 21757        a FIFO, and an at sign ( '@' ) after each that is a symbolic link. For other file types,  
 21758        other symbols may be written.

21759        **-H**        If a symbolic link referencing a file of type directory is specified on the command  
 21760        line, *ls* shall evaluate the file information and file type to be those of the file  
 21761        referenced by the link, and not the link itself; however, *ls* shall write the name of  
 21762        the link itself and not the file referenced by the link.

21763        **-L**        Evaluate the file information and file type for all symbolic links (whether named  
 21764        on the command line or encountered in a file hierarchy) to be those of the file  
 21765        referenced by the link, and not the link itself; however, *ls* shall write the name of  
 21766        the link itself and not the file referenced by the link. When **-L** is used with **-l**, write  
 21767        the contents of symbolic links in the long format (see the STDOUT section).

21768        **-R**        Recursively list subdirectories encountered.

21769        **-a**        Write out all directory entries, including those whose names begin with a period  
 21770        ( '.' ). Entries beginning with a period shall not be written out unless explicitly

|           |           |                                                                                                                                                                            |
|-----------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 21771     |           | referenced, the <b>-a</b> option is supplied, or an implementation-defined condition shall                                                                                 |
| 21772     |           | cause them to be written.                                                                                                                                                  |
| 21773     | <b>-c</b> | Use time of last modification of the file status information (see <b>&lt;sys/stat.h&gt;</b> in the                                                                         |
| 21774     |           | System Interfaces volume of IEEE Std. 1003.1-200x) instead of last modification of                                                                                         |
| 21775     |           | the file itself for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                                        |
| 21776     | <b>-d</b> | Do not follow symbolic links named as operands unless the <b>-H</b> or <b>-L</b> options are                                                                               |
| 21777     |           | specified. Do not treat directories differently than other types of files. The use of <b>-d</b>                                                                            |
| 21778     |           | with <b>-R</b> produces unspecified results.                                                                                                                               |
| 21779 XSI | <b>-f</b> | Force each argument to be interpreted as a directory and list the name found in                                                                                            |
| 21780     |           | each slot. This option shall turn off <b>-l</b> , <b>-t</b> , <b>-s</b> , and <b>-r</b> , and shall turn on <b>-a</b> ; the order                                          |
| 21781     |           | is the order in which entries appear in the directory.                                                                                                                     |
| 21782 XSI | <b>-g</b> | The same as <b>-l</b> , except that the owner shall not be written.                                                                                                        |
| 21783     | <b>-i</b> | For each file, write the file's file serial number (see <i>stat()</i> in the System Interfaces                                                                             |
| 21784     |           | volume of IEEE Std. 1003.1-200x).                                                                                                                                          |
| 21785     | <b>-l</b> | (The letter ell.) Do not follow symbolic links named as operands unless the <b>-H</b> or                                                                                   |
| 21786     |           | <b>-L</b> options are specified. Write out in long format (see the STDOUT section). When                                                                                   |
| 21787     |           | <b>-l</b> (ell) is specified, <b>-1</b> (one) shall be assumed.                                                                                                            |
| 21788 XSI | <b>-m</b> | Stream output format; list files across the page, separated by commas.                                                                                                     |
| 21789 XSI | <b>-n</b> | The same as <b>-l</b> , except that the owner's UID and GID numbers are written, rather                                                                                    |
| 21790     |           | than the associated character strings.                                                                                                                                     |
| 21791 XSI | <b>-o</b> | The same as <b>-l</b> , except that the group is not written.                                                                                                              |
| 21792 XSI | <b>-p</b> | Write a slash ( <code>' / '</code> ) after each file name if that file is a directory.                                                                                     |
| 21793     | <b>-q</b> | Force each instance of non-printable file name characters and <code>&lt;tab&gt;</code> characters to                                                                       |
| 21794     |           | be written as the question-mark ( <code>' ? '</code> ) character. Implementations may provide                                                                              |
| 21795     |           | this option by default if the output is to a terminal device.                                                                                                              |
| 21796     | <b>-r</b> | Reverse the order of the sort to get reverse collating sequence or oldest first.                                                                                           |
| 21797 XSI | <b>-s</b> | Indicate the total number of file system blocks consumed by each file displayed.                                                                                           |
| 21798     |           | The block size is implementation-defined.                                                                                                                                  |
| 21799     | <b>-t</b> | Sort by time modified (most recently modified first) before sorting the operands by                                                                                        |
| 21800     |           | the collating sequence.                                                                                                                                                    |
| 21801     | <b>-u</b> | Use time of last access (see <b>&lt;sys/stat.h&gt;</b> in the System Interfaces volume of                                                                                  |
| 21802     |           | IEEE Std. 1003.1-200x) instead of last modification of the file for sorting ( <b>-t</b> ) or                                                                               |
| 21803     |           | writing ( <b>-l</b> ).                                                                                                                                                     |
| 21804 XSI | <b>-x</b> | The same as <b>-C</b> , except that the multi-text-column output is produced with entries                                                                                  |
| 21805     |           | sorted across, rather than down, the columns.                                                                                                                              |
| 21806     | <b>-1</b> | (The numeric digit one.) Force output to be one entry per line.                                                                                                            |
| 21807     |           | Specifying more than one of the options in the following mutually exclusive pairs shall not be                                                                             |
| 21808 XSI |           | considered an error: <b>-C</b> and <b>-l</b> (ell), <b>-m</b> and <b>-l</b> (ell), <b>-x</b> and <b>-l</b> (ell), <b>-C</b> and <b>-1</b> (one), <b>-H</b> and <b>-L</b> , |
| 21809     |           | <b>-c</b> and <b>-u</b> . The last option specified in each pair shall determine the output format.                                                                        |

21810 **OPERANDS**

21811 The following operand shall be supported:

21812 *file* A path name of a file to be written. If the file specified is not found, a diagnostic  
21813 message shall be output on standard error.

21814 **STDIN**

21815 Not used.

21816 **INPUT FILES**

21817 None.

21818 **ENVIRONMENT VARIABLES**

21819 The following environment variables shall affect the execution of *ls*:

21820 *COLUMNS* Determine the user's preferred column position width for writing multiple text-  
21821 column output. If this variable contains a string representing a decimal integer, the  
21822 *ls* utility shall calculate how many path name text columns to write (see *-C*) based  
21823 on the width provided. If *COLUMNS* is not set or invalid, an implementation-  
21824 defined number of column positions shall be assumed, based on the  
21825 implementation's knowledge of the output device. The column width chosen to  
21826 write the names of files in any given directory shall be constant. File names shall  
21827 not be truncated to fit into the multiple text-column output.

21828 *LANG* Provide a default value for the internationalization variables that are unset or null.  
21829 If *LANG* is unset or null, the corresponding value from the implementation-  
21830 defined default locale shall be used. If any of the internationalization variables  
21831 contains an invalid setting, the utility shall behave as if none of the variables had  
21832 been defined.

21833 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
21834 internationalization variables.

21835 *LC\_COLLATE*

21836 Determine the locale for character collation information in determining the path  
21837 name collation sequence.

21838 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
21839 characters (for example, single-byte as opposed to multi-byte characters in  
21840 arguments) and which characters are defined as printable (character class **print**).

21841 *LC\_MESSAGES*

21842 Determine the locale that should be used to affect the format and contents of  
21843 diagnostic messages written to standard error.

21844 *LC\_TIME* Determine the format and contents for date and time strings written by *ls*.

21845 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

21846 *TZ* Determine the timezone for date and time strings written by *ls*.

21847 **ASYNCHRONOUS EVENTS**

21848 Default.

21849 **STDOUT**

21850 The default format shall be to list one entry per line to standard output; the exceptions are to  
21851 XSI terminals or when one of the *-C*, *-m*, or *-x* options is specified. If the output is to a terminal, the  
21852 format is implementation-defined.

21853 XSI When **-m** is specified, the format used shall be:

21854 "%s, %s, ...\n", <filename1>, <filename2>

21855 where the largest number of file names shall be written without exceeding the length of the line.

21856 If the **-i** option is specified, the file's file serial number (see <sys/stat.h> in the System Interfaces  
21857 volume of IEEE Std. 1003.1-200x) shall be written in the following format before any other  
21858 output for the corresponding entry:

21859 %u ", <file serial number>

21860 If the **-l** option is specified without **-L**, the following information shall be written:

21861 "%s %u %s %s %u %s %s\n", <file mode>, <number of links>,  
21862 <owner name>, <group name>, <number of bytes in the file>,  
21863 <date and time>, <pathname>

21864 If the file is a symbolic link, this information shall be about the link itself and the <pathname>  
21865 field shall be of the form:

21866 "%s -> %s", <pathname of link>, <contents of link>

21867 If both **-l** and **-L** are specified, the following information shall be written:

21868 "%s %u %s %s %u %s %s0, <file mode>, <number of links>,  
21869 <owner name>, <group name>, <number of bytes in the file>,  
21870 <date and time>, <pathname of link>

21871 where all fields except <pathname of link> shall be for the file resolved from the symbolic link.

21872 XSI The **-g**, **-n**, and **-o** options use the same format as **-l**, but with omitted items and their  
21873 associated <blank> characters. See the OPTIONS section.

21874 XSI In both the preceding **-l** forms, If <owner name> or <group name> cannot be determined, or if **-n**  
21875 is given, they shall be replaced with their associated numeric values using the format %u.

21876 The <date and time> field shall contain the appropriate date and timestamp of when the file was  
21877 last modified. In the POSIX locale, the field shall be the equivalent of the output of the following  
21878 date command:

21879 date "+%b %e %H:%M"

21880 if the file has been modified in the last six months, or:

21881 date "+%b %e %Y"

21882 (where two <space> characters are used between %e and %Y) if the file has not been modified in  
21883 the last six months or if the modification date is in the future, except that, in both cases, the final  
21884 <newline> character produced by date shall not be included and the output shall be as if the date  
21885 command were executed at the time of the last modification date of the file rather than the  
21886 current time. When the LC\_TIME locale category is not set to the POSIX locale, a different format  
21887 and order of presentation of this field may be used.

21888 If the file is a character special or block special file, the size of the file may be replaced with  
21889 implementation-defined information associated with the device in question.

21890 If the path name was specified as a file operand, it shall be written as specified.

21891 XSI The file mode written under the **-l**, **-g**, **-n**, and **-o** options shall consist of the following format:

21892 "%c%s%s%c", <entry type>, <owner permissions>,  
21893 <group permissions>, <other permissions>,

- 21894            *<optional alternate access method flag>*
- 21895            The *<optional alternate access method flag>* shall be a single <space> character if there is no  
21896 alternate or additional access control method associated with the file; otherwise, a printable  
21897 character shall be used.
- 21898            The *<entry type>* character shall describe the type of file, as follows:
- 21899            d        Directory.
- 21900            b        Block special file.
- 21901            c        Character special file.
- 21902            l (ell) Symbolic link.
- 21903            p        FIFO.
- 21904            -        Regular file.
- 21905            Implementations may add other characters to this list to represent other implementation-defined  
21906 file types.
- 21907            The next three fields shall be three characters each:
- 21908            *<owner permissions>*  
21909            Permissions for the file owner class (see the Base Definitions volume of  
21910 IEEE Std. 1003.1-200x, Section 4.1, File Access Permissions).
- 21911            *<group permissions>*  
21912            Permissions for the file group class.
- 21913            *<other permissions>*  
21914            Permissions for the file other class.
- 21915            Each field shall have three character positions:
- 21916            1. If 'r', the file is readable; if '-', the file is not readable.
- 21917            2. If 'w', the file is writable; if '-', the file is not writable.
- 21918            3. The first of the following that applies:
- 21919            S        If in *<owner permissions>*, the file is not executable and set-user-ID mode is set. If in  
21920 *<group permissions>*, the file is not executable and set-group-ID mode is set.
- 21921            s        If in *<owner permissions>*, the file is executable and set-user-ID mode is set. If in  
21922 *<group permissions>*, the file is executable and set-group-ID mode is set.
- 21923            x        The file is executable or the directory is searchable.
- 21924            -        None of the attributes of 'S', 's', or 'x' applies.
- 21925            Implementations may add other characters to this list for the third character position. Such  
21926 additions shall, however, be written in lowercase if the file is executable or searchable, and  
21927 in uppercase if it is not.
- 21928 XSI          If any of the **-l**, **-g**, **-n**, **-o**, or **-s** options is specified, each list of files within the directory shall be  
21929 preceded by a status line indicating the number of file system blocks occupied by files in the  
21930 directory in 512-byte units, rounded up to the next integral number of units, if necessary. In the  
21931 POSIX locale, the format shall be:
- 21932            "total %u\n", *<number of units in the directory>*

21933 If more than one directory, or a combination of non-directory files and directories are written,  
21934 either as a result of specifying multiple operands, or the **-R** option, each list of files within a  
21935 directory shall be preceded by:

21936 "\n%s:\n", <directory name>

21937 If this string is the first thing to be written, the first <newline> character shall not be written.  
21938 This output shall precede the number of units in the directory.

21939 XSI If the **-s** option is given, each file shall be written with the number of blocks used by the file.  
21940 Along with **-C**, **-l**, **-m**, or **-x**, the number and a <space> character shall precede the file name;  
21941 with **-g**, **-l**, **-n**, or **-o**, they shall precede each line describing a file.

#### 21942 **STDERR**

21943 Used only for diagnostic messages.

#### 21944 **OUTPUT FILES**

21945 None.

#### 21946 **EXTENDED DESCRIPTION**

21947 None.

#### 21948 **EXIT STATUS**

21949 The following exit values shall be returned:

21950 0 Successful completion.

21951 >0 An error occurred.

#### 21952 **CONSEQUENCES OF ERRORS**

21953 Default.

#### 21954 **APPLICATION USAGE**

21955 Many implementations use the equal sign ('=') and the at sign('@') to denote sockets bound to  
21956 the file system and symbolic links, respectively, for the **-F** option. Similarly, many historical  
21957 implementations use the 's' character and the 'l' character to denote sockets and symbolic  
21958 links, respectively, as the entry type characters for the **-l** option.

21959 It is difficult for an application to use every part of the file modes field of *ls -l* in a portable  
21960 manner. Certain file types and executable bits are not guaranteed to be exactly as shown, as  
21961 implementations may have extensions. Applications can use this field to pass directly to a user  
21962 printout or prompt, but actions based on its contents should generally be deferred, instead, to  
21963 the *test* utility.

21964 The output of *ls* (with the **-l** and related options) contains information that logically could be  
21965 used by utilities such as *chmod* and *touch* to restore files to a known state. However, this  
21966 information is presented in a format that cannot be used directly by those utilities or be easily  
21967 translated into a format that can be used. A character has been added to the end of the  
21968 permissions string so that applications at least have an indication that they may be working in  
21969 an area they do not understand instead of assuming that they can translate the permissions  
21970 string into something that can be used. Future issues or related documents may define one or  
21971 more specific characters to be used based on different standard additional or alternative access  
21972 control mechanisms.

21973 As with many of the utilities that deal with file names, the output of *ls* for multiple files or in one  
21974 of the long listing formats must be used carefully on systems where file names can contain  
21975 embedded white space. Systems and system administrators should institute policies and user  
21976 training to limit the use of such file names.

21977 The number of disk blocks occupied by the file that it reports varies depending on underlying  
 21978 file system type, block size units reported, and the method of calculating the number of blocks.  
 21979 On some file system types, the number is the actual number of blocks occupied by the file  
 21980 (counting indirect blocks and ignoring holes in the file); on others it is calculated based on the  
 21981 file size (usually making an allowance for indirect blocks, but ignoring holes).

#### 21982 EXAMPLES

21983 An example of a small directory tree being fully listed with `ls -laRF a` in the POSIX locale:

```
21984 total 11
21985 drwxr-xr-x 3 hlj prog 64 Jul 4 12:07 ./
21986 drwxrwxrwx 4 hlj prog 3264 Jul 4 12:09 ../
21987 drwxr-xr-x 2 hlj prog 48 Jul 4 12:07 b/
21988 -rwxr--r-- 1 hlj prog 572 Jul 4 12:07 foo*

21989 a/b:
21990 total 4
21991 drwxr-xr-x 2 hlj prog 48 Jul 4 12:07 ./
21992 drwxr-xr-x 3 hlj prog 64 Jul 4 12:07 ../
21993 -rw-r--r-- 1 hlj prog 700 Jul 4 12:07 bar
```

#### 21994 RATIONALE

21995 Some historical implementations of the `ls` utility show all entries in a directory except dot and  
 21996 dot-dot when a superuser invokes `ls` without specifying the `-a` option. When “normal” users  
 21997 invoke `ls` without specifying `-a`, they should not see information about any files with names  
 21998 beginning with period unless they were named as *file* operands.

21999 Implementations are expected to traverse arbitrary depths when processing the `-R` option. The  
 22000 only limitation on depth should be based on running out of physical storage for keeping track of  
 22001 untraversed directories.

22002 The `-1` (one) option is currently found in BSD and BSD-derived implementations only. It is  
 22003 required in this volume of IEEE Std. 1003.1-200x so that portable applications might ensure that  
 22004 output is one entry per line, even if the output is to a terminal.

22005 Generally, this volume of IEEE Std. 1003.1-200x is silent about what happens when options are  
 22006 given multiple times. In the cases of `-C`, `-l`, and `-1`, however, it does specify the results of these  
 22007 overlapping options. Since `ls` is one of the most aliased commands, it is important that the  
 22008 implementation perform intuitively. For example, if the alias were:

```
22009 alias ls="ls -C"
```

22010 and the user typed `ls -1`, single-text-column output should result, not an error.

22011 The BSD `ls` provides a `-A` option (like `-a`, but dot and dot-dot are not written out). The small  
 22012 difference from `-a` did not seem important enough to require both.

22013 Implementations are allowed to make `-q` the default for terminals to prevent trojan horse  
 22014 attacks on terminals with special escape sequences. This is not required because:

- 22015 • Some control characters may be useful on some terminals; for example, a system might write  
 22016 them as `"\001"` or `"^A"`.

- 22017 • Special behavior for terminals is not relevant to application portability.

22018 An early proposal specified that the optional alternate access method flag had to be `'+'` if there  
 22019 was an alternate access method used on the file or `<space>` if there was not. This was changed to  
 22020 be `<space>` if there is not and a single printable character if there is. This was done for three  
 22021 reasons:

- 22022 1. There are historical implementations using characters other than '+'.
- 22023 2. There are implementations that vary this character used in that position to distinguish  
22024 between various alternate access methods in use.
- 22025 3. The standard developers did not want to preclude futures specifications that might need a  
22026 way to specify more than one alternate access method.
- 22027 Nonetheless, implementations providing a single alternate access method are encouraged to use  
22028 '+'.
- 22029 In an early proposal, the units used to specify the number of blocks occupied by files in a  
22030 directory in an *ls -l* listing was implementation-defined. This was because BSD systems have  
22031 historically used 1 024-byte units and System V systems have historically used 512-byte units. It  
22032 was pointed out by BSD developers that their system has used 512-byte units in some places and  
22033 1 024-byte units in other places. (System V has consistently used 512.) Therefore, this volume of  
22034 IEEE Std. 1003.1-200x usually specifies 512. Future releases of BSD are expected to consistently  
22035 provide 512 bytes as a default with a way of specifying 1 024-byte units where appropriate.
- 22036 The *<date and time>* field in the *-l* format is specified only for the POSIX locale. As noted, the  
22037 format can be different in other locales. No mechanism for defining this is present in this volume  
22038 of IEEE Std. 1003.1-200x, as the appropriate vehicle is a messaging system; that is, the format  
22039 should be specified as a "message".
- 22040 **FUTURE DIRECTIONS**
- 22041 The *-s* uses implementation-defined units and cannot be used portably; it may be withdrawn in  
22042 a future issue.
- 22043 **SEE ALSO**
- 22044 *chmod, find*, the System Interfaces volume of IEEE Std. 1003.1-200x, *<sys/stat.h>*
- 22045 **CHANGE HISTORY**
- 22046 First released in Issue 2.
- 22047 **Issue 4**
- 22048 Aligned with the ISO/IEC 9945-2: 1993 standard.
- 22049 **Issue 5**
- 22050 Second FUTURE DIRECTION added.
- 22051 **Issue 6**
- 22052 The following new requirements on POSIX implementations derive from alignment with the  
22053 Single UNIX Specification:
- 22054 • In the *-F* option, other symbols are allowed for other file types.
- 22055 Treatment of symbolic links is added, as defined in the IEEE P1003.2b draft standard.



22056 **NAME**22057 m4 — macro processor (**DEVELOPMENT**)22058 **SYNOPSIS**22059 xSI m4 [-s][-D *name*[=*val*]]...[-U *name*]... *file*...

22060

22061 **DESCRIPTION**22062 The *m4* utility is a macro processor that shall read one or more text files, process them according  
22063 to their included macro statements, and write the results to standard output.22064 **OPTIONS**22065 The *m4* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
22066 12.2, Utility Syntax Guidelines, except that the order of the **-D** and **-U** options shall be  
22067 significant.

22068 The following options shall be supported:

22069 **-s** Enable line synchronization output for the *c99* preprocessor phase (that is, **#line**  
22070 directives).22071 **-D *name*[=*val*]**  
22072 Define *name* to *val* or to null if *=val* is omitted.22073 **-U *name*** Undefine *name*.22074 **OPERANDS**

22075 The following operand shall be supported:

22076 *file* A path name of a text file to be processed. If no *file* is given, or if it is '-', the  
22077 standard input shall be read.22078 **STDIN**22079 The standard input shall be a text file that is used if no *file* operand is given, or if it is '-'.22080 **INPUT FILES**22081 The input file named by the *file* operand shall be a text file.22082 **ENVIRONMENT VARIABLES**22083 The following environment variables shall affect the execution of *m4*:22084 *LANG* Provide a default value for the internationalization variables that are unset or null.  
22085 If *LANG* is unset or null, the corresponding value from the implementation-  
22086 defined default locale shall be used. If any of the internationalization variables  
22087 contains an invalid setting, the utility shall behave as if none of the variables had  
22088 been defined.22089 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
22090 internationalization variables.22091 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
22092 characters (for example, single-byte as opposed to multi-byte characters in  
22093 arguments and input files).22094 *LC\_MESSAGES*  
22095 Determine the locale that should be used to affect the format and contents of  
22096 diagnostic messages written to standard error.22097 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

22098 **ASYNCHRONOUS EVENTS**

22099 Default.

22100 **STDOUT**22101 The standard output shall be the same as the input files, after being processed for macro  
22102 expansion.22103 **STDERR**22104 Used to display strings with the **errprint** macro, macro tracing enabled by the **traceon** macro, the  
22105 defined text for macros written by the **dumpdef** macro, or for diagnostic messages.22106 **OUTPUT FILES**

22107 None.

22108 **EXTENDED DESCRIPTION**22109 The *m4* utility shall compare each token from the input against the set of built-in and user-  
22110 defined macros. If the token matches the name of a macro, then the token shall be replaced by  
22111 the macros defining text, if any, and rescanned for matching macro names. Once no portion of  
22112 the token matches the name of a macro, it shall be written to standard output. Macros may have  
22113 arguments, in which case the arguments shall be substituted into the defining text before it is  
22114 rescanned.

22115 Macro calls have the form:

22116 *name(arg1, arg2, ..., argn)*22117 Macro names shall consist of letters, digits, and underscores, where the first character is not a  
22118 digit. Tokens not of this form shall not be treated as macro names.22119 The application shall ensure that the left parenthesis immediately follows the name of the  
22120 macro. If a token matching the name of a macro is not followed by a left parenthesis, it is  
22121 handled as a use of that macro without arguments.22122 If a macro name is followed by a left parenthesis, its arguments are the comma-separated tokens  
22123 between the left parenthesis and the matching right parenthesis. Unquoted <blank> and  
22124 <newline> characters preceding each argument shall be ignored. All other characters, including  
22125 trailing <blank> and <newline> characters, are retained. Commas enclosed between left and  
22126 right parenthesis characters do not delimit arguments.22127 Arguments are positionally defined and referenced. The string "\$1" in the defining text shall be  
22128 replaced by the first argument. Systems shall support at least nine arguments; only the first nine  
22129 can be referenced, using the strings "\$1" to "\$9", inclusive. The string "\$0" is replaced with  
22130 the name of the macro. The string "\$#" is replaced by the number of arguments as a string. The  
22131 string "\$\*" is replaced by a list of all of the arguments, separated by commas. The string "\$@"  
22132 is replaced by a list of all of the arguments separated by commas, and each argument is quoted  
22133 using the current left and right quoting strings.22134 If fewer arguments are supplied than are in the macro definition, the omitted arguments are  
22135 taken to be null. It is not an error if more arguments are supplied than are in the macro  
22136 definition.22137 No special meaning is given to any characters enclosed between matching left and right quoting  
22138 strings, but the quoting strings are themselves discarded. By default, the left quoting string  
22139 consists of a grave accent (``) and the right quoting string consists of an acute accent (') see  
22140 also the **changequote** macro.22141 Comments are written but not scanned for matching macro names; by default, the begin-  
22142 comment string consists of the number sign character and the end-comment string consists of a  
22143 <newline> character. See also the **changecom** and **dnl** macros.

- 22144 The *m4* utility makes available the following built-in macros. They can be redefined, but once  
 22145 this is done the original meaning is lost. Their values are null unless otherwise stated. In the  
 22146 descriptions below, the term *defining text* refers to the value of the macro: the second argument  
 22147 to the **define** macro, among other things. Except for the first argument to the **eval** macro, all  
 22148 numeric built-in macro arguments shall be interpreted as decimal values. The string values  
 22149 produced as the defining text of the **decr**, **divnum**, **incr**, **index**, **len**, and **sysval** built-in macros  
 22150 shall be in the form of a decimal-constant as defined in the C language.
- 22151 **changeocom** The **changeocom** macro sets the begin-comment and end-comment strings. With no  
 22152 arguments, the comment mechanism is disabled. With a single argument, that  
 22153 argument becomes the begin-comment string and the <newline> character  
 22154 becomes the end-comment string. With two arguments, the first argument  
 22155 becomes the begin-comment string and the second argument becomes the end-  
 22156 comment string. Systems support comment strings of at least five characters.
- 22157 **changequote** The **changequote** macro sets the begin-quote and end-quote strings. With no  
 22158 arguments, the quote strings are set to the default values (that is, ` `'). With a  
 22159 single argument, that argument becomes the begin-quote string and the <newline>  
 22160 character becomes the end-quote string. With two arguments, the first argument  
 22161 becomes the begin-quote string and the second argument becomes the end-quote  
 22162 string. Systems support quote strings of at least five characters.
- 22163 **decr** The defining text of the **decr** macro is its first argument decremented by 1. It is an  
 22164 error to specify an argument containing any non-numeric characters.
- 22165 **define** The second argument is specified as the defining text of the macro whose name is  
 22166 the first argument.
- 22167 **defn** The defining text of the **defn** macro is the quoted definition (using the current  
 22168 quoting strings) of its arguments.
- 22169 **divert** The *m4* utility maintains ten temporary buffers, numbered 0 to 9, inclusive.

## 22170 *Notes to Reviewers*

- 22171 *This section with side shading will not appear in the final copy. - Ed.*
- 22172 Re D1, XCU, ERN 286: Buffer 0 seems strange: it's one of the 10 buffers, and thus  
 22173 should be a diversion buffer, but at 19704 it implies that it's the name of the main  
 22174 output. What is it (or are there really only 9 diversion buffers?) Also, see austin-  
 22175 group mail sequence #295.
- 22176 When the last of the input has been processed, any output that has been placed in  
 22177 these buffers is written to standard output in buffer-numerical order. The **divert**  
 22178 macro diverts future output to the buffer specified by its argument. Specifying no  
 22179 argument or an argument of 0 resumes the normal output process. Output  
 22180 diverted to a stream other than 0 to 9 is discarded. It is an error to specify an  
 22181 argument containing any non-numeric characters.
- 22182 **divnum** The defining text of the **divnum** macro is the number of the current output stream  
 22183 as a string.
- 22184 **dnl** The **dnl** macro shall cause *m4* to discard all input characters up to and including  
 22185 the next <newline> character.
- 22186 **dumpdef** The **dumpdef** macro writes the defined text to standard error for each of the  
 22187 macros specified as arguments, or, if no arguments are specified, for all macros.

|       |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22188 | <b>errprint</b>           | The <b>errprint</b> macro writes its arguments to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 22189 | <b>eval</b>               | The <b>eval</b> macro evaluates its first argument as an arithmetic expression, using 32-bit signed integer arithmetic. All of the C-language operators are supported, except for:                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22190 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22191 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22192 |                           | [ ]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22193 |                           | ->                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22194 |                           | ++                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22195 |                           | --                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22196 |                           | ( <i>type</i> )                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22197 |                           | unary *                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 22198 |                           | <i>sizeof</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 22199 |                           | ,                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 22200 |                           | .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 22201 |                           | ?:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22202 |                           | unary &                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 22203 |                           | and all assignment operators. It is an error to specify any of these operators.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22204 |                           | Precedence and associativity are as in C. Systems support octal and hexadecimal numbers as in C. The second argument, if specified, sets the radix for the result; the default is 10. The third argument, if specified, sets the minimum number of digits in the result. It is an error to specify the second or third argument containing any non-numeric characters.                                                                                                                                                                                                                                                                  |
| 22205 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22206 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22207 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22208 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22209 | <b>ifdef</b>              | If the first argument to the <b>ifdef</b> macro is defined, the defining text is the second argument. Otherwise, the defining text is the third argument, if specified, or the null string, if not.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22210 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22211 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22212 | <b>ifndef</b>             | If the first argument (or the defining text of the first argument if it is a macro name) to the <b>ifndef</b> macro is the same as the second argument (or the defining text of the second argument if it is a macro name), then the defining text is the third argument.                                                                                                                                                                                                                                                                                                                                                               |
| 22213 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22214 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22215 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22216 | <b>Notes to Reviewers</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22217 |                           | <i>This section with side shading will not appear in the final copy. - Ed.</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 22218 |                           | D1, XCU, ERN 287 (as modified by email #297) suggests the following replacement text for <b>ifelse</b> : "This function takes 3n+0 or 3n+1 arguments. For each group of 3 arguments, if the first and second are the same, the result is the third of the group. If the strings are not equal, and no arguments remain, the defining text is null. If one argument remains, it becomes the defining text. If three or more arguments remain, the process is repeated with the new group of three arguments. If 3n+2 arguments are provided, the evaluation proceeds as above, but a warning is generated and the last argument ignored. |
| 22219 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22220 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22221 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22222 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22223 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22224 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22225 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22226 |                           | If there are more than four arguments, the initial comparison of the first and second arguments are repeated for each group of three arguments. If no match is found, the defining text is the argument following the last set of three compared; otherwise, it is null.                                                                                                                                                                                                                                                                                                                                                                |
| 22227 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22228 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22229 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22230 | <b>include</b>            | The defining text for the <b>include</b> macro is the contents of the file named by the first argument. It is an error if the file cannot be read.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22231 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22232 | <b>incr</b>               | The defining text of the <b>incr</b> macro is its first argument incremented by 1. It is an error to specify an argument containing any non-numeric characters.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22233 |                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

|       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22234 | <b>index</b>    | The defining text of the <b>index</b> macro is the first character position (as a string) in the first argument where a string matching the second argument begins (zero origin), or -1 if the second argument does not occur.                                                                                                                                                                                                                                                                                                                                      |
| 22235 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22236 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22237 | <b>len</b>      | The defining text of the <b>len</b> macro is the length (as a string) of the first argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 22238 | <b>m4exit</b>   | Exit from the <i>m4</i> utility. If the first argument is specified, it is the exit code. The default is zero. It is an error to specify an argument containing any non-numeric characters.                                                                                                                                                                                                                                                                                                                                                                         |
| 22239 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22240 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22241 | <b>m4wrap</b>   | The first argument is processed when EOF is reached. If the <b>m4wrap</b> macro is used multiple times, the arguments specified are processed in the order in which the <b>m4wrap</b> macros were processed.                                                                                                                                                                                                                                                                                                                                                        |
| 22242 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22243 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22244 | <b>maketemp</b> | The defining text is the first argument, with any trailing 'x' characters replaced with the current process ID as a string.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22245 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22246 | <b>popdef</b>   | The <b>popdef</b> macro deletes the current definition of its arguments, replacing that definition with the previous one. If there is no previous definition, the macro is undefined.                                                                                                                                                                                                                                                                                                                                                                               |
| 22247 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22248 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22249 | <b>pushdef</b>  | The <b>pushdef</b> macro is identical to the <b>define</b> macro with the exception that it preserves any current definition for future retrieval using the <b>popdef</b> macro.                                                                                                                                                                                                                                                                                                                                                                                    |
| 22250 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22251 | <b>shift</b>    | The defining text for the <b>shift</b> macro is all of its arguments except for the first one.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 22252 | <b>sinclude</b> | The <b>sinclude</b> macro is identical to the <b>include</b> macro, except that it is not an error if the file is inaccessible.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22253 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22254 | <b>substr</b>   | The defining text for the <b>substr</b> macro is the substring of the first argument beginning at the zero-offset character position specified by the second argument. The third argument, if specified, is the number of characters to select; if not specified, the characters from the starting point to the end of the first argument become the defining text. It is not an error to specify a starting point beyond the end of the first argument and the defining text is null. It is an error to specify an argument containing any non-numeric characters. |
| 22255 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22256 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22257 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22258 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22259 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22260 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22261 | <b>syscmd</b>   | The <b>syscmd</b> macro interprets its first argument as a shell command line. The defining text is the string result of that command. No output redirection is performed by the <i>m4</i> utility. The exit status value from the command can be retrieved using the <b>sysval</b> macro.                                                                                                                                                                                                                                                                          |
| 22262 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22263 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22264 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22265 | <b>sysval</b>   | The defining text of the <b>sysval</b> macro is the exit value of the utility last invoked by the <b>syscmd</b> macro (as a string).                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 22266 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22267 | <b>traceon</b>  | The <b>traceon</b> macro enables tracing for the macros specified as arguments, or, if no arguments are specified, for all macros. The trace output is written to standard error in an unspecified format.                                                                                                                                                                                                                                                                                                                                                          |
| 22268 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22269 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22270 | <b>traceoff</b> | The <b>traceoff</b> macro disables tracing for the macros specified as arguments, or, if no arguments are specified, for all macros.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 22271 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22272 | <b>translit</b> | The defining text of the <b>translit</b> macro is the first argument with every character that occurs in the second argument replaced with the corresponding character from the third argument.                                                                                                                                                                                                                                                                                                                                                                     |
| 22273 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22274 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 22275 | <b>undefine</b> | The <b>undefine</b> macro deletes all definitions (including those preserved using the <b>pushdef</b> macro) of the macros named by its arguments.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22276 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

22277        **undivert**     The **undivert** macro shall cause immediate output of any text in temporary buffers  
 22278                    named as arguments, or all temporary buffers if no arguments are specified.  
 22279                    Buffers can be undiverted into other temporary buffers. Undiverting discards the  
 22280                    contents of the temporary buffer. It is an error to specify an argument containing  
 22281                    any non-numeric characters.

#### 22282 **EXIT STATUS**

22283            The following exit values shall be returned:

22284            0    Successful completion.

22285            >0  An error occurred

22286            If the **m4exit** macro is used, the exit value can be specified by the input file.

#### 22287 **CONSEQUENCES OF ERRORS**

22288            Default.

#### 22289 **APPLICATION USAGE**

22290            The **defn** macro is useful for renaming macros, especially built-ins.

#### 22291 **EXAMPLES**

22292            An example of a single *m4* input file capable of generating two output files follows. The file  
 22293            **file1.m4** could contain lines such as:

22294            if(VER, 1, *do\_something*)

22295            if(VER, 2, *do\_something*)

22296            The makefile for the program might include:

22297            file1.1.c : file1.m4

22298                    m4 -D VER=1 file1.m4 > file1.1.c

22299                    ...

22300            file1.2.c : file1.m4

22301                    m4 -D VER=2 file1.m4 > file1.2.c

22302                    ...

22303            The **-U** option can be used to undefine **VER**. If **file1.m4** contains:

22304            if(VER, 1, *do\_something*)

22305            if(VER, 2, *do\_something*)

22306            ifndef(VER, *do\_something*)

22307            then the makefile would contain:

22308            file1.0.c : file1.m4

22309                    m4 -U VER file1.m4 > file1.0.c

22310                    ...

22311            file1.1.c : file1.m4

22312                    m4 -D VER=1 file1.m4 > file1.1.c

22313                    ...

22314            file1.2.c : file1.m4

22315                    m4 -D VER=2 file1.m4 > file1.2.c

22316                    ...

#### 22317 **RATIONALE**

22318            None.

22319 **FUTURE DIRECTIONS**

22320 None.

22321 **SEE ALSO**22322 *c99*22323 **CHANGE HISTORY**

22324 First released in Issue 2.

22325 **Issue 4**

22326 Format reorganized.

22327 Utility Syntax Guideline support mandated.

22328 Internationalized environment variable support mandated.

22329 **Issue 5**

22330 The phrase “the defined text for macros written by the **dumpdef** macro” is added to the  
22331 description of **STDERR**, and the description of **dumpdef** is updated to indicate that output is  
22332 written to standard error. The description of **eval** is updated to indicate that the list of excluded  
22333 C operators excludes unary **&** and **..**. In the description of **ifdef**, the phrase “and it is not  
22334 defined to be zero” is deleted.

22335 **Issue 6**

22336 In the **EXTENDED DESCRIPTION**, the **eval** text is updated to include a **&** character in the  
22337 excepted list.

22338 The normative text is reworded to avoid use of the term “must” for application requirements.

22339 The Open Group Base Resolution bwg2000-006 is applied.

## 22340 NAME

22341 mailx — process messages

## 22342 SYNOPSIS

22343 **Send Mode**22344 mailx [-s *subject*] *address...*22345 **Receive Mode**

22346 mailx -e

22347 mailx [-HiNn][-F][-u *user*]22348 mailx -f[-HiNn][-F][*file*]

## 22349 DESCRIPTION

22350 The *mailx* utility provides a message sending and receiving facility. It has two major modes,  
 22351 selected by the options used: Send Mode and Receive Mode.

22352 On systems that do not support the User Portability Utilities option, an application using *mailx*  
 22353 shall have the ability to send messages in an unspecified manner (Send Mode). Unless the first  
 22354 character of one or more lines is tilde ('~'), all characters in the input message shall appear in  
 22355 the delivered message, but additional characters may be inserted in the message before it is  
 22356 retrieved.

22357 On systems supporting the User Portability Utilities option, mail-receiving capabilities and other  
 22358 interactive features, Receive Mode, described below, also shall be enabled.

22359 **Send Mode**

22360 Send Mode can be used by applications or users to send messages from the text in standard  
 22361 input.

22362 **Receive Mode**

22363 Receive Mode is more oriented to interactive users. Mail can be read and sent in this interactive  
 22364 mode.

22365 When reading mail, *mailx* provides commands to facilitate saving, deleting, and responding to  
 22366 messages. When sending mail, *mailx* allows editing, reviewing, and other modification of the  
 22367 message as it is entered.

22368 Incoming mail shall be stored in one or more unspecified locations for each user, collectively  
 22369 called the system *mailbox* for that user. When *mailx* is invoked in Receive Mode, the system  
 22370 mailbox shall be the default place to find new mail. As messages are read, they shall be marked  
 22371 to be moved to a secondary file for storage, unless specific action is taken. This secondary file is  
 22372 called the **mbox** and is normally located in the directory referred to by the *HOME* environment  
 22373 variable (see *MBOX* in the ENVIRONMENT VARIABLES section for a description of this file).  
 22374 Messages shall remain in this file until explicitly removed. When the **-f** option is used to read  
 22375 mail messages from secondary files, messages shall be retained in those files unless specifically  
 22376 removed. All three of these locations—system mailbox, **mbox**, and secondary file—are referred  
 22377 to in this section as simply “mailboxes”, unless more specific identification is required.



22378 **OPTIONS**

22379 The *mailx* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
22380 12.2, Utility Syntax Guidelines.

22381 The following options shall be supported. (Only the **-s subject** option shall be required on all  
22382 systems. The other options are required only on systems supporting the User Portability Utilities  
22383 option.)

22384 **-e** Test for the presence of mail in the system mailbox. The *mailx* utility shall write  
22385 nothing and exit with a successful return code if there is mail to read.

22386 **-f** Read messages from the file named by the *file* operand instead of the system  
22387 mailbox. (See also **folder**.) If no *file* operand is specified, read messages from the  
22388 **mbox** instead of the system mailbox.

22389 **-F** Record the message in a file named after the first recipient. The name is the login-  
22390 name portion of the address found first on the **To:** line in the mail header.  
22391 Overrides the **record** variable, if set (see **Internal Variables in mailx** (on page  
22392 2801).)

22393 **-H** Write a header summary only.

22394 **-i** Ignore interrupts. (See also **ignore**).

22395 **-n** Do not initialize from the system default start-up file. See the EXTENDED  
22396 DESCRIPTION section.

22397 **-N** Do not write an initial header summary.

22398 **-s subject** Set the **Subject** header field to *subject*. All characters in the *subject* string shall  
22399 appear in the delivered message. The results are unspecified if *subject* is longer  
22400 than {LINE\_MAX} – 10 bytes or contains a <newline> character.

22401 **-u user** Read the system mailbox of the login name *user*. This shall only be successful if  
22402 the invoking user has the appropriate privileges to read the system mailbox of that  
22403 user.

22404 **OPERANDS**

22405 The following operands shall be supported:

22406 *address* Addressee of message. When **-n** is specified and no user start-up files are accessed  
22407 (see the EXTENDED DESCRIPTION section), the user or application shall ensure  
22408 this is an address to pass to the mail delivery system. Any system or user start-up  
22409 files may enable aliases (see **alias** under **Commands in mailx** (on page 2804)) that  
22410 may modify the form of *address* before it is passed to the mail delivery system.

22411 *file* A path name of a file to be read instead of the system mailbox when **-f** is specified.  
22412 The meaning of the *file* option-argument shall be affected by the contents of the  
22413 **folder** internal variable; see **Internal Variables in mailx** (on page 2801).

22414 **STDIN**

22415 When *mailx* is invoked in Send Mode (the first synopsis line), standard input shall be the  
22416 message to be delivered to the specified addresses. When in Receive Mode, user commands are  
22417 accepted from *stdin*. If the User Portability Utilities option is not supported, standard input lines  
22418 beginning with a tilde ('~') character produce unspecified results.

22419 If the User Portability Utilities option is supported, then in both Send and Receive Modes,  
22420 standard input lines beginning with the escape character (usually tilde ('~')) affect processing  
22421 as described in **Command Escapes in mailx** (on page 2812).

## 22422 INPUT FILES

22423 When *mailx* is used as described by this volume of IEEE Std. 1003.1-200x, the *file* option-  
 22424 argument (see the *-f* option) and the **mbox** shall be text files containing mail messages,  
 22425 formatted as described in the OUTPUT FILES section. The nature of the system mailbox is  
 22426 unspecified; it need not be a file.

## 22427 ENVIRONMENT VARIABLES

22428 The following environment variables shall affect the execution of *mailx*:

22429 **DEAD** Determine the path name of the file in which to save partial messages in case of  
 22430 interrupts or delivery errors. The default shall be **dead.letter** in the directory  
 22431 named by the *HOME* variable. The behavior of *mailx* in saving partial messages is  
 22432 unspecified if the User Portability Utilities option is not supported and *DEAD* is  
 22433 not defined with the value **/dev/null**.

22434 **EDITOR** Determine the name of a utility to invoke when the **edit** (see **Commands in mailx**  
 22435 (on page 2804)) or **~e** (see **Command Escapes in mailx** (on page 2812)) command is  
 22436 XSI used. The default editor is unspecified. On XSI-conformant systems it is *ed*. The  
 22437 effects of this variable are unspecified if the User Portability Utilities option is not  
 22438 supported.

22439 **HOME** Determine the path name of the user's home directory.

22440 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 22441 If *LANG* is unset or null, the corresponding value from the implementation-  
 22442 defined default locale shall be used. If any of the internationalization variables  
 22443 contains an invalid setting, the utility shall behave as if none of the variables had  
 22444 been defined.

22445 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 22446 internationalization variables.

22447 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 22448 characters (for example, single-byte as opposed to multi-byte characters in  
 22449 arguments and input files) and the handling of case-insensitive address and  
 22450 header-field comparisons.

22451 **LC\_TIME** Determine the format and contents of the date and time strings written by *mailx*.

## 22452 LC\_MESSAGES

22453 Determine the locale that should be used to affect the format and contents of  
 22454 diagnostic messages written to standard error and informative messages written to  
 22455 standard output.

22456 **LISTER** Determine a string representing the command for writing the contents of the  
 22457 **folder** directory to standard output when the **folders** command is given (see  
 22458 **folders** in **Commands in mailx** (on page 2804)). Any string acceptable as a  
 22459 *command\_string* operand to the *sh -c* command shall be valid. If this variable is null  
 22460 or not set, the output command shall be *ls*. The effects of this variable are  
 22461 unspecified if the User Portability Utilities option is not supported.

22462 **MAILRC** Determine the path name of the start-up file. The default shall be **.mailrc** in the  
 22463 directory referred to by the *HOME* environment variable. The behavior of *mailx* is  
 22464 unspecified if the User Portability Utilities option is not supported and *MAILRC* is  
 22465 not defined with the value **/dev/null**.

22466 **MBOX** Determine a path name of the file to save messages from the system mailbox that  
 22467 have been read. The **exit** command shall override this function, as shall saving the

- 22468 message explicitly in another file. The default shall be **mbox** in the directory  
 22469 named by the *HOME* variable. The effects of this variable are unspecified if the  
 22470 User Portability Utilities option is not supported.
- 22471 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 22472 **PAGER** Determine a string representing an output filtering or pagination command for  
 22473 writing the output to the terminal. Any string acceptable as a *command\_string*  
 22474 operand to the *sh -c* command shall be valid. When standard output is a terminal  
 22475 device, the message output shall be piped through the command if the *mailx*  
 22476 internal variable **crf** is set to a value less the number of lines in the message; see  
 22477 **Internal Variables in mailx** (on page 2801). If the *PAGER* variable is null or not  
 22478 set, the paginator shall be either *more* or another paginator utility documented in  
 22479 the system documentation. The effects of this variable are unspecified if the User  
 22480 Portability Utilities option is not supported.
- 22481 **SHELL** Determine the name of a preferred command interpreter. The default shall be *sh*.  
 22482 The effects of this variable are unspecified if the User Portability Utilities option is  
 22483 not supported.
- 22484 **TERM** Determine the name of the terminal type, to indicate in an unspecified manner, if  
 22485 the internal variable **screen** is not specified, the number of lines in a screenful of  
 22486 headers. If *TERM* is not set or is set to null, an unspecified default terminal type  
 22487 shall be used and the value of a screenful is unspecified. The effects of this variable  
 22488 are unspecified if the User Portability Utilities option is not supported.
- 22489 **VISUAL** Determine a path name of a utility to invoke when the **visual** command (see  
 22490 **Commands in mailx** (on page 2804)) or *~v* command-escape (see **Command**  
 22491 **Escapes in mailx** (on page 2812)) is used. If this variable is null or not set, the full-  
 22492 screen editor shall be *vi*. The effects of this variable are unspecified if the User  
 22493 Portability Utilities option is not supported.
- 22494 **ASYNCHRONOUS EVENTS**
- 22495 When *mailx* is in Send Mode and standard input is not a terminal, it shall take the standard  
 22496 action for all signals.
- 22497 In Receive Mode, or in Send Mode when standard input is a terminal, if a SIGINT signal is  
 22498 received:
- 22499 1. If in command mode, the current command, if there is one, shall be aborted, and a  
 22500 command-mode prompt shall be written.
  - 22501 2. If in input mode:
    - 22502 a. If **ignore** is set, *mailx* shall write "@\n", discard the current input line, and continue  
 22503 processing, bypassing the message-abort mechanism described in item 2b.
    - 22504 b. If the interrupt was received while sending mail, either when in Receive Mode or in  
 22505 Send Mode, a message shall be written, and another subsequent interrupt, with no  
 22506 other intervening characters typed, shall be required to abort the mail message. If in  
 22507 Receive Mode and another interrupt is received, a command-mode prompt shall be  
 22508 written. If in Send Mode and another interrupt is received, *mailx* shall terminate with  
 22509 a non-zero status.
- 22510 In both cases listed in item b, if the message is not empty:
- 22511 i. If **save** is enabled and the file named by *DEAD* can be created, the message  
 22512 shall be written to the file named by *DEAD*. If the file exists, the message shall  
 22513 be written to replace the contents of the file.

22514                   ii. If **save** is not enabled, or the file named by *DEAD* cannot be created, the  
22515                   message shall not be saved.

22516           The *mailx* utility shall take the standard action for all other signals.

#### 22517 **STDOUT**

22518           In command and input modes, all output, including prompts and messages, shall be written to  
22519           standard output.

#### 22520 **STDERR**

22521           Used only for diagnostic messages.

#### 22522 **OUTPUT FILES**

22523           Various *mailx* commands and command escapes can create or add to files, including the **mbox**,  
22524           the dead-letter file, and secondary mailboxes. When *mailx* is used as described in this volume of  
22525           IEEE Std. 1003.1-200x, these files shall be text files, formatted as follows:

22526           line beginning with **From**<space>  
22527           [one or more *header-lines*; see **Commands in mailx** (on page 2804) ]  
22528           *empty line*  
22529           [zero or more *body lines*  
22530           *empty line*]  
22531           [line beginning with **From**<space>...]

22532           where each message begins with the **From** <space> line shown, preceded by the beginning of  
22533           the file or an empty line. (The **From** <space> line is considered to be part of the message header,  
22534           but not one of the header-lines referred to in **Commands in mailx** (on page 2804); thus, it shall  
22535           not be affected by the **discard**, **ignore**, or **retain** commands.) The formats of the remainder of the  
22536           **From** <space> line and any additional header lines are unspecified, except that none shall be  
22537           empty. The format of a message body line is also unspecified, except that no line following an  
22538           empty line shall start with **From** <space>; *mailx* shall modify any such user-entered message  
22539           body lines (following an empty line and beginning with **From** <space>) by adding one or more  
22540           characters to precede the 'F'; it may add these characters to **From** <space> lines that are not  
22541           preceded by an empty line.

22542           When a message from the system mailbox or entered by the user is not a text file, it is  
22543           implementation-defined how such a message is stored in files written by *mailx*.

#### 22544 **EXTENDED DESCRIPTION**

22545           The entire EXTENDED DESCRIPTION section shall apply only to implementations supporting  
22546           the User Portability Utilities option.

22547           The *mailx* utility cannot guarantee support for all character encodings in all circumstances. For  
22548           example, inter-system mail may be restricted to 7-bit data by the underlying network, 8-bit data  
22549           need not be portable to non-internationalized systems, and so on. Under these circumstances, it  
22550           is recommended that only characters defined in the ISO/IEC 646:1991 standard International  
22551           Reference Version (equivalent to ASCII) 7-bit range of characters be used.

22552           When *mailx* is invoked using one of the Receive Mode synopsis forms, it shall write a page of  
22553           header-summary lines (if **-N** was not specified and there are messages, see below), followed by  
22554           a prompt indicating that *mailx* can accept regular commands (see **Commands in mailx** (on page  
22555           2804)); this is termed *command mode*. The page of header-summary lines shall contain the first  
22556           new message if there are new messages, or the first unread message if there are unread  
22557           messages, or the first message. When *mailx* is invoked using the Send Mode synopsis and  
22558           standard input is a terminal, if no subject is specified on the command line and the **asksub**  
22559           variable is set, a prompt for the subject shall be written. At this point, *mailx* is in input mode.  
22560           This input mode is also entered when using one of the Receive Mode synopsis forms and a reply

22561 or new message is composed using the **reply**, **Reply**, **followup**, **Followup**, or **mail** commands  
 22562 and standard input is a terminal. When the message is typed and the end of message is  
 22563 encountered, the message shall be passed to the mail delivery software. Commands can be  
 22564 entered by beginning a line with the escape character (by default, tilde ('~')) followed by a  
 22565 single command letter and optional arguments. See **Commands in mailx** (on page 2804) for a  
 22566 summary of these commands. It is unspecified what effect these commands will have if  
 22567 standard input is not a terminal when a message is entered using either the Send Mode synopsis,  
 22568 or the Read Mode commands **reply**, **Reply**, **followup**, **Followup**, or **mail**.

22569 **Note:** For notational convenience, this section uses the default escape character, tilde, in all  
 22570 references and examples.

22571 At any time, the behavior of *mailx* shall be governed by a set of environmental and internal  
 22572 variables. These are flags and valued parameters that can be set and cleared via the *mailx set*  
 22573 and *unset* commands.

22574 Regular commands are of the form:

22575 [*command*] [*msglist*] [*argument ...*]

22576 If no *command* is specified in command mode, **next** shall be assumed. In input mode, commands  
 22577 shall be recognized by the escape character, and lines not treated as commands shall be taken as  
 22578 input for the message.

22579 In command mode, each message shall be assigned a sequential number, starting with 1.

22580 All messages have a state that affects how they are displayed in the header summary and how  
 22581 they are retained or deleted upon termination of *mailx*. There is at any time the notion of a  
 22582 *current* message, marked by a '>' at the beginning of a line in the header summary. When *mailx*  
 22583 is invoked using one of the Receive Mode synopsis forms, the current message shall be the first  
 22584 new message, if there is a new message, or the first unread message if there is an unread  
 22585 message, or the first message if there are any messages, or unspecified if there are no messages  
 22586 in the mailbox. Each command that takes an optional list of messages (*msglist*) or an optional  
 22587 single message (*message*) on which to operate shall leave the current message set to the highest-  
 22588 numbered message of the messages specified, unless the command deletes messages, in which  
 22589 case the current message shall be set to the first undeleted message (that is, a message not in the  
 22590 deleted state) after the highest-numbered message deleted by the command, if one exists, or the  
 22591 first undeleted message before the highest-numbered message deleted by the command, if one  
 22592 exists, or to an unspecified value if there are no remaining undeleted messages. All messages are  
 22593 in one of the following states:

22594 *new* The message is present in the system mailbox and has not been viewed by the user  
 22595 or moved to any other state. Messages in state *new* when *mailx* quits shall be  
 22596 retained in the system mailbox.

22597 *unread* The message has been present in the system mailbox for more than one invocation  
 22598 of *mailx* and has not been viewed by the user or moved to any other state.  
 22599 Messages in state *unread* when *mailx* quits shall be retained in the system mailbox.

22600 *read* The message has been processed by one of the following commands: **~f**, **~m**, **~F**, **~M**,  
 22601 **copy**, **mbox**, **next**, **pipe**, **print**, **Print**, **top**, **type**, **Type**, **undelete**. The **delete**, **dp**, and  
 22602 **dt** commands may also cause the next message to be marked as *read*, depending on  
 22603 the value of the **autoprint** variable. Messages that are in the system mailbox and in  
 22604 state *read* when *mailx* quits shall be saved in the **mbox**, unless the internal variable  
 22605 **hold** was set. Messages that are in the **mbox** or in a secondary mailbox and in state  
 22606 *read* when *mailx* quits shall be retained in their current location.

- 22607        *deleted*        The message has been processed by one of the following commands: **delete**, **dp**,  
 22608                    **dt**. Messages in state *deleted* when *mailx* quits shall be deleted. Deleted messages  
 22609                    shall be ignored until *mailx* quits or changes mailboxes or they are specified to the  
 22610                    undelete command; for example, the message specification */string* shall only  
 22611                    search the subject lines of messages that have not yet been deleted, unless the  
 22612                    command operating on the list of messages is **undelete**. No deleted message or  
 22613                    deleted message header shall be displayed by any *mailx* command other than  
 22614                    **undelete**.
- 22615        *preserved*        The message has been processed by a **preserve** command. When *mailx* quits, the  
 22616                    message shall be retained in its current location.
- 22617        *saved*            The message has been processed by one of the following commands: **save** or  
 22618                    **write**. If the current mailbox is the system mailbox, and the internal variable  
 22619                    **keepsave** is set, messages in the state *saved* shall be saved to the file designated by  
 22620                    the *MBOX* variable (see the ENVIRONMENT VARIABLES section). If the current  
 22621                    mailbox is the system mailbox, messages in the state *saved* shall be deleted from  
 22622                    the current mailbox, when the **quit** or **file** command is used to exit the current  
 22623                    mailbox.
- 22624                    The header-summary line for each message shall indicate the state of the message.
- 22625                    Many commands take an optional list of messages (*msglist*) on which to operate, which defaults  
 22626                    to the current message. A *msglist* is a list of message specifications separated by <blank>  
 22627                    characters, which can include:
- 22628        *n*            Message number *n*.
- 22629        **+**            The next undeleted message, or the next deleted message for the **undelete** command.
- 22630        **-**            The next previous undeleted message, or the next previous deleted message for the  
 22631                    **undelete** command.
- 22632        **.**            The current message.
- 22633        **^**            The first undeleted message, or the first deleted message for the **undelete** command.
- 22634        **\$**            The last message.
- 22635        **\***            All messages.
- 22636        *n-m*          An inclusive range of message numbers.
- 22637        *address*        All messages from *address*; any address as shown in a header summary shall be  
 22638                    matchable in this form.
- 22639        */string*        All messages with *string* in the subject line (case ignored).
- 22640        **:c**            All messages of type *c*, where *c* shall be one of:
- 22641                    **d**    Deleted messages.
- 22642                    **n**    New messages.
- 22643                    **o**    Old messages (any not in state *read* or *new*).
- 22644                    **r**    Read messages.
- 22645                    **u**    Unread messages.
- 22646                    Other commands take an optional message (*message*) on which to operate, which defaults to the  
 22647                    current message. All of the forms allowed for *msglist* are also allowed for *message*, but if more  
 22648                    than one message is specified, only the first shall be operated on.

22649 Other arguments are usually arbitrary strings whose usage depends on the command involved.

### 22650 **Start-Up in mailx**

22651 At start-up time, *mailx* shall take the following steps in sequence:

- 22652 1. Establish all variables at their stated default values.
- 22653 2. Process command line options, overriding corresponding default values.
- 22654 3. Import any of the *DEAD*, *EDITOR*, *MBOX*, *LISTER*, *PAGER*, *SHELL*, or *VISUAL* variables  
22655 that are present in the environment, overriding the corresponding default values.
- 22656 4. Read *mailx* commands from an unspecified system start-up file, unless the `-n` option is  
22657 given, to initialize any internal *mailx* variables and aliases.
- 22658 5. Process the start-up file of *mailx* commands named in the user *MAILRC* variable.

22659 Most regular *mailx* commands are valid inside start-up files, the most common use being to set  
22660 up initial display options and alias lists. The following commands shall be invalid in the start-up  
22661 file: **!**, **edit**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, **visual**, **Copy**, **followup**, and **Followup**.  
22662 Any errors in the start-up file shall either cause *mailx* to terminate with a diagnostic message and  
22663 a non-zero status or to continue after writing a diagnostic message, ignoring the remainder of  
22664 the lines in the start-up file.

22665 A blank line in a start-up file shall be ignored.

### 22666 **Internal Variables in mailx**

22667 The following variables are internal *mailx* variables. Each internal variable can be set via the  
22668 *mailx set* command at any time. The **unset** and **set no name** commands can be used to erase  
22669 variables.

22670 In the following list, variables shown as:

22671 `variable`

22672 represent Boolean values. Variables shown as:

22673 `variable=value`

22674 shall be assigned string or numeric values. For string values, the rules in **Commands in mailx**  
22675 (on page 2804) concerning file names and quoting also apply.

22676 The defaults specified here may be changed by the implementation-defined system start-up file  
22677 unless the user specifies the `-n` option.

22678 **allnet** All network names whose login name components match are treated as identical.  
22679 This shall cause the *msglist* message specifications to behave similarly. The default  
22680 shall be **noallnet**. See also the **alternates** command and the **metoo** variable.

22681 **append** Append messages to the end of the **mbox** file upon termination instead of placing  
22682 them at the beginning. The default shall be **noappend**. This variable shall not  
22683 affect the **save** command when saving to the **mbox**.

### 22684 **ask, asksub**

22685 Prompt for a subject line on outgoing mail if one is not specified on the command  
22686 line with the `-s` option. The **ask** and **asksub** forms are synonyms; the system shall  
22687 refer to **asksub** and **noasksub** in its messages, but shall accept **ask** and **noask** as  
22688 user input to mean **asksub** and **noasksub**. It shall not be possible to set both **ask**  
22689 and **noasksub**, or **noask** and **asksub**. The default shall be **asksub**, but no

|           |                            |                                                                                                                                                                                                                                                                                                                                                                |
|-----------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22690     |                            | prompting shall be done if standard input is not a terminal.                                                                                                                                                                                                                                                                                                   |
| 22691     | <b>askbcc</b>              | Prompt for the blind copy list. The default shall be <b>noaskbcc</b> .                                                                                                                                                                                                                                                                                         |
| 22692     | <b>askcc</b>               | Prompt for the copy list. The default shall be <b>noaskcc</b> .                                                                                                                                                                                                                                                                                                |
| 22693     | <b>autoprint</b>           | Enable automatic writing of messages after <b>delete</b> and <b>undelete</b> commands. The default shall be <b>noautoprint</b> .                                                                                                                                                                                                                               |
| 22694     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22695     | <b>bang</b>                | Enable the special-case treatment of exclamation marks ('!') in escape command lines; see the <b>escape</b> command and <b>Command Escapes in mailx</b> (on page 2812). The default shall be <b>nobang</b> , disabling the expansion of '!' in the <i>command</i> argument to the '~!' command and the '~! <i>command</i> escape.                              |
| 22696     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22697     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22698     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22699     | <b>cmd=command</b>         |                                                                                                                                                                                                                                                                                                                                                                |
| 22700     |                            | Set the default command to be invoked by the <b>pipe</b> command. The default shall be <b>nocmd</b> .                                                                                                                                                                                                                                                          |
| 22701     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22702     | <b>crt=number</b>          | Pipe messages having more than <i>number</i> lines through the command specified by the value of the <i>PAGER</i> variable. The default shall be <b>nocrt</b> . If it is set to null, the value used is implementation-defined.                                                                                                                                |
| 22703     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22704     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22705 XSI | <b>debug</b>               | Enable verbose diagnostics for debugging. Messages are not delivered. The default shall be <b>nodebug</b> .                                                                                                                                                                                                                                                    |
| 22706     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22707     | <b>dot</b>                 | When <b>dot</b> is set, a period on a line by itself during message input from a terminal shall also signify end-of-file (in addition to normal end-of-file). The default shall be <b>nodot</b> . If <b>ignoreeof</b> is set (see below), a setting of <b>nodot</b> shall be ignored and the period is the only method to terminate input mode.                |
| 22708     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22709     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22710     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22711     | <b>escape=c</b>            | Set the command escape character to be the character 'c'. By default, the command escape character shall be tilde. If <b>escape</b> is unset, tilde shall be used; if it is set to null, command escaping shall be disabled.                                                                                                                                   |
| 22712     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22713     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22714     | <b>flipr</b>               | Reverse the meanings of the <b>R</b> and <b>r</b> commands. The default shall be <b>noflipr</b> .                                                                                                                                                                                                                                                              |
| 22715     | <b>folder=directory</b>    |                                                                                                                                                                                                                                                                                                                                                                |
| 22716     |                            | The default directory for saving mail files. User-specified file names beginning with a plus sign ('+') shall be expanded by preceding the file name with this directory name to obtain the real path name. If <i>directory</i> does not start with a slash ('/'), the contents of <i>HOME</i> shall be prefixed to it. The default shall be <b>nofolder</b> . |
| 22717     |                            | If <b>folder</b> is unset or set to null, user-specified file names beginning with '+' shall refer to files in the current directory that begin with the literal '+' character. See also <b>outfolder</b> below. The <b>folder</b> value need not affect the processing of the files named in <i>MBOX</i> and <i>DEAD</i> .                                    |
| 22718     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22719     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22720     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22721     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22722     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22723     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22724     | <b>header</b>              | Enable writing of the header summary when entering <i>mailx</i> in Receive Mode. The default shall be <b>header</b> .                                                                                                                                                                                                                                          |
| 22725     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22726     | <b>hold</b>                | Preserve all messages that are read in the system mailbox instead of putting them in the <b>mbox</b> save file. The default shall be <b>nohold</b> .                                                                                                                                                                                                           |
| 22727     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22728     | <b>ignore</b>              | Ignore interrupts while entering messages. The default shall be <b>noignore</b> .                                                                                                                                                                                                                                                                              |
| 22729     | <b>ignoreeof</b>           | Ignore normal end-of-file during message input. Input can be terminated only by entering a period ('.') on a line by itself or by the '~.' command escape. The default shall be <b>noignoreeof</b> . See also <b>dot</b> above.                                                                                                                                |
| 22730     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22731     |                            |                                                                                                                                                                                                                                                                                                                                                                |
| 22732     | <b>indentprefix=string</b> |                                                                                                                                                                                                                                                                                                                                                                |
| 22733     |                            | A string that shall be added as a prefix to each line that is inserted into the message                                                                                                                                                                                                                                                                        |



|           |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22734     |                      | by the <code>~m</code> command escape. This variable shall default to one <code>&lt;tab&gt;</code> character.                                                                                                                                                                                                                                                                                                                |
| 22735     | <b>keep</b>          | When a system mailbox, secondary mailbox, or <b>mbox</b> is empty, truncate it to zero length instead of removing it. The default shall be <b>nokeep</b> .                                                                                                                                                                                                                                                                   |
| 22736     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22737     | <b>keepsave</b>      | Keep the messages that have been saved from the system mailbox into other files in the file designated by the variable <i>MBOX</i> , instead of deleting them. The default shall be <b>nokeepsave</b> .                                                                                                                                                                                                                      |
| 22738     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22739     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22740     | <b>metoo</b>         | Suppress the deletion of the login name of the user from the recipient list when replying to a message or sending to a group. The default shall be <b>nometoo</b> .                                                                                                                                                                                                                                                          |
| 22741     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22742 XSI | <b>onehop</b>        | When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (that is, one hop away). The default shall be <b>noonehop</b> . |
| 22743     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22744     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22745     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22746     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22747     | <b>outfolder</b>     | Cause the files used to record outgoing messages to be located in the directory specified by the <b>folder</b> variable unless the path name is absolute. The default shall be <b>nooutfolder</b> . See the <b>record</b> variable.                                                                                                                                                                                          |
| 22748     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22749     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22750     | <b>page</b>          | Insert a <code>&lt;form-feed&gt;</code> after each message sent through the pipe created by the <b>pipe</b> command. The default shall be <b>nopage</b> .                                                                                                                                                                                                                                                                    |
| 22751     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22752     | <b>prompt=string</b> |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22753     |                      | Set the command-mode prompt to <i>string</i> . If <i>string</i> is null or if <b>noprompt</b> is set, no prompting shall occur. The default shall be to prompt with the string " ? ".                                                                                                                                                                                                                                        |
| 22754     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22755     | <b>quiet</b>         | Refrain from writing the opening message and version when entering <i>mailx</i> . The default shall be <b>noquiet</b> .                                                                                                                                                                                                                                                                                                      |
| 22756     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22757     | <b>record=file</b>   | Record all outgoing mail in the file with the path name <i>file</i> . The default shall be <b>no record</b> . See also <b>outfolder</b> above.                                                                                                                                                                                                                                                                               |
| 22758     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22759     | <b>save</b>          | Enable saving of messages in the dead-letter file on interrupt or delivery error. See the variable <i>DEAD</i> for the location of the dead-letter file. The default shall be <b>save</b> .                                                                                                                                                                                                                                  |
| 22760     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22761     | <b>screen=number</b> |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22762     |                      | Set the number of lines in a screenful of headers for the <b>headers</b> and <b>z</b> commands. If <b>screen</b> is not specified, a value based on the terminal type identified by the <i>TERM</i> environment variable, the window size, the baud rate, or some combination of these shall be used.                                                                                                                        |
| 22763     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22764     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22765     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22766     | <b>sendwait</b>      | Wait for the background mailer to finish before returning. The default shall be <b>nosendwait</b> .                                                                                                                                                                                                                                                                                                                          |
| 22767     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22768     | <b>showto</b>        | When the sender of the message was the user who is invoking <i>mailx</i> , write the information from the <b>To:</b> line instead of the <b>From:</b> line in the header summary. The default shall be <b>noshowto</b> .                                                                                                                                                                                                     |
| 22769     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22770     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22771     | <b>sign=string</b>   | Set the variable inserted into the text of a message when the <code>~a</code> command escape is given. The default shall be <b>nosign</b> . The character sequences <code>'\t'</code> and <code>'\n'</code> shall be recognized in the variable as <code>&lt;tab&gt;</code> and <code>&lt;newline&gt;</code> characters, respectively. (See also <code>~i</code> in <b>Command Escapes in mailx</b> (on page 2812).)         |
| 22772     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22773     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22774     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22775     | <b>Sign=string</b>   | Set the variable inserted into the text of a message when the <code>~A</code> command escape is given. The default shall be <b>noSign</b> . The character sequences <code>'\t'</code> and <code>'\n'</code> shall be recognized in the variable as <code>&lt;tab&gt;</code> and <code>&lt;newline&gt;</code> characters, respectively.                                                                                       |
| 22776     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 22777     |                      |                                                                                                                                                                                                                                                                                                                                                                                                                              |

22778           **toplines=number**  
 22779                         Set the number of lines of the message to write with the **top** command. The default  
 22780                         shall be 5.

## 22781           **Commands in mailx**

22782           The following *mailx* commands shall be provided. In the following list, header refers to lines  
 22783           from the message header, as shown in the OUTPUT FILES section. Header-line refers to lines  
 22784           within the header that begin with one or more non-white-space characters, immediately  
 22785           followed by a colon and white space and continuing until the next line beginning with a non-  
 22786           white-space character or an empty line. Header-field refers to the portion of a header line prior  
 22787           to the first colon in that line.

22788           For each of the commands listed below, the command can be entered as the abbreviation (those  
 22789           characters in the Synopsis command word preceding the '['), the full command (all characters  
 22790           shown for the command word, omitting the '[' and ']'), or any truncation of the full  
 22791           command down to the abbreviation. For example, the **exit** command (shown as **ex[it]** in the  
 22792           Synopsis) can be entered as **ex**, **exi**, or **exit**.

22793           The arguments to commands can be quoted, using the following methods:

- 22794           • An argument can be enclosed between paired double-quotes (" ") or single-quotes (' '); any  
 22795           white space, shell word expansion, or backslash characters within the quotes shall be treated  
 22796           literally as part of the argument. A double-quote shall be treated literally within single-  
 22797           quotes and *vice versa*. These special properties of the quote marks shall occur only when they  
 22798           are paired at the beginning and end of the argument.
- 22799           • A backslash outside of the enclosing quotes shall be discarded and the following character  
 22800           treated literally as part of the argument.
- 22801           • An unquoted backslash at the end of a command line shall be discarded and the next line  
 22802           shall continue the command.

22803           File names, where expected, shall be subjected to the process of shell word expansions (see  
 22804           Section 2.6 (on page 2244)); if more than a single path name results and the command is  
 22805           expecting one file, the effects are unspecified. If the file name begins with an unquoted plus sign,  
 22806           it shall not be expanded, but treated as the named file (less the leading plus) in the **folder**  
 22807           directory. (See the **folder** variable.)

## 22808           **Declare Aliases**

22809           *Synopsis:*     a[lias] [alias [address...]]  
 22810                         g[roup] [alias [address...]]

22811           Add the given addresses to the alias specified by *alias*. The names shall be substituted when  
 22812           *alias* is used as a recipient address specified by the user in an outgoing message (that is, other  
 22813           recipients addressed indirectly through the **reply** command shall not be substituted in this  
 22814           manner). Mail address alias substitution shall apply only when the alias string is used as a full  
 22815           address; for example, when **hlj** is an alias, *hlj@posix.com* does not trigger the alias substitution. If  
 22816           no arguments are given, write a listing of the current aliases to standard output. If only an *alias*  
 22817           argument is given, write a listing of the specified alias to standard output. These listings need  
 22818           not reflect the same order of addresses that were entered.

22819 **Declare Alternatives**22820 *Synopsis:* alt[ernates] name...

22821 (See also the **metoo** command.) Declare a list of alternative names for the user's login. When  
 22822 responding to a message, these names shall be removed from the list of recipients for the  
 22823 response. The comparison of names shall be in a case-insensitive manner. With no arguments,  
 22824 **alternates** shall write the current list of alternative names.

22825 **Change Current Directory**22826 *Synopsis:* cd [directory]

22827 ch[dir] [directory]

22828 Change directory. If *directory* is not specified, the contents of *HOME* shall be used.22829 **Copy Messages**22830 *Synopsis:* c[opy] [file]

22831 c[opy] [msglist] file

22832 C[opy] [msglist]

22833 Copy messages to the file named by the path name *file* without marking the messages as saved.  
 22834 Otherwise, it shall be equivalent to the **save** command.

22835 In the capitalized form, save the specified messages in a file whose name is derived from the  
 22836 author of the message to be saved, without marking the messages as saved. Otherwise, it shall  
 22837 be equivalent to the **Save** command.

22838 **Delete Messages**22839 *Synopsis:* d[ele]te [msglist]

22840 Mark messages for deletion from the mailbox. The deletions shall not occur until *mailx* quits (see  
 22841 the **quit** command) or changes mailboxes (see the **folder** command). If **autoprint** is set and there  
 22842 are messages remaining after the **delete** command, the current message shall be written as  
 22843 described for the **print** command (see the **print** command); otherwise, the *mailx* prompt shall be  
 22844 written.

22845 **Discard Header Fields**22846 *Synopsis:* di[scard] [header-field...]

22847 ig[nore] [header-field...]

22848 Suppress the specified header fields when writing messages. Specified *header-fields* shall be  
 22849 added to the list of suppressed header fields. Examples of header fields to ignore are **status** and  
 22850 **cc**. The fields shall be included when the message is saved. The **Print** and **Type** commands shall  
 22851 override this command. The comparison of header fields shall be in a case-insensitive manner. If  
 22852 no arguments are specified, write a list of the currently suppressed header fields to standard  
 22853 output; the listing need not reflect the same order of header fields that were entered.

22854 If both **retain** and **discard** commands are given, **discard** commands shall be ignored.

22855 **Delete Messages and Display**

22856 *Synopsis:* dp [*msglist*]  
22857 dt [*msglist*]

22858 Delete the specified messages as described for the **delete** command, except that the **autoprint**  
22859 variable shall have no effect, and the current message shall be written only if it was set to a  
22860 message after the last message deleted by the command. Otherwise, an informational message  
22861 to the effect that there are no further messages in the mailbox shall be written, followed by the  
22862 *mailx* prompt.

22863 **Echo a String**

22864 *Synopsis:* ec[*ho*] *string* ...

22865 Echo the given strings, equivalent to the shell *echo* utility.

22866 **Edit Messages**

22867 *Synopsis:* e[*dit*] [*msglist*]

22868 Edit the given messages. The messages shall be placed in a temporary file and the utility named  
22869 by the *EDITOR* variable is invoked to edit each file in sequence. The default *EDITOR* is  
22870 unspecified.

22871 The **edit** command does not modify the contents of those messages in the mailbox.

22872 **Exit**

22873 *Synopsis:* ex[*it*]  
22874 x[*it*]

22875 Exit from *mailx* without changing the mailbox. No messages shall be saved in the **mbox** (see also  
22876 **quit**).

22877 **Change Folder**

22878 *Synopsis:* fi[*le*] [*file*]  
22879 fold[*er*] [*file*]

22880 Quit (see the **quit** command) from the current file of messages and read in the file named by the  
22881 path name *file*. If no argument is given, the name and status of the current mailbox shall be  
22882 written.

22883 Several unquoted special characters shall be recognized when used as *file* names, with the  
22884 following substitutions:

22885 % The system mailbox for the invoking user.

22886 %*user* The system mailbox for *user*.

22887 # The previous file.

22888 & The current **mbox**.

22889 +*file* The named file in the **folder** directory. (See the **folder** variable.)

22890 The default file shall be the current mailbox.

22891 **Display List of Folders**22892 *Synopsis:* folders22893 Write the names of the files in the directory set by the **folder** variable. The command specified by  
22894 the *LISTER* environment variable shall be used (see the ENVIRONMENT VARIABLES section).22895 **Follow Up Specified Messages**22896 **Notes to Reviewers**22897 *This section with side shading will not appear in the final copy. - Ed.*

22898 D1, XCU, ERN 300 says that it appears the second sentence below applies to both forms.

22899 *Synopsis:* fo[llowup] [*message*]22900 F[ollowup] [*msglist*]22901 In the lowercase form, respond to a message, recording the response in a file whose name is  
22902 derived from the author of the message. Overrides the **record** variable, if set. See also the **save**  
22903 and **copy** commands and **outfolder**.22904 In the capitalized form, respond to the first message in the *msglist*, sending the message to the  
22905 author of each message in the *msglist*. The subject line shall be taken from the first message and  
22906 the response shall be recorded in a file whose name is derived from the author of the first  
22907 message. See also the **Save** and **Copy** commands and **outfolder**.22908 **Display Header Summary for Specified Messages**22909 *Synopsis:* f[rom] [*msglist*]

22910 Write the header summary for the specified messages.

22911 **Display Header Summary**22912 *Synopsis:* h[eaders] [*message*]22913 Write the page of headers that includes the message specified. If the *message* argument is not  
22914 specified, the current message shall not change. However, if the *message* argument is specified,  
22915 the current message shall become the message that appears at the top of the page of headers that  
22916 includes the message specified. The **screen** variable sets the number of headers per page. See  
22917 also the **z** command.22918 **Help**22919 *Synopsis:* hel[p]

22920 ?

22921 Write a summary of commands.

22922 **Hold Messages**22923 *Synopsis:* ho[ld] [*msglist*]22924 pre[serve] [*msglist*]22925 Mark the messages in *msglist* to be retained in the mailbox when *mailx* terminates. This shall  
22926 override any commands that might previously have marked the messages to be deleted. During  
22927 the current invocation of *mailx*, only the **delete**, **dp**, or **dt** commands shall remove the *preserve*  
22928 marking of a message.

22929       **Execute Commands Conditionally**

22930       *Synopsis:*    i[f] s|r  
 22931                    mail-commands  
 22932                    el[se]  
 22933                    mail-commands  
 22934                    en[dif]

22935       Execute commands conditionally, where **if s** executes the following *mail-commands*, up to an  
 22936       **else** or **endif**, if the program is in Send Mode, and **if r** shall cause the *mail-commands* to be  
 22937       executed only in Receive Mode.

22938       **List Available Commands**

22939       *Synopsis:*    l[ist]

22940       Write a list of all commands available. No explanation shall be given.

22941       **Mail a Message**

22942       *Synopsis:*    m[ail] address...

22943       Mail a message to the specified addresses or aliases.

22944       **Direct Messages to mbox**

22945       *Synopsis:*    mb[ox] [msglist]

22946       Arrange for the given messages to end up in the **mbox** save file when *mailx* terminates normally.  
 22947       See *MBOX*. See also the **exit** and **quit** commands.

22948       **Process Next Specified Message**

22949       *Synopsis:*    n[ext] [message]

22950       If the current message has not been written (for example, by the **print** command) since *mailx*  
 22951       started or since any other message was the current message, behave as if the **print** command  
 22952       was entered. Otherwise, if there is an undeleted message after the current message, make it the  
 22953       current message and behave as if the **print** command was entered. Otherwise, an informational  
 22954       message to the effect that there are no further messages in the mailbox shall be written, followed  
 22955       by the *mailx* prompt.

22956       **Pipe Message**

22957       *Synopsis:*    pi[pe] [[msglist] command]  
 22958                    | [[msglist] command]

22959       Pipe the messages through the given *command* by invoking the command interpreter specified  
 22960       by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) The application shall ensure  
 22961       that the command is given as a single argument. Quoting, described previously, can be used to  
 22962       accomplish this. If no arguments are given, the current message shall be piped through the  
 22963       command specified by the value of the **cmd** variable. If the **page** variable is set, a <form-feed>  
 22964       character shall be inserted after each message.

22965 **Display Message with Headers**

22966 *Synopsis:* P[rint] [msglist]  
 22967 T[ype] [msglist]

22968 Write the specified messages, including all header lines, to standard output. Override  
 22969 suppression of lines by the **discard**, **ignore**, and **retain** commands. If **crt** is set, the messages  
 22970 longer than the number of lines specified by the **crt** variable shall be paged through the  
 22971 command specified by the *PAGER* environment variable.

22972 **Display Message**

22973 *Synopsis:* p[rint] [msglist]  
 22974 t[ype] [msglist]

22975 Write the specified messages to standard output. If **crt** is set, the messages longer than the  
 22976 number of lines specified by the **crt** variable shall be paged through the command specified by  
 22977 the *PAGER* environment variable.

22978 **Quit**

22979 *Synopsis:* q[uit]  
 22980 end-of-file

22981 Terminate *mailx*, storing messages that were read in **mbox** (if the current mailbox is the system  
 22982 mailbox and unless **hold** is set), deleting messages that have been explicitly saved (unless  
 22983 **keepsave** is set), discarding messages that have been deleted, and saving all remaining messages  
 22984 in the mailbox.

22985 **Reply to a Message List**

22986 *Synopsis:* R[eply] [msglist]  
 22987 R[espond] [msglist]

22988 Mail a reply message to the sender of each message in the *msglist*. The subject line shall be  
 22989 formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject  
 22990 from the first message. If **record** is set to a file name, the response shall be saved at the end of  
 22991 that file.

22992 See also the **flipr** variable.

22993 **Reply to a Message**

22994 *Synopsis:* r[eply] [message]  
 22995 r[espond] [message]

22996 Mail a reply message to all recipients included in the header of the message. The subject line  
 22997 shall be formed by concatenating **Re:**<space> (unless it already begins with that string) and the  
 22998 subject from the message. If **record** is set to a file name, the response shall be saved at the end of  
 22999 that file.

23000 See also the **flipr** variable.

23001       **Retain Header Fields**

23002       *Synopsis:*     ret[ain] [*header-field...*]

23003       Retain the specified header fields when writing messages. This command shall override all  
23004       **discard** and **ignore** commands. The comparison of header fields shall be in a case-insensitive  
23005       manner. If no arguments are specified, write a list of the currently retained header fields to  
23006       standard output; the listing need not reflect the same order of header fields that were entered.

23007       **Save Messages**

23008       *Synopsis:*     s[ave] [*file*]  
23009                   s[ave] [*msglist*] *file*  
23010                   S[ave] [*msglist*]

23011       Save the specified messages in the file named by the path name *file*, or the **mbox** if the *file*  
23012       argument is omitted. The file shall be created if it does not exist; otherwise, the messages shall be  
23013       appended to the file. The message shall be put in the state *saved*, and shall behave as specified in  
23014       the description of the *saved* state when the current mailbox is exited by the **quit** or **file**  
23015       command.

23016       In the capitalized form, save the specified messages in a file whose name is derived from the  
23017       author of the first message. The name of the file shall be taken to be the author's name with all  
23018       network addressing stripped off. See also the **Copy**, **followup**, and **Followup** commands and  
23019       **outfolder** variable.

23020       **Set Variables**

23021       *Synopsis:*     se[t] [*name*[=*string*]] ...] [*name*=*number* ...] [*noname* ...]

23022       Define one or more variables called *name*. The variable can be given a null, string, or numeric  
23023       value. Quoting and backslash escapes can occur anywhere in *string*, as described previously, as  
23024       if the *string* portion of the argument were the entire argument. The forms *name* and *name=* shall  
23025       be equivalent to *name=""* for variables that take string values. The **set** command without  
23026       arguments shall write a list of all defined variables and their values. The **no name** form shall be  
23027       equivalent to **unset name**.

23028       **Invoke a Shell**

23029       *Synopsis:*     sh[ell]

23030       Invoke an interactive command interpreter (see also *SHELL*).

23031       **Display Message Size**

23032       *Synopsis:*     si[ze] [*msglist*]

23033       Write the size in bytes of each of the specified messages.

23034       **Read mailx Commands From a File**

23035       *Synopsis:*     so[urce] *file*

23036       Read and execute commands from the file named by the path name *file* and return to command  
23037       mode.



**23038 Display Beginning of Messages**

23039 *Synopsis:* to[p] [msglist]

23040 Write the top few lines of each of the specified messages. If the **toplines** variable is set, it is taken  
23041 as the number of lines to write. The default shall be 5.

**23042 Touch Messages**

23043 *Synopsis:* tou[ch] [msglist]

23044 Touch the specified messages. If any message in *msglist* is not specifically deleted nor saved in a  
23045 file, it shall be placed in the **mbox** upon normal termination. See **exit** and **quit**.

**23046 Delete Aliases**

23047 *Synopsis:* una[lias] [alias]...

23048 Delete the specified alias names. If a specified alias does not exist, the results are unspecified.

**23049 Undelete Messages**

23050 *Synopsis:* u[ndelete] [msglist]

23051 Change the state of the specified messages from deleted to read. If **autoprint** is set, the last  
23052 message of those restored shall be written. If *msglist* is not specified, the message shall be  
23053 selected as follows:

- 23054 • If there are any deleted messages that follow the current message, the first of these shall be  
23055 chosen.
- 23056 • Otherwise, the last deleted message that also precedes the current message shall be chosen.

**23057 Unset Variables**

23058 *Synopsis:* uns[et] name...

23059 Cause the specified variables to be erased.

**23060 Edit Message with Full-Screen Editor**

23061 *Synopsis:* v[isual] [msglist]

23062 Edit the given messages with a screen editor. Each message shall be placed in a temporary file,  
23063 and the utility named by the *VISUAL* variable shall be invoked to edit each file in sequence. The  
23064 default editor shall be *vi*.

23065 The **visual** command does not modify the contents of those messages in the mailbox.

**23066 Write Messages to a File**

23067 *Synopsis:* w[rite] [msglist] file

23068 Write the given messages to the file specified by the path name *file*, minus the message header.  
23069 Otherwise, it shall be equivalent to the **save** command.

23070 **Scroll Header Display**23071 *Synopsis:* z[+|-]

23072 Scroll the header display forward (if '+' is specified or if no option is specified) or backward (if  
 23073 '-' is specified) one screenful. The number of headers written shall be set by the **screen**  
 23074 variable.

23075 **Invoke Shell Command**23076 *Synopsis:* !*command*

23077 Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*.  
 23078 (See also *sh -c*.) If the **bang** variable is set, each unescaped occurrence of '!' in *command* shall  
 23079 be replaced with the command executed by the previous ! *command* or '! *command* escape.

23080 **Null Command**23081 *Synopsis:* # *comment*23082 This null command (comment) shall be ignored by *mailx*.23083 **Display Current Message Number**23084 *Synopsis:* =

23085 Write the current message number.

23086 **Command Escapes in mailx**

23087 The following commands can be entered only from input mode, by beginning a line with the  
 23088 escape character (by default, tilde ('~')). See the **escape** variable description for changing this  
 23089 special character. The format for the commands shall be:

23090 <ESC><*command-char*><*separator*>[<*arguments*>]23091 where the <*separator*> can be zero or more <blank> characters.

23092 In the following descriptions, the application shall ensure that the argument *command* (but not  
 23093 *mailx-command*) is a shell command string. Any string acceptable to the command interpreter  
 23094 specified by the *SHELL* variable when it is invoked as *SHELL -c command\_string* shall be valid.  
 23095 The command can be presented as multiple arguments (that is, quoting is not required).

23096 Command escapes that are listed with *msglist* or *mailx-command* arguments are invalid in Send  
 23097 Mode and produce unspecified results.

23098 **~! *command*** Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and  
 23099 *command*; and then return to input mode. If the **bang** variable is set, each  
 23100 unescaped occurrence of '!' in *command* shall be replaced with the command  
 23101 executed by the previous ! *command* or '! *command* escape.

23102 **~.** Simulate end-of-file (terminate message input).

23103 **~: *mailx-command*, ~\_ *mailx-command***  
 23104 Perform the command-level request.

23105 **~?** Write a summary of command escapes.23106 **~A** This shall be equivalent to **~i Sign**.23107 **~a** This shall be equivalent to **~i sign**.

- 23108       ~**b** *name...*    Add the *names* to the blind carbon copy (**Bcc**) list.
- 23109       ~**c** *name...*    Add the *names* to the carbon copy (**Cc**) list.
- 23110       ~**d**                Read in the dead-letter file. See *DEAD* for a description of this file.
- 23111       ~**e**                Invoke the editor, as specified by the *EDITOR* environment variable, on the partial message.
- 23112
- 23113       ~**f** [*msglist*]    Forward the specified messages. The specified messages shall be inserted into the current message without alteration. This command escape also shall insert message headers into the message with field selection affected by the **discard**, **ignore**, and **retain** commands.
- 23114
- 23115
- 23116
- 23117       ~**F** [*msglist*]    This shall be the equivalent of the ~**f** command escape, except that all headers shall be included in the message, regardless of previous **discard**, **ignore**, and **retain** commands.
- 23118
- 23119
- 23120       ~**h**                If standard input is a terminal, prompt for a **Subject** line and the **To**, **Cc**, and **Bcc** lists. Other implementation-defined headers may also be presented for editing. If the field is written with an initial value, it can be edited as if it had just been typed.
- 23121
- 23122
- 23123       ~**i** *string*        Insert the value of the named variable, followed by a <newline> character, into the text of the message. If the string is unset or null, the message shall not be changed.
- 23124
- 23125       ~**m** [*msglist*]    Insert the specified messages into the message, prefixing non-empty lines with the string in the **indentprefix** variable. This command escape also shall insert message headers into the message, with field selection affected by the **discard**, **ignore**, and **retain** commands.
- 23126
- 23127
- 23128
- 23129       ~**M** [*msglist*]    This shall be the equivalent of the ~**m** command escape, except that all headers shall be included in the message, regardless of previous **discard**, **ignore**, and **retain** commands.
- 23130
- 23131
- 23132       ~**p**                Write the message being entered. If the message is longer than **crt** lines (see **Internal Variables in mailx** (on page 2801)), the output shall be paginated as described for the *PAGER* variable.
- 23133
- 23134
- 23135       ~**q**                Quit (see the **quit** command) from input mode by simulating an interrupt. If the body of the message is not empty, the partial message shall be saved in the dead-letter file. See *DEAD* for a description of this file.
- 23136
- 23137
- 23138       "**r** *file*, ~< *file*, ~**r** !*command*, ~< !*command*"
- 23139                Read in the file specified by the path name *file*. If the argument begins with an exclamation mark ('!'), the rest of the string shall be taken as an arbitrary system command; the command interpreter specified by *SHELL* shall be invoked with two arguments: **-c** and *command*. The standard output of *command* shall be inserted into the message.
- 23140
- 23141
- 23142
- 23143
- 23144       ~**s** *string*        Set the subject line to *string*.
- 23145
- 23145       ~**t** *name...*    Add the given *names* to the **To** list.
- 23146
- 23147       ~**v**                Invoke the full-screen editor, as specified by the *VISUAL* environment variable, on the partial message.
- 23148
- 23149       ~**w** *file*            Write the partial message, without the header, onto the file named by the path name *file*. The file shall be created or the message shall be appended to it if the file exists.
- 23150

23151       ~x           Exit as with ~q, except the message shall not be saved in the dead-letter file.

23152       ~| *command* Pipe the body of the message through the given *command* by invoking the  
23153           command interpreter specified by *SHELL* with two arguments: -c and *command*.  
23154           If the *command* returns a successful exit status, the standard output of the  
23155           command shall replace the message. Otherwise, the message shall remain  
23156           unchanged. If the *command* fails, an error message giving the exit status shall be  
23157           written.

23158 **EXIT STATUS**

23159       When the -e option is specified, the following exit values are returned:

23160       0   Mail was found.

23161       >0  Mail was not found or an error occurred.

23162       Otherwise, the following exit values are returned:

23163       0   Successful completion; note that this status implies that all messages were *sent*, but it gives  
23164           no assurances that any of them were actually *delivered*.

23165       >0  An error occurred.

23166 **CONSEQUENCES OF ERRORS**

23167       When in input mode (Receive Mode) or Send Mode:

23168       • If an error is encountered processing a command escape (see **Command Escapes in mailx**  
23169           (on page 2812)), a diagnostic message shall be written to standard error, and the message  
23170           being composed may be modified, but this condition shall not prevent the message from  
23171           being sent.

23172       • Other errors shall prevent the sending of the message.

23173       When in command mode:

23174       • Default.

23175 **APPLICATION USAGE**

23176       Delivery of messages to remote systems requires the existence of communication paths to such  
23177       systems. These need not exist.

23178       Input lines are limited to {LINE\_MAX} bytes, but mailers between systems may impose more  
23179       severe line-length restrictions. This volume of IEEE Std. 1003.1-200x does not place any  
23180       restrictions on the length of messages handled by *mailx*, and for delivery of local messages the  
23181       only limitations should be the normal problems of available disk space for the target mail file.  
23182       When sending messages to external machines, applications are advised to limit messages to less  
23183       than 100 kilobytes because some mail gateways impose message-length restrictions.

23184       The format of the system mailbox is intentionally unspecified. Not all systems implement  
23185       system mailboxes as flat files, particularly with the advent of multimedia mail messages. Some  
23186       system mailboxes may be multiple files, others records in a database. The internal format of the  
23187       messages themselves are specified with the historical format from Version 7, but only after they  
23188       have been saved in some file other than the system mailbox. This was done so that many  
23189       historical applications expecting text-file mailboxes are not broken.

23190       Some new formats for messages can be expected in the future, probably including binary data,  
23191       bit maps, and various multimedia objects. As described here, *mailx* is not prohibited from  
23192       handling such messages, but it must store them as text files in secondary mailboxes (unless  
23193       some extension, such as a variable or command line option, is used to change the stored format).  
23194       Its method of doing so is implementation-defined and might include translating the data into

23195 text file-compatible or readable form or omitting certain portions of the message from the stored  
23196 output.

23197 The **discard** and **ignore** commands are not inverses of the **retain** command. The **retain**  
23198 command discards all header-fields except those explicitly retained. The **discard** command  
23199 keeps all header-fields except those explicitly discarded. If headers exist on the retained header  
23200 list, **discard** and **ignore** commands are ignored.

#### 23201 EXAMPLES

23202 None.

#### 23203 RATIONALE

23204 The standard developers felt strongly that a method for applications to send messages to  
23205 specific users was necessary. The obvious example is a batch utility, running non-interactively,  
23206 that wishes to communicate errors or results to a user. However, the actual format, delivery  
23207 mechanism, and method of reading the message are clearly beyond the scope of this volume of  
23208 IEEE Std. 1003.1-200x.

23209 The intent of this command is to provide a simple, portable interface for sending messages non-  
23210 interactively. It merely defines a “front-end” to the historical mail system. It is suggested that  
23211 implementations explicitly denote the sender and recipient in the body of the delivered message.  
23212 Further specification of formats for either the message envelope or the message itself were  
23213 deliberately not made, as the industry is in the midst of changing from the current standards to  
23214 a more internationalized standard and it is probably incorrect, at this time, to require either one.

23215 Implementations are encouraged to conform to the various delivery mechanisms described in  
23216 the CCITT X.400 standards or to the equivalent Internet standards, described in Internet Request  
23217 for Comment (RFC) documents RFC 819, RFC 822, RFC 920, RFC 921, and RFC 1123.

23218 Many historical systems modified each body line that started with **From** by prefixing the ‘F’  
23219 with ‘>’. It is unnecessary, but allowed, to do that when the string does not follow a blank line  
23220 because it cannot be confused with the next header.

23221 The *edit* and *visual* commands merely edit the specified messages in a temporary file. They do  
23222 not modify the contents of those messages in the mailbox; such a capability could be added as an  
23223 extension, such as by using different command names.

23224 The restriction on a subject line being {LINE\_MAX}-10 bytes is based on the historical format  
23225 that consumes 10 bytes for **Subject:** and the trailing <newline>. Many historical mailers that a  
23226 message may encounter on other systems are not able to handle lines that long, however.

23227 Like the utilities *logger* and *lp*, *mailx* admittedly is difficult to test. This was not deemed sufficient  
23228 justification to exclude this utility from this volume of IEEE Std. 1003.1-200x. It is also arguable  
23229 that it is, in fact, testable, but that the tests themselves are not portable.

23230 When *mailx* is being used by an application that wishes to receive the results as if none of the  
23231 User Portability Utilities option features were supported, the *DEAD* environment variable must  
23232 be set to **/dev/null**. Otherwise, it may be subject to the file creations described in *mailx*  
23233 ASYNCHRONOUS EVENTS. Similarly, if the *MAILRC* environment variable is not set to  
23234 **/dev/null**, historical versions of *mailx* and *Mail* read initialization commands from a file before  
23235 processing begins. Since the initialization that a user specifies could alter the contents of  
23236 messages an application is trying to send, such applications must set *MAILRC* to **/dev/null**.

23237 The description of *LC\_TIME* uses “may affect” because many historical implementations do not  
23238 or cannot manipulate the date and time strings in the incoming mail headers. Some headers  
23239 found in incoming mail do not have enough information to determine the timezone in which the  
23240 mail originated, and, therefore, *mailx* cannot convert the date and time strings into the internal  
23241 form that then is parsed by routines like *strftime()* that can take *LC\_TIME* settings into account.

23242 Changing all these times to a user-specified format is allowed, but not required.

23243 The paginator selected when *PAGER* is null or unset is partially unspecified to allow the System  
23244 V historical practice of using *pg* as the default. Bypassing the pagination function, such as by  
23245 declaring that *cat* is the paginator, would not meet with the intended meaning of this  
23246 description. However, any “portable user” would have to set *PAGER* explicitly to get his or her  
23247 preferred paginator on all systems. The paginator choice was made partially unspecified, unlike  
23248 the *VISUAL* editor choice (mandated to be *vi*) because most historical pagers follow a common  
23249 theme of user input, whereas editors differ dramatically.

23250 Options to specify addresses as **cc** (carbon copy) or **bcc** (blind carbon copy) were considered to  
23251 be format details and were omitted.

23252 A zero exit status implies that all messages were *sent*, but it gives no assurances that any of them  
23253 were actually *delivered*. The reliability of the delivery mechanism is unspecified and is an  
23254 appropriate marketing distinction between systems.

23255 In order to conform to the Utility Syntax Guidelines, a solution was required to the optional *file*  
23256 option-argument to *-f*. By making *file* an operand, the guidelines are satisfied and users remain  
23257 portable. However, it does force implementations to support usage such as:

23258 `mailx -fin mymail.box`

23259 The **no name** method of unsetting variables is not present in all historical systems, but it is in  
23260 System V and provides a logical set of commands corresponding to the format of the display of  
23261 options from the *mailx set* command without arguments.

23262 The **ask** and **asksub** variables are the names selected by BSD and System V, respectively, for the  
23263 same feature. They are synonyms in this volume of IEEE Std. 1003.1-200x.

23264 The *mailx echo* command was not documented in the BSD version and has been omitted here  
23265 because it is not obviously useful for interactive users.

23266 The default prompt on the System V *mailx* is a question mark, on BSD *Mail* an ampersand. Since  
23267 this volume of IEEE Std. 1003.1-200x chose the *mailx* name, it kept the System V default,  
23268 assuming that BSD users would not have difficulty with this minor incompatibility (that they  
23269 can override).

23270 The meanings of **r** and **R** are reversed between System V *mailx* and SunOS *Mail*. Once again,  
23271 since this volume of IEEE Std. 1003.1-200x chose the *mailx* name, it kept the System V default,  
23272 but allows the SunOS user to achieve the desired results using **flpr**, an internal variable in  
23273 System V *mailx*, although it has not been documented in the SVID

23274 The **indentprefix** variable, the **retain** and **unalias** commands, and the **~F** and **~M** command  
23275 escapes were adopted from 4.3 BSD *Mail*.

23276 The **version** command was not included because no sufficiently general specification of the  
23277 version information could be devised that would still be useful to a portable user. This  
23278 command name should be used by suppliers who wish to provide version information about the  
23279 *mailx* command.

23280 The “implementation-specific (unspecified) system start-up file” historically has been named  
23281 **/etc/mailx.rc**, but this specific name and location are not required.

23282 The intent of the wording for the **next** command is that if any command has already displayed  
23283 the current message it should display a following message, but, otherwise, it should display the  
23284 current message. Consider the command sequence:

23285 `next 3`  
23286 `delete 3`

- 23287 next
- 23288 where the **autoprint** option was not set. The normative text specifies that the second **next**  
23289 command should display a message following the third message, because even though the  
23290 current message has not been displayed since it was set by the **delete** command, it has been  
23291 displayed since the current message was anything other than message number 3. This does not  
23292 always match historical practice in some implementations, where the command file address  
23293 followed by **next** (or the default command) would skip the message for which the user had  
23294 searched.
- 23295 **FUTURE DIRECTIONS**
- 23296 None.
- 23297 **SEE ALSO**
- 23298 *ed, ls, more, vi*
- 23299 **CHANGE HISTORY**
- 23300 First released in Issue 2.
- 23301 **Issue 4**
- 23302 Aligned with the ISO/IEC 9945-2:1993 standard.
- 23303 This utility is now mandatory; it is optional in Issue 3.
- 23304 **Issue 5**
- 23305 The description of the EDITOR environment variable is changed to indicate that *ed* is the default  
23306 editor if this variable is not set. In previous issues, this default was not stated explicitly at this  
23307 point but was implied further down in the text.
- 23308 FUTURE DIRECTIONS section added.
- 23309 **Issue 6**
- 23310 The following new requirements on POSIX implementations derive from alignment with the  
23311 Single UNIX Specification:
- 23312 • The **-F** option is added.
  - 23313 • The **allnet**, **debug**, and **sendwait** internal variables are added.
  - 23314 • The **C**, **ec**, **fo**, **F**, and **S** *mailx* commands are added.
- 23315 In the DESCRIPTION and ENVIRONMENT VARIABLES sections, text stating “*HOME*  
23316 directory” is replaced by “directory referred to by the *HOME* environment variable”.
- 23317 The *mailx* utility is aligned with the IEEE P1003.2b draft standard, which included various  
23318 clarifications to resolve IEEE PASC Interpretations submitted for the ISO POSIX-2:1993  
23319 standard. In particular, the changes here address IEEE PASC Interpretations 1003.2 #10, #11,  
23320 #103, #106, #108, #114, #115, #122, and #129.
- 23321 The normative text is reworded to avoid use of the term “must” for application requirements.

## 23322 NAME

23323 make — maintain, update, and regenerate groups of programs (**DEVELOPMENT**)

## 23324 SYNOPSIS

```
23325 SD make [-einpqrst][-f makefile...] [-k | -S][macro=value]...
23326 [target_name...]
```

23327

## 23328 DESCRIPTION

23329 The *make* utility can be used as a part of software development to update files that are derived  
 23330 from other files. A typical case is one where object files are derived from the corresponding  
 23331 source files. The *make* utility examines time relationships and updates those derived files (called  
 23332 targets) that have modified times earlier than the modified times of the files (called  
 23333 prerequisites) from which they are derived. A description file (makefile) contains a description  
 23334 of the relationships between files, and the commands that need to be executed to update the  
 23335 targets to reflect changes in their prerequisites. Each specification, or rule, shall consist of a  
 23336 target, optional prerequisites, and optional commands to be executed when a prerequisite is  
 23337 newer than the target. There are two types of rule:

23338 1. *Inference rules*, which have one target name with at least one period ( '.' ) and no slash  
 23339 ( '/' )

23340 2. *Target rules*, which can have more than one target name

23341 In addition, *make* shall have a collection of built-in macros and inference rules that infer  
 23342 prerequisite relationships to simplify maintenance of programs.

23343 To receive exactly the behavior described in this section, the user shall ensure that a portable  
 23344 makefile:

- 23345 • Includes the special target **.POSIX**
- 23346 • Omits any special target reserved for implementations (a leading period followed by  
 23347 uppercase letters) that has not been specified by this section

23348 The behavior of *make* is unspecified if either or both of these conditions are not met.

## 23349 OPTIONS

23350 The *make* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 23351 12.2, Utility Syntax Guidelines.

23352 The following options shall be supported:

23353 **-e** Cause environment variables, including those with null values, to override macro  
 23354 assignments within makefiles.

23355 **-f *makefile*** Specify a different makefile. The argument *makefile* is a path name of a description  
 23356 file, which is also referred to as the *makefile*. A path name of '-' shall denote the  
 23357 standard input. There can be multiple instances of this option, and they shall be  
 23358 processed in the order specified. The effect of specifying the same option-  
 23359 argument more than once is unspecified.

23360 **-i** Ignore error codes returned by invoked commands. This mode is the same as if the  
 23361 special target **.IGNORE** were specified without prerequisites.

23362 **-k** Continue to update other targets that do not depend on the current target if a non-  
 23363 ignored error occurs while executing the commands to bring a target up-to-date.

23364 **-n** Write commands that would be executed on standard output, but do not execute  
 23365 them. However, lines with a plus sign ( '+' ) prefix shall be executed. In this mode,



- 23366 lines with an at sign ('@') character prefix shall be written to standard output.
- 23367 **-p** Write to standard output the complete set of macro definitions and target  
23368 descriptions. The output format is unspecified.
- 23369 **-q** Return a zero exit value if the target file is up-to-date; otherwise, return an exit  
23370 value of 1. Targets shall not be updated if this option is specified. However, a  
23371 makefile command line (associated with the targets) with a plus sign ('+') prefix  
23372 shall be executed.
- 23373 **-r** Clear the suffix list and does not use the built-in rules.
- 23374 **-S** Terminate *make* if an error occurs while executing the commands to bring a target  
23375 up-to-date. This shall be the default and the opposite of **-k**.
- 23376 **-s** Do not write makefile command lines or touch messages (see **-t**) to standard  
23377 output before executing. This mode shall be the same as if the special target  
23378 **.SILENT** were specified without prerequisites.
- 23379 **-t** Update the modification time of each target as though a *touch target* had been  
23380 executed. Targets that have prerequisites but no commands (see **Target Rules** (on  
23381 page 2822)), or that are already up-to-date, shall not be touched in this manner.  
23382 Write messages to standard output for each target file indicating the name of the  
23383 file and that it was touched. Normally, the makefile command lines associated  
23384 with each target are not executed. However, a command line with a plus sign  
23385 ('+') prefix shall be executed.
- 23386 Any options specified in the *MAKEFLAGS* environment variable shall be evaluated before any  
23387 options specified on the *make* utility command line. If the **-k** and **-S** options are both specified  
23388 on the *make* utility command line or by the *MAKEFLAGS* environment variable, the last option  
23389 specified shall take precedence. If the **-f** or **-p** options appear in the *MAKEFLAGS* environment  
23390 variable, the result is undefined.
- 23391 **OPERANDS**
- 23392 The following operands shall be supported:
- 23393 *target\_name* Target names, as defined in the EXTENDED DESCRIPTION section. If no target is  
23394 specified, while *make* is processing the makefiles, the first target that *make*  
23395 encounters that is not a special target or an inference rule shall be used.
- 23396 *macro=value* Macro definitions, as defined in **Macros** (on page 2824).
- 23397 If the *target\_name* and *macro=value* operands are intermixed on the *make* utility command line,  
23398 the results are unspecified.
- 23399 **STDIN**
- 23400 The standard input shall be used only if the *makefile* option-argument is '-'. See the INPUT  
23401 FILES section.
- 23402 **INPUT FILES**
- 23403 The input file, otherwise known as the makefile, is a text file containing rules, macro definitions,  
23404 and comments.
- 23405 **ENVIRONMENT VARIABLES**
- 23406 The following environment variables shall affect the execution of *make*:
- 23407 *LANG* Provide a default value for the internationalization variables that are unset or null.  
23408 If *LANG* is unset or null, the corresponding value from the implementation-  
23409 defined default locale shall be used. If any of the internationalization variables  
23410 contains an invalid setting, the utility shall behave as if none of the variables had

- 23411                    been defined.
- 23412            *LC\_ALL*    If set to a non-empty string value, override the values of all the other  
23413                    internationalization variables.
- 23414            *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
23415                    characters (for example, single-byte as opposed to multi-byte characters in  
23416                    arguments and input files).
- 23417            *LC\_MESSAGES*  
23418                    Determine the locale that should be used to affect the format and contents of  
23419                    diagnostic messages written to standard error.
- 23420            *MAKEFLAGS*  
23421                    This variable shall be interpreted as a character string representing a series of  
23422                    option characters to be used as the default options. The implementation shall  
23423                    accept both of the following formats (but need not accept them when intermixed):
- 23424                    • The characters are option letters without the leading hyphens or <blank>  
23425                    character separation used on a *make* utility command line.
  - 23426                    • The characters are formatted in a manner similar to a portion of the *make* utility  
23427                    command line: options are preceded by hyphens and <blank> character-  
23428                    separated as described in the Base Definitions volume of IEEE Std. 1003.1-200x,  
23429                    Section 12.2, Utility Syntax Guidelines. The *macro=value* macro definition  
23430                    operands can also be included. The difference between the contents of  
23431                    *MAKEFLAGS* and the *make* utility command line is that the contents of the  
23432                    variable shall not be subjected to the word expansions (see Section 2.6 (on page  
23433                    2244)) associated with parsing the command line values.
- 23434 XSI            *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 23435 XSI            *PROJECTDIR*  
23436                    Provide a directory to be used to search for SCCS files not found in the current  
23437                    directory. In all of the following cases, the search for SCCS files is made in the  
23438                    directory **SCCS** in the identified directory. If the value of *PROJECTDIR* begins  
23439                    with a slash, it shall be considered an absolute path name; otherwise, the value of  
23440                    *PROJECTDIR* is treated as a user name and that user's initial working directory  
23441                    shall be examined for a subdirectory **src** or **source**. If such a directory is found, it  
23442                    shall be used. Otherwise, the value is used as a relative path name.
- 23443                    If *PROJECTDIR* is not set or has a null value, the search for SCCS files shall be  
23444                    made in the directory **SCCS** in the current directory.
- 23445                    The setting of *PROJECTDIR* affects all files listed in the remainder of this utility  
23446                    description for files with a component named **SCCS**.
- 23447                    The value of the *SHELL* environment variable shall not be used as a macro and shall not be  
23448                    modified by defining the **SHELL** macro in a makefile or on the command line. All other  
23449                    environment variables, including those with null values, shall be used as macros, as defined in  
23450                    **Macros** (on page 2824).
- 23451 **ASYNCHRONOUS EVENTS**  
23452                    If not already ignored, *make* shall trap SIGHUP, SIGTERM, SIGINT, and SIGQUIT and remove  
23453                    the current target unless the target is a directory or the target is a prerequisite of the special  
23454                    target **.PRECIOUS** or unless one of the **-n**, **-p**, or **-q** options was specified. Any targets removed  
23455                    in this manner shall be reported in diagnostic messages of unspecified format, written to  
23456                    standard error. After this cleanup process, if any, *make* shall take the standard action for all other

23457 signals.

#### 23458 **STDOUT**

23459 The *make* utility shall write all commands to be executed to standard output unless the `-s` option  
 23460 was specified, the command is prefixed with an at sign, or the special target `.SILENT` has either  
 23461 the current target as a prerequisite or has no prerequisites. If *make* is invoked without any work  
 23462 needing to be done, it shall write a message to standard output indicating that no action was  
 23463 taken. If the `-t` option is present and a file is touched, *make* shall write to standard output a  
 23464 message of unspecified format indicating that the file was touched, including the file name of  
 23465 the file.

#### 23466 **STDERR**

23467 Used only for diagnostic messages.

#### 23468 **OUTPUT FILES**

23469 Files can be created when the `-t` option is present. Additional files can also be created by the  
 23470 utilities invoked by *make*.

#### 23471 **EXTENDED DESCRIPTION**

23472 The *make* utility attempts to perform the actions required to ensure that the specified targets are  
 23473 up-to-date. A target is considered out-of-date if it is older than any of its prerequisites or if it  
 23474 does not exist. The *make* utility shall treat all prerequisites as targets themselves and recursively  
 23475 ensure that they are up-to-date, processing them in the order in which they appear in the rule.  
 23476 The *make* utility shall use the modification times of files to determine whether the corresponding  
 23477 targets are out-of-date.

23478 After *make* has ensured that all of the prerequisites of a target are up-to-date and if the target is  
 23479 out-of-date, the commands associated with the target entry shall be executed. If there are no  
 23480 commands listed for the target, the target shall be treated as up-to-date.

#### 23481 **Makefile Syntax**

23482 A makefile can contain rules, macro definitions (see **Macros** (on page 2824)), and comments.  
 23483 There are two kinds of rules: *inference rules* and *target rules*. The *make* utility shall contain a set of  
 23484 built-in inference rules. If the `-r` option is present, the built-in rules shall not be used and the  
 23485 suffix list shall be cleared. Additional rules of both types can be specified in a makefile. If a rule  
 23486 is defined more than once, the value of the rule shall be that of the last one specified. Macros can  
 23487 also be defined more than once, and the value of the macro is specified in **Macros** (on page  
 23488 2824). Comments start with a number sign (`'#'`) and continue until an unescaped `<newline>`  
 23489 character is reached.

23490 By default, the following files shall be tried in sequence: `./makefile` and `./Makefile`. If neither  
 23491 `./makefile` or `./Makefile` are found, other implementation-defined files may also be tried. On  
 23492 XSI-conformant systems, the additional files `./s.makefile`, `SCCS/s.makefile`, `./s.Makefile`, and  
 23493 `SCCS/s.Makefile` shall also be tried.

23494 The `-f` option shall direct *make* to ignore any of these default files and use the specified argument  
 23495 as a makefile instead. If the `'-'` argument is specified, standard input shall be used.

23496 The term *makefile* is used to refer to any rules provided by the user, whether in `./makefile` or its  
 23497 variants, or specified by the `-f` option.

23498 The rules in makefiles shall consist of the following types of lines: target rules, including special  
 23499 targets (see **Target Rules** (on page 2822)), inference rules (see **Inference Rules** (on page 2825)),  
 23500 macro definitions (see **Macros** (on page 2824)), empty lines, and comments.

23501 When an escaped `<newline>` (one preceded by a backslash) is found anywhere in the makefile  
 23502 except in a command line, it shall be replaced, along with any leading white space on the

23503 following line, with a single <space>. When an escaped <newline> is found in a command line  
 23504 in a makefile, the command line shall contain the backslash, the <newline>, and the next line,  
 23505 except that the first character of the next line shall not be included if it is a <tab>.

### 23506 **Makefile Execution**

23507 Makefile command lines shall be processed one at a time by writing the makefile command line  
 23508 to the standard output (unless one of the conditions listed under '@' suppresses the writing)  
 23509 and executing the command(s) in the line. A <tab> character may precede the command to  
 23510 standard output. Command execution shall be as if the makefile command line were the  
 23511 argument to the *system()* function. The environment for the command being executed shall  
 23512 contain all of the variables in the environment of *make*.

23513 By default, when *make* receives a non-zero status from the execution of a command, it terminates  
 23514 with an error message to standard error.

23515 Makefile command lines can have one or more of the following prefixes: a hyphen ('-'), an at  
 23516 sign ('@'), or a plus sign ('+'). These modify the way in which *make* processes the command.  
 23517 When a command is written to standard output, the prefix shall not be included in the output.

23518 – If the command prefix contains a hyphen, or the **-i** option is present, or the special target  
 23519 **.IGNORE** has either the current target as a prerequisite or has no prerequisites, any error  
 23520 found while executing the command shall be ignored.

23521 @ If the command prefix contains an at sign and the *make* utility command line **-n** option is  
 23522 not specified, or the **-s** option is present, or the special target **.SILENT** has either the current  
 23523 target as a prerequisite or has no prerequisites, the command shall not be written to  
 23524 standard output before it is executed.

23525 + If the command prefix contains a plus sign, this indicates a makefile command line that  
 23526 shall be executed even if **-n**, **-q**, or **-t** is specified.

### 23527 **Target Rules**

23528 Target rules are formatted as follows:

```
23529 target [target...]: [prerequisite...][;command]
23530 [<tab>command
23531 <tab>command
23532 ...]
```

23533 *line that does not begin with <tab>*

23534 Target entries are specified by a <blank> character-separated, non-null list of targets, then a  
 23535 colon, then a <blank> character-separated, possibly empty list of prerequisites. Text following a  
 23536 semicolon, if any, and all following lines that begin with a <tab> character, are makefile  
 23537 command lines to be executed to update the target. The first non-empty line that does not begin  
 23538 with a <tab> character or '#' shall begin a new entry. An empty or blank line, or a line  
 23539 beginning with '#', may begin a new entry.

23540 Applications shall select target names from the set of characters consisting solely of periods,  
 23541 underscores, digits, and alphabetic characters from the portable character set (see the Base Definitions  
 23542 volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set). Implementations may  
 23543 allow other characters in target names as extensions. The interpretation of targets containing the  
 23544 characters '%' and '"' is implementation-defined.

23545 A target that has prerequisites, but does not have any commands, can be used to add to the  
 23546 prerequisite list for that target. Only one target rule for any given target can contain commands.

23547 Lines that begin with one of the following are called *special targets* and control the operation of  
23548 *make*:

23549 **.DEFAULT** If the makefile uses this special target, the application shall ensure that it is  
23550 specified with commands, but without prerequisites. The commands shall be used  
23551 by *make* if there are no other rules available to build a target.

23552 **.IGNORE** Prerequisites of this special target are targets themselves; this shall cause errors  
23553 from commands associated with them to be ignored in the same manner as  
23554 specified by the `-i` option. Subsequent occurrences of **.IGNORE** shall add to the  
23555 list of targets ignoring command errors. If no prerequisites are specified, *make* shall  
23556 behave as if the `-i` option had been specified and errors from all commands  
23557 associated with all targets shall be ignored.

23558 **.POSIX** The application shall ensure that this special target is specified without  
23559 prerequisites or commands. If it appears as the first non-comment line in the  
23560 makefile, *make* shall process the makefile as specified by this section; otherwise, the  
23561 behavior of *make* is unspecified.

23562 **.PRECIOUS** Prerequisites of this special target shall not be removed if *make* receives one of the  
23563 asynchronous events explicitly described in the ASYNCHRONOUS EVENTS  
23564 section. Subsequent occurrences of **.PRECIOUS** shall add to the list of precious  
23565 files. If no prerequisites are specified, all targets in the makefile shall be treated as  
23566 if specified with **.PRECIOUS**.

23567 XSI **.SCCS\_GET** The application shall ensure that this special target is specified without  
23568 prerequisites. If this special target is included in a makefile, the commands  
23569 specified with this target shall replace the default commands associated with this  
23570 special target (see **Default Rules** (on page 2828)). The commands specified with  
23571 this target are used to get all SCCS files that are not found in the current directory.

23572 When source files are named in a dependency list, *make* treats them just like any  
23573 other target. Because the source file is presumed to be present in the directory,  
23574 there is no need to add an entry for it to the makefile. When a target has no  
23575 dependencies, but is present in the directory, *make* assumes that that file is up-to-  
23576 date. If, however, an SCCS file named **SCCS/s.source\_file** is found for a target  
23577 *source\_file*, *make* does some additional checking to assure that the target is up-to-  
23578 date. If the target is missing, or if the SCCS file is newer, *make* automatically issues  
23579 the commands specified for the **.SCCS\_GET** special target to retrieve the most  
23580 recent version. However, if the target is writable by anyone, *make* does not retrieve  
23581 a new version.

23582 **.SILENT** Prerequisites of this special target are targets themselves; this shall cause  
23583 commands associated with them to not be written to the standard output before  
23584 they are executed. Subsequent occurrences of **.SILENT** shall add to the list of  
23585 targets with silent commands. If no prerequisites are specified, *make* shall behave  
23586 as if the `-s` option had been specified and no commands or touch messages  
23587 associated with any target shall be written to standard output.

23588 **.SUFFIXES** Prerequisites of **.SUFFIXES** shall be appended to the list of known suffixes and are  
23589 used in conjunction with the inference rules (see **Inference Rules** (on page 2825)).  
23590 If **.SUFFIXES** does not have any prerequisites, the list of known suffixes shall be  
23591 cleared.

23592 The special targets **.IGNORE**, **.POSIX**, **.PRECIOUS**, **.SILENT**, and **.SUFFIXES** shall be specified  
23593 without commands.

23594 Targets with names consisting of a leading period followed by the uppercase letters "POSIX"  
 23595 and then any other characters are reserved for future standardization. Targets with names  
 23596 consisting of a leading period followed by one or more uppercase letters are reserved for  
 23597 implementation extensions.

## 23598 **Macros**

23599 Macro definitions are in the form:

```
23600 string1 = [string2]
```

23601 The macro named *string1* is defined as having the value of *string2*, where *string2* is defined as all  
 23602 characters, if any, after the equal sign, up to a comment character ('#') or an unescaped  
 23603 <newline> character. Any <blank> characters immediately before or after the equal sign shall be  
 23604 ignored.

23605 Applications shall select macro names from the set of characters consisting solely of periods,  
 23606 underscores, digits, and alphabetic characters from the portable character set (see the Base Definitions  
 23607 volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set). A macro name shall not  
 23608 contain an equals sign. Implementations may allow other characters in macro names as  
 23609 extensions.

23610 Macros can appear anywhere in the makefile.  $\$(string1)$  or  $\${string1}$  shall be replaced by  
 23611 *string2*, as follows:

- 23612 • Macros in target lines shall be evaluated when the target line is read.
- 23613 • Macros in makefile command lines shall be evaluated when the command is executed.
- 23614 • Macros in the string before the equals sign in a macro definition shall be evaluated when the  
 23615 macro assignment is made.
- 23616 • Macros after the equals sign in a macro definition shall not be evaluated until the defined  
 23617 macro is used in a rule or command, or before the equals sign in a macro definition.

23618 The parentheses or braces are optional if *string1* is a single character. The macro  $\$\$$  shall be  
 23619 replaced by the single character '\$'.

23620 The forms  $\$(string1[:subst1=[subst2]])$  or  $\${string1[:subst1=[subst2]]}$  can be used to replace all  
 23621 occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* to be  
 23622 replaced shall be recognized when it is a suffix at the end of a word in *string1* (where a *word*, in  
 23623 this context, is defined to be a string delimited by the beginning of the line, a <blank> or  
 23624 <newline> character).

23625 Macro definitions shall be taken from the following sources, in the following logical order,  
 23626 before the makefile(s) are read.

- 23627 1. Macros specified on the *make* utility command line, in the order specified on the command  
 23628 line. It is unspecified whether the internal macros defined in **Internal Macros** (on page  
 23629 2826) are accepted from this source.
- 23630 2. Macros defined by the *MAKEFLAGS* environment variable, in the order specified in the  
 23631 environment variable. It is unspecified whether the internal macros defined in **Internal  
 23632 Macros** (on page 2826) are accepted from this source.
- 23633 3. The contents of the environment, excluding the *MAKEFLAGS* and *SHELL* variables and  
 23634 including the variables with null values.
- 23635 4. Macros defined in the inference rules built into *make*.

23636 Macro definitions from these sources shall not override macro definitions from a lower-  
 23637 numbered source. Macro definitions from a single source (for example, the *make* utility  
 23638 command line, the *MAKEFLAGS* environment variable, or the other environment variables) shall  
 23639 override previous macro definitions from the same source.

23640 Macros defined in the makefile(s) shall override macro definitions that occur before them in the  
 23641 makefile(s) and macro definitions from source 4. If the *-e* option is not specified, macros defined  
 23642 in the makefile(s) shall override macro definitions from source 3. Macros defined in the  
 23643 makefile(s) shall not override macro definitions from source 1 or source 2.

23644 Before the makefile(s) are read, all of the *make* utility command line options (except *-f* and *-p*)  
 23645 and *make* utility command line macro definitions (except any for the *MAKEFLAGS* macro), not  
 23646 already included in the *MAKEFLAGS* macro, shall be added to the *MAKEFLAGS* macro. Other  
 23647 implementation-defined options and macros may also be added to the *MAKEFLAGS* macro. If  
 23648 this modifies the value of the *MAKEFLAGS* macro, or, if the *MAKEFLAGS* macro is modified at  
 23649 any subsequent time, the *MAKEFLAGS* environment variable shall be modified to match the  
 23650 new value of the *MAKEFLAGS* macro.

23651 Before the makefile(s) are read, all of the *make* utility command line macro definitions (except the  
 23652 *MAKEFLAGS* macro or the *SHELL* macro) shall be added to the environment of *make*. Other  
 23653 implementation-defined variables may also be added to the environment of *make*.

23654 The *SHELL* macro shall be treated specially. It shall be provided by *make* and set to the path  
 23655 name of the shell command language interpreter (see *sh* (on page 3060)). The *SHELL*  
 23656 environment variable shall not affect the value of the *SHELL* macro. If *SHELL* is defined in the  
 23657 makefile or is specified on the command line, it shall replace the original value of the *SHELL*  
 23658 macro, but shall not affect the *SHELL* environment variable. Other effects of defining *SHELL* in  
 23659 the makefile or on the command line are implementation-defined.

## 23660 Inference Rules

23661 Inference rules are formatted as follows:

```
23662 target:
23663 <tab>command
23664 [<tab>command]
23665 ...
23666 line that does not begin with <tab> or #
```

23667 The application shall ensure that the *target* portion is a valid target name (see **Target Rules** (on  
 23668 page 2822)) of the form *.s2* or *.s1.s2* (where *.s1* and *.s2* are suffixes that have been given as  
 23669 prerequisites of the *.SUFFIXES* special target and *s1* and *s2* do not contain any slashes or  
 23670 periods.) If there is only one period in the target, it is a single-suffix inference rule. Targets with  
 23671 two periods are double-suffix inference rules. Inference rules can have only one target before the  
 23672 colon.

23673 The application shall ensure that the makefile does not specify prerequisites for inference rules;  
 23674 no characters other than white space shall follow the colon in the first line, except when creating  
 23675 the *empty rule*, described below. Prerequisites are inferred, as described below.

23676 Inference rules can be redefined. A target that matches an existing inference rule shall overwrite  
 23677 the old inference rule. An empty rule can be created with a command consisting of simply a  
 23678 semicolon (that is, the rule still exists and is found during inference rule search, but since it is  
 23679 empty, execution has no effect). The empty rule also can be formatted as follows:

```
23680 rule: ;
```

- 23681 where zero or more <blank> characters separate the colon and semicolon.
- 23682 The *make* utility uses the suffixes of targets and their prerequisites to infer how a target can be  
 23683 made up-to-date. A list of inference rules defines the commands to be executed. By default, *make*  
 23684 contains a built-in set of inference rules. Additional rules can be specified in the makefile.
- 23685 The special target **.SUFFIXES** contains as its prerequisites a list of suffixes that shall be used by  
 23686 the inference rules. The order in which the suffixes are specified defines the order in which the  
 23687 inference rules for the suffixes are used. New suffixes shall be appended to the current list by  
 23688 specifying a **.SUFFIXES** special target in the makefile. A **.SUFFIXES** target with no prerequisites  
 23689 shall clear the list of suffixes. An empty **.SUFFIXES** target followed by a new **.SUFFIXES** list is  
 23690 required to change the order of the suffixes.
- 23691 Normally, the user would provide an inference rule for each suffix. The inference rule to update  
 23692 a target with a suffix **.s1** from a prerequisite with a suffix **.s2** is specified as a target **.s2.s1**. The  
 23693 internal macros provide the means to specify general inference rules (see **Internal Macros**).
- 23694 When no target rule is found to update a target, the inference rules shall be checked. The suffix  
 23695 of the target (**.s1**) to be built is compared to the list of suffixes specified by the **.SUFFIXES** special  
 23696 targets. If the **.s1** suffix is found in **.SUFFIXES**, the inference rules shall be searched in the order  
 23697 defined for the first **.s2.s1** rule whose prerequisite file (**\$\*.s2**) exists. If the target is out-of-date  
 23698 with respect to this prerequisite, the commands for that inference rule shall be executed.
- 23699 If the target to be built does not contain a suffix and there is no rule for the target, the single  
 23700 suffix inference rules shall be checked. The single-suffix inference rules define how to build a  
 23701 target if a file is found with a name that matches the target name with one of the single suffixes  
 23702 appended. A rule with one suffix **.s2** is the definition of how to build *target* from **target.s2**. The  
 23703 other suffix (**.s1**) is treated as null.
- 23704 XSI A tilde ('~') in the above rules refers to an SCCS file in the current directory. Thus, the rule **.c~.o**  
 23705 would transform an SCCS C-language source file into an object file (**.o**). Because the **s.** of the  
 23706 SCCS files is a prefix, it is incompatible with *make*'s suffix point of view. Hence, the '**~**' is a way  
 23707 of changing any file reference into an SCCS file reference.
- 23708 **Libraries**
- 23709 If a target or prerequisite contains parentheses, it shall be treated as a member of an archive  
 23710 library. For the *lib(member.o)* expression *lib* refers to the name of the archive library and *member.o*  
 23711 to the member name. The application shall ensure that the member is an object file with the **.o**  
 23712 suffix. The modification time of the expression is the modification time for the member as kept  
 23713 in the archive library; see *ar* (on page 2348). The **.a** suffix refers to an archive library. The **.s2.a**  
 23714 rule is used to update a member in the library from a file with a suffix **.s2**.
- 23715 **Internal Macros**
- 23716 The *make* utility shall maintain five internal macros that can be used in target and inference rules.  
 23717 In order to clearly define the meaning of these macros, some clarification of the terms *target rule*,  
 23718 *inference rule*, *target*, and *prerequisite* is necessary.
- 23719 Target rules are specified by the user in a makefile for a particular target. Inference rules are  
 23720 user-specified or *make*-specified rules for a particular class of target name. Explicit prerequisites  
 23721 are those prerequisites specified in a makefile on target lines. Implicit prerequisites are those  
 23722 prerequisites that are generated when inference rules are used. Inference rules are applied to  
 23723 implicit prerequisites or to explicit prerequisites that do not have target rules defined for them in  
 23724 the makefile. Target rules are applied to targets specified in the makefile.



23725 Before any target in the makefile is updated, each of its prerequisites (both explicit and implicit)  
 23726 shall be updated. This shall be accomplished by recursively processing each prerequisite. Upon  
 23727 recursion, each prerequisite shall become a target itself. Its prerequisites in turn shall be  
 23728 processed recursively until a target is found that has no prerequisites, at which point the  
 23729 recursion stops. The recursion then shall back up, updating each target as it goes.

23730 In the definitions that follow, the word *target* refers to one of:

- 23731 • A target specified in the makefile
- 23732 • An explicit prerequisite specified in the makefile that becomes the target when *make*  
 23733 processes it during recursion
- 23734 • An implicit prerequisite that becomes a target when *make* processes it during recursion

23735 In the definitions that follow, the word *prerequisite* refers to one of the following:

- 23736 • An explicit prerequisite specified in the makefile for a particular target
- 23737 • An implicit prerequisite generated as a result of locating an appropriate inference rule and  
 23738 corresponding file that matches the suffix of the target

23739 The five internal macros are:

23740 **\$@** The **\$@** shall evaluate to the full target name of the current target, or the archive file  
 23741 name part of a library archive target. It shall be evaluated for both target and inference  
 23742 rules.

23743 For example, in the **.c.a** inference rule, **\$@** represents the out-of-date **.a** file to be built.  
 23744 Similarly, in a makefile target rule to build **lib.a** from **file.c**, **\$@** represents the out-of-  
 23745 date **lib.a**.

23746 **\$%** The **\$%** macro shall be evaluated only when the current target is an archive library  
 23747 member of the form *libname(member.o)*. In these cases, **\$@** shall evaluate to *libname* and  
 23748 **\$%** shall evaluates to *member.o*. The **\$%** macro shall be evaluated for both target and  
 23749 inference rules.

23750 For example, in a makefile target rule to build **lib.a(file.o)**, **\$%** represents **file.o**, as  
 23751 opposed to **\$@**, which represents **lib.a**.

23752 **\$?** The **\$?** macro shall evaluate to the list of prerequisites that are newer than the current  
 23753 target. It shall be evaluated for both target and inference rules.

23754 For example, in a makefile target rule to build *prog* from **file1.o**, **file2.o**, and **file3.o**, and  
 23755 where *prog* is not out of date with respect to **file1.o**, but is out of date with respect to  
 23756 **file2.o** and **file3.o**, **\$?** represents **file2.o** and **file3.o**.

23757 **\$<** In an inference rule, the **\$<** macro shall evaluate to the file name whose existence  
 23758 allowed the inference rule to be chosen for the target. In the **.DEFAULT** rule, the **\$<**  
 23759 macro shall evaluate to the current target name. The meaning of the **\$<** macro is  
 23760 otherwise unspecified.

23761 For example, in the **.c.a** inference rule, **\$<** represents the prerequisite **.c** file.

23762 **\$\*** The **\$\*** macro shall evaluate to the current target name with its suffix deleted. It shall be  
 23763 evaluated at least for inference rules.

23764 For example, in the **.c.a** inference rule, **\$\*.o** represents the out-of-date **.o** file that  
 23765 corresponds to the prerequisite **.c** file.

23766 Each of the internal macros has an alternative form. When an uppercase 'D' or 'F' is appended  
 23767 to any of the macros, the meaning is changed to the *directory part* for 'D' and *file name part* for

23768 'F'. The directory part is the path prefix of the file without a trailing slash; for the current  
 23769 directory, the directory part is '.'. When the \$? macro contains more than one prerequisite file  
 23770 name, the \$(?D) and \$(?F) (or \${?D} and \${?F}) macros expand to a list of directory name parts  
 23771 and file name parts respectively.

23772 For the target *lib(member.o)* and the *s2.a* rule, the internal macros are defined as:

23773 \$< *member.s2*

23774 \$\* *member*

23775 \$@ *lib*

23776 \$? *member.s2*

23777 \$% *member.o*

### 23778 **Default Rules**

23779 The default rules for *make* shall achieve results that are the same as if the following were used.  
 23780 Implementations that do not support the C-Language Development Utilities option may omit  
 23781 **CC**, **CFLAGS**, **YACC**, **YFLAGS**, **LEX**, **LFLAGS**, **LDFLAGS**, and the *.c*, *.y*, and *.l* inference rules.  
 23782 Implementations that do not support FORTRAN may omit **FC**, **FFLAGS**, and the *.f* inference  
 23783 rules. Implementations may provide additional macros and rules.

### 23784 *SPECIAL TARGETS*

23785 XSI `.SCCS_GET: sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) $@`

23786

23787 XSI `.SUFFIXES: .o .c .y .l .a .sh .f .c~ .y~ .l~ .sh~ .f~`

### 23788 **MACROS**

23789 MAKE=make

23790 AR=ar

23791 ARFLAGS=-rv

23792 YACC=yacc

23793 YFLAGS=

23794 LEX=lex

23795 LFLAGS=

23796 LDFLAGS=

23797 CC=c99

23798 CFLAGS=-O

23799 FC=fort77

23800 FFLAGS=-O 1

23801 XSI GET=get

23802 GFLAGS=

23803 SCCSFLAGS=

23804 SCCSGETFLAGS=-s

23805

### 23806 *SINGLE SUFFIX RULES*

23807 *.c*:

23808 \$(CC) \$(CFLAGS) \$(LDFLAGS) -o \$@ \$<

23809 *.f*:

23810 \$(FC) \$(FFLAGS) \$(LDFLAGS) -o \$@ \$<

```

23811 .sh:
23812 cp $< $@
23813 chmod a+x $@

23814 XSI .c~:
23815 $(GET) $(GFLAGS) -p $< > $*.c
23816 $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $*.c

23817 .f~:
23818 $(GET) $(GFLAGS) -p $< > $*.f
23819 $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $*.f

23820 .sh~:
23821 $(GET) $(GFLAGS) -p $< > $*.sh
23822 cp $*.sh $@
23823 chmod a+x $@
23824

23825 DOUBLE SUFFIX RULES

23826 .c.o:
23827 $(CC) $(CFLAGS) -c $<

23828 .f.o:
23829 $(FC) $(FFLAGS) -c $<

23830 .y.o:
23831 $(YACC) $(YFLAGS) $<
23832 $(CC) $(CFLAGS) -c y.tab.c
23833 rm -f y.tab.c
23834 mv y.tab.o $@

23835 .l.o:
23836 $(LEX) $(LFLAGS) $<
23837 $(CC) $(CFLAGS) -c lex.yy.c
23838 rm -f lex.yy.c
23839 mv lex.yy.o $@

23840 .y.c:
23841 $(YACC) $(YFLAGS) $<
23842 mv y.tab.c $@

23843 .l.c:
23844 $(LEX) $(LFLAGS) $<
23845 mv lex.yy.c $@

23846 XSI .c~.o:
23847 $(GET) $(GFLAGS) -p $< > $*.c
23848 $(CC) $(CFLAGS) -c $*.c

23849 .f~.o:
23850 $(GET) $(GFLAGS) -p $< > $*.f
23851 $(FC) $(FFLAGS) -c $*.f

23852 .y~.o:
23853 $(GET) $(GFLAGS) -p $< > $*.y
23854 $(YACC) $(YFLAGS) $*.y
23855 $(CC) $(CFLAGS) -c y.tab.c
23856 rm -f y.tab.c

```

```

23857 mv y.tab.o $@
23858 .l~.o:
23859 $(GET) $(GFLAGS) -p $< > $*.l
23860 $(LEX) $(LFLAGS) $*.l
23861 $(CC) $(CFLAGS) -c lex.yy.c
23862 rm -f lex.yy.c
23863 mv lex.yy.o $@
23864 .y~.c:
23865 $(GET) $(GFLAGS) -p $< > $*.y
23866 $(YACC) $(YFLAGS) $*.y
23867 mv y.tab.c $@
23868 .l~.c:
23869 $(GET) $(GFLAGS) -p $< > $*.l
23870 $(LEX) $(LFLAGS) $*.l
23871 mv lex.yy.c $@
23872
23873 .c.a:
23874 $(CC) -c $(CFLAGS) $<
23875 $(AR) $(ARFLAGS) $@ $*.o
23876 rm -f $*.o
23877 .f.a:
23878 $(FC) -c $(FFLAGS) $<
23879 $(AR) $(ARFLAGS) $@ $*.o
23880 rm -f $*.o

```

### 23881 EXIT STATUS

23882 When the `-q` option is specified, the *make* utility shall exit with one of the following values:

- 23883 0 Successful completion.
- 23884 1 The target was not up-to-date.
- 23885 >1 An error occurred.

23886 When the `-q` option is not specified, the *make* utility shall exit with one of the following values:

- 23887 0 Successful completion.
- 23888 >0 An error occurred.

### 23889 CONSEQUENCES OF ERRORS

23890 Default.

### 23891 APPLICATION USAGE

23892 If there is a source file (such as `./source.c`) and there are two SCCS files corresponding to it  
 23893 (`./s.source.c` and `./SCCS/s.source.c`), on XSI-conformant systems *make* uses the SCCS file in the  
 23894 current directory. However, users are advised to use the underlying SCCS utilities (*admin*, *delta*,  
 23895 *get*, and so on) or the *sccs* utility for all source files in a given directory. If both forms are used for  
 23896 a given source file, future developers are very likely to be confused.

23897 It is incumbent upon portable makefiles to specify the `.POSIX` special target in order to  
 23898 guarantee that they are not affected by local extensions.

23899 The `-k` and `-S` options are both present so that the relationship between the command line, the  
 23900 `MAKEFLAGS` variable, and the makefile can be controlled precisely. If the `k` flag is passed in

23901 *MAKEFLAGS* and a command is of the form:

23902 `$(MAKE) -S foo`

23903 then the default behavior is restored for the child *make*.

23904 When the `-n` option is specified, it is always added to *MAKEFLAGS*. This allows a recursive  
23905 *make -n target* to be used to see all of the action that would be taken to update *target*.

23906 Because of widespread historical practice, interpreting a '#' number sign inside a variable as  
23907 the start of a comment has the unfortunate side effect of making it impossible to place a number  
23908 sign in a variable, thus forbidding something like:

23909 `CFLAGS = "-D COMMENT_CHAR='#'"`

23910 Many historical *make* utilities stop chaining together inference rules when an intermediate target  
23911 is nonexistent. For example, it might be possible for a *make* to determine that both *.y.c* and *.c.o*  
23912 could be used to convert a *.y* to a *.o*. Instead, in this case, *make* requires the use of a *.y.o* rule.

23913 The best way to provide portable makefiles is to include all of the rules needed in the makefile  
23914 itself. The rules provided use only features provided by other parts of this volume of  
23915 IEEE Std. 1003.1-200x. The default rules include rules for optional commands in this volume of  
23916 IEEE Std. 1003.1-200x. Only rules pertaining to commands that are provided are needed in an  
23917 implementation's default set.

23918 Macros used within other macros are evaluated when the new macro is used rather than when  
23919 the new macro is defined. Therefore:

23920 `MACRO = value1`  
23921 `NEW = $(MACRO)`  
23922 `MACRO = value2`

23923 `target:`  
23924 `echo $(NEW)`

23925 would produce *value2* and not *value1* since *NEW* was not expanded until it was needed in the  
23926 *echo* command line.

23927 Some historical applications have been known to intermix *target\_name* and *macro=name* operands  
23928 on the command line, expecting that all of the macros are processed before any of the targets are  
23929 dealt with. Portable applications do not do this, although some backward compatibility support  
23930 may be included in some implementations.

23931 The following characters in file names may give trouble: '=', ':', '\', '\'', and '@'. For  
23932 inference rules, the description of *\$<* and *\$?* seem similar. However, an example shows the  
23933 minor difference. In a makefile containing:

23934 `foo.o: foo.h`

23935 if *foo.h* is newer than *foo.o*, yet *foo.c* is older than *foo.o*, the built-in rule to make *foo.o* from  
23936 *foo.c* is used, with *\$<* equal to *foo.c* and *\$?* equal to *foo.h*. If *foo.c* is also newer than *foo.o*, *\$<* is  
23937 equal to *foo.c* and *\$?* is equal to *foo.h foo.c*.

#### 23938 EXAMPLES

23939 1. The following command:

23940 `make`

23941 makes the first target found in the makefile.

- 23942           2. The following command:
- 23943           make junk
- 23944           makes the target **junk**.
- 23945           3. The following makefile says that **pgm** depends on two files, **a.o** and **b.o**, and that they in
- 23946           turn depend on their corresponding source files (**a.c** and **b.c**), and a common file **incl.h**:
- 23947           pgm: a.o b.o
- 23948                 c99 a.o b.o -o pgm
- 23949           a.o: incl.h a.c
- 23950                 c99 -c a.c
- 23951           b.o: incl.h b.c
- 23952                 c99 -c b.c
- 23953           4. An example for making optimized **.o** files from **.c** files is:
- 23954           .c.o:
- 23955                 c99 -c -O \$\*.c
- 23956           or:
- 23957           .c.o:
- 23958                 c99 -c -O \$<
- 23959           5. The most common use of the archive interface follows. Here, it is assumed that the source
- 23960           files are all C-language source:
- 23961           lib: lib(file1.o) lib(file2.o) lib(file3.o)
- 23962                 @echo lib is now up-to-date
- 23963           The **.c.a** rule is used to make **file1.o**, **file2.o**, and **file3.o** and insert them into **lib**.
- 23964           The treatment of escaped <newline> characters throughout the makefile is historical
- 23965           practice. For example, the inference rule:
- 23966           .c.o\  
23967           :
- 23968           works, and the macro:
- 23969           f= bar baz\  
23970                 biz
- 23971           a:  
23972                 echo ==\$f==
- 23973           echoes "==bar baz biz==".
- 23974           If **\$?** were:
- 23975           /usr/include/stdio.h /usr/include/unistd.h foo.h
- 23976           then **\$(?D)** would be:
- 23977           /usr/include /usr/include .
- 23978           and **\$(?F)** would be:
- 23979           stdio.h unistd.h foo.h
- 23980           6. The contents of the built-in rules can be viewed by running:

23981 `make -p -f /dev/null 2>/dev/null`

## 23982 RATIONALE

23983 The *make* utility described in this volume of IEEE Std. 1003.1-200x is intended to provide the  
 23984 means for changing portable source code into executables that can be run on a  
 23985 IEEE Std. 1003.1-200x-conforming system. It reflects the most common features present in  
 23986 System V and BSD *makes*.

23987 Historically, the *make* utility has been an especially fertile ground for vendor and research  
 23988 organization-specific syntax modifications and extensions. Examples include:

- 23989 • Syntax supporting parallel execution (such as from various multiprocessor vendors, GNU,  
 23990 and others)
- 23991 • Additional “operators” separating targets and their prerequisites (System V, BSD, and  
 23992 others)
- 23993 • Specifying that command lines containing the strings `$(MAKE)` and `$(MAKE)` are executed  
 23994 when the `-n` option is specified (GNU and System V)
- 23995 • Modifications of the meaning of internal macros when referencing libraries (BSD and others)
- 23996 • Using a single instance of the shell for all of the command lines of the target (BSD and others)
- 23997 • Allowing spaces as well as tabs to delimit command lines (BSD)
- 23998 • Adding C preprocessor-style “include” and “ifdef” constructs (System V, GNU, BSD, and  
 23999 others)
- 24000 • Remote execution of command lines (Sprite and others)
- 24001 • Specifying additional special targets (BSD, System V, and most others)

24002 Additionally, many vendors and research organizations have rethought the basic concepts of  
 24003 *make*, creating vastly extended, as well as completely new, syntaxes. Each of these versions of  
 24004 *make* fulfills the needs of a different community of users; it is unreasonable for this volume of  
 24005 IEEE Std. 1003.1-200x to require behavior that would be incompatible (and probably inferior) to  
 24006 historical practice for such a community.

24007 In similar circumstances, when the industry has enough sufficiently incompatible formats as to  
 24008 make them irreconcilable, this volume of IEEE Std. 1003.1-200x has followed one or both of two  
 24009 courses of action. Commands have been renamed (*cksum*, *echo*, and *pax*) and/or command line  
 24010 options have been provided to select the desired behavior (*grep*, *od*, and *pax*).

24011 Because the syntax specified for the *make* utility is, by and large, a subset of the syntaxes  
 24012 accepted by almost all versions of *make*, it was decided that it would be counter-productive to  
 24013 change the name. And since the makefile itself is a basic unit of portability, it would not be  
 24014 completely effective to reserve a new option letter, such as *make -P*, to achieve the portable  
 24015 behavior. Therefore, the special target `.POSIX` was added to the makefile, allowing users to  
 24016 specify “standard” behavior. This special target does not preclude extensions in the *make* utility,  
 24017 nor does it preclude such extensions being used by the makefile specifying the target; it does,  
 24018 however, preclude any extensions from being applied that could alter the behavior of previously  
 24019 valid syntax; such extensions must be controlled via command line options or new special  
 24020 targets. It is incumbent upon portable makefiles to specify the `.POSIX` special target in order to  
 24021 guarantee that they are not affected by local extensions.

24022 The portable version of *make* described in this reference page is not intended to be the state-of-  
 24023 the-art software generation tool and, as such, some newer and more leading-edge features have  
 24024 not been included. An attempt has been made to describe the portable makefile in a manner that  
 24025 does not preclude such extensions as long as they do not disturb the portable behavior described

24026 here.

24027 When the `-n` option is specified, it is always added to `MAKEFLAGS`. This allows a recursive  
24028 `make -n target` to be used to see all of the action that would be taken to update `target`.

24029 The definition of `MAKEFLAGS` allows both the System V letter string and the BSD command line  
24030 formats. The two formats are sufficiently different to allow implementations to support both  
24031 without ambiguity.

24032 Early proposals stated that an “unquoted” number sign was treated as the start of a comment.  
24033 The `make` utility does not pay any attention to quotes. A number sign starts a comment  
24034 regardless of its surroundings.

24035 The text about “other implementation-defined path names may also be tried” in addition to  
24036 `./makefile` and `./Makefile` is to allow such extensions as `SCCS/s.Makefile` and other variations.  
24037 It was made an implementation-defined requirement (as opposed to unspecified behavior) to  
24038 highlight surprising implementations that might select something unexpected like  
24039 `/etc/Makefile`. XSI-conformant systems also try `./s.makefile`, `SCCS/s.makefile`, `./s.Makefile`,  
24040 and `SCCS/s.Makefile`.

24041 Early proposals contained the macro `NPROC` as a means of specifying that `make` should use `n`  
24042 processes to do the work required. While this feature is a valuable extension for many systems, it  
24043 is not common usage and could require other non-trivial extensions to makefile syntax. This  
24044 extension is not required by this volume of IEEE Std. 1003.1-200x, but could be provided as a  
24045 compatible extension. The macro `PARALLEL` is used by some historical systems with essentially  
24046 the same meaning (but without using a name that is a common system limit value). It is  
24047 suggested that implementors recognize the existing use of `NPROC` and/or `PARALLEL` as  
24048 extensions to `make`.

24049 The default rules are based on System V. The default `CC=` value is `c99` instead of `cc` because this  
24050 volume of IEEE Std. 1003.1-200x does not standardize the utility named `cc`. Thus, every  
24051 conforming application would be required to define `CC=c99` to expect to run. There is no  
24052 advantage conferred by the hope that the makefile might hit the “preferred” compiler because  
24053 this cannot be guaranteed to work. Also, since the portable makescript can only use the `c99`  
24054 options, no advantage is conferred in terms of what the script can do. It is a quality-of-  
24055 implementation issue as to whether `c99` is as valuable as `cc`.

24056 The `-d` option to `make` is frequently used to produce debugging information, but is too  
24057 implementation-defined to add to this volume of IEEE Std. 1003.1-200x.

24058 The `-p` option is not passed in `MAKEFLAGS` on most historical implementations and to change  
24059 this would cause many implementations to break without sufficiently increased portability.

24060 Commands that begin with a plus sign (`' + '`) are executed even if the `-n` option is present. Based  
24061 on the GNU version of `make`, the behavior of `-n` when the plus-sign prefix is encountered has  
24062 been extended to apply to `-q` and `-t` as well. However, the System V convention of forcing  
24063 command execution with `-n` when the command line of a target contains either of the strings  
24064 `$(MAKE)` or `${MAKE}` has not been adopted. This functionality appeared in early proposals, but  
24065 the danger of this approach was pointed out with the following example of a portion of a  
24066 makefile:

```
24067 subdir:
24068 cd subdir; rm all_the_files; $(MAKE)
```

24069 The loss of the System V behavior in this case is well-balanced by the safety afforded to other  
24070 makefiles that were not aware of this situation. In any event, the command line plus-sign prefix  
24071 can provide the desired functionality.



24072 The double colon in the target rule format is supported in BSD systems to allow more than one  
 24073 target line containing the same target name to have commands associated with it. Since this is  
 24074 not functionality described in the SVID or XPG3 it has been allowed as an extension, but not  
 24075 mandated.

24076 The default rules are provided with text specifying that the built-in rules shall be the same *as if*  
 24077 the listed set were used. The intent is that implementations should be able to use the rules  
 24078 without change, but will be allowed to alter them in ways that do not affect the primary  
 24079 behavior.

24080 The best way to provide portable makefiles is to include all of the rules needed in the makefile  
 24081 itself. The rules provided use only features provided by other portions of this volume of  
 24082 IEEE Std. 1003.1-200x. The default rules include rules for optional commands in this volume of  
 24083 IEEE Std. 1003.1-200x. Only rules pertaining to commands that are provided are needed in the  
 24084 default set of an implementation.

24085 One point of discussion was whether to drop the default rules list from this volume of  
 24086 IEEE Std. 1003.1-200x. They provide convenience, but do not enhance portability of applications.  
 24087 The prime benefit is in portability of users who wish to type *make command* and have the  
 24088 command build from a **command.c** file.

24089 The historical *MAKESHELL* feature was omitted. In some implementations it is used to let a user  
 24090 override the shell to be used to run *make* commands. This was confusing; for a portable *make*, the  
 24091 shell should be chosen by the makefile writer or specified on the *make* command line and not by  
 24092 a user running *make*.

24093 The *make* utilities in most historical implementations process the prerequisites of a target in left-  
 24094 to-right order, and the makefile format requires this. It supports the standard idiom used in  
 24095 many makefiles that produce yacc programs; for example:

```
24096 foo: y.tab.o lex.o main.o
24097 $(CC) $(CFLAGS) -o $@ t.tab.o lex.o main.o
```

24098 In this example, if *make* chose any arbitrary order, the **lex.o** might not be made with the correct  
 24099 **y.tab.h**. Although there may be better ways to express this relationship, it is widely used  
 24100 historically. Implementations that desire to update prerequisites in parallel should require an  
 24101 explicit extension to *make* or the makefile format to accomplish it, as described previously.

24102 The algorithm for determining a new entry for target rules is partially unspecified. Some  
 24103 historical *makes* allow blank, empty, or comment lines within the collection of commands  
 24104 marked by leading <tab>s. A conforming makefile must ensure that each command starts with  
 24105 a <tab>, but implementations are free to ignore blank, empty, and comment lines without  
 24106 triggering the start of a new entry.

24107 The ASYNCHRONOUS EVENTS section includes having SIGTERM and SIGHUP, along with  
 24108 the more traditional SIGINT and SIGQUIT, remove the current target unless directed not to do  
 24109 so. SIGTERM and SIGHUP were added to parallel other utilities that have historically cleaned  
 24110 up their work as a result of these signals. When *make* receives any signal other than SIGQUIT, it  
 24111 is required to resend itself the signal it received so that it exits with a status that reflects the  
 24112 signal. The results from SIGQUIT are partially unspecified because, on systems that create **core**  
 24113 files upon receipt of SIGQUIT, the **core** from *make* would conflict with a core file from the  
 24114 command that was running when the SIGQUIT arrived. The main concern was to prevent  
 24115 damaged files from appearing up-to-date when *make* is rerun.

24116 The **.PRECIOUS** special target was extended to affect all targets globally (by specifying no  
 24117 prerequisites). The **.IGNORE** and **.SILENT** special targets were extended to allow prerequisites;  
 24118 it was judged to be more useful in some cases to be able to turn off errors or echoing for a list of

24119 targets than for the entire makefile. These extensions to the *make* in System V were made to  
24120 match historical practice from the BSD *make*.

24121 Macros are not exported to the environment of commands to be run. This was never the case in  
24122 any historical *make* and would have serious consequences. The environment is the same as the  
24123 environment to *make* except that *MAKEFLAGS* and macros defined on the *make* command line  
24124 are added.

24125 Some implementations do not use *system()* for all command lines, as required by the portable  
24126 makefile format; as a performance enhancement, they select lines without shell metacharacters  
24127 for direct execution by *execve()*. There is no requirement that *system()* be used specifically, but  
24128 merely that the same results be achieved. The metacharacters typically used to bypass the direct  
24129 *execve()* execution have been any of:

24130 = | ^ ( ) ; & < > \* ? [ ] : \$ \ ' " \ \n

24131 The default in some advanced versions of *make* is to group all the command lines for a target and  
24132 execute them using a single shell invocation; the System V method is to pass each line  
24133 individually to a separate shell. The single-shell method has the advantages in performance and  
24134 the lack of a requirement for many continued lines. However, converting to this newer method  
24135 has caused portability problems with many historical makefiles, so the behavior with the POSIX  
24136 makefile is specified to be the same as that of System V. It is suggested that the special target  
24137 *.ONESHELL* be used as an implementation extension to achieve the single-shell grouping for a  
24138 target or group of targets.

24139 Novice users of *make* have had difficulty with the historical need to start commands with a  
24140 <tab> character. Since it is often difficult to discern differences between <tab> and <space>  
24141 characters on terminals or printed listings, confusing bugs can arise. In early proposals, an  
24142 attempt was made to correct this problem by allowing leading <blank>s instead of <tab>s.  
24143 However, implementors reported many makefiles that failed in subtle ways following this  
24144 change, and it is difficult to implement a *make* that unambiguously can differentiate between  
24145 macro and command lines. There is extensive historical practice of allowing leading spaces  
24146 before macro definitions. Forcing macro lines into column 1 would be a significant backwards-  
24147 compatibility problem for some makefiles. Therefore, historical practice was restored.

24148 The System V INCLUDE feature was considered, but not included. This would treat a line that  
24149 began in the first column and contained INCLUDE <filename> as an indication to read <filename>  
24150 at that point in the makefile. This is difficult to use in a portable way, and it raises concerns  
24151 about nesting levels and diagnostics. System V, BSD, GNU, and others have used different  
24152 methods for including files.

24153 The System V dynamic dependency feature was not included. It would support:

24154 cat: \$\$@.c

24155 that would expand to;

24156 cat: cat.c

24157 This feature exists only in the new version of System V *make* and, while useful, is not in wide  
24158 usage. This means that macros are expanded twice for prerequisites: once at makefile parse time  
24159 and once at target update time.

24160 Consideration was given to adding metarules to the POSIX *make*. This would make *%o*: *%c* the  
24161 same as *.c.o*:. This is quite useful and available from some vendors, but it would cause too many  
24162 changes to this *make* to support. It would have introduced rule chaining and new substitution  
24163 rules. However, the rules for target names have been set to reserve the '%' and '"' characters.  
24164 These are traditionally used to implement metarules and quoting of target names, respectively.

- 24165 Implementors are strongly encouraged to use these characters only for these purposes.
- 24166 A request was made to extend the suffix delimiter character from a period to any character. The  
24167 metarules feature in newer *makes* solves this problem in a more general way. This volume of  
24168 IEEE Std. 1003.1-200x is staying with the more conservative historical definition.
- 24169 The standard output format for the `-p` option is not described because it is primarily a  
24170 debugging option and because the format is not generally useful to programs. In historical  
24171 implementations the output is not suitable for use in generating makefiles. The `-p` format has  
24172 been variable across historical implementations. Therefore, the definition of `-p` was only to  
24173 provide a consistently named option for obtaining *make* script debugging information.
- 24174 Some historical implementations have not cleared the suffix list with `-r`.
- 24175 Implementations should be aware that some historical applications have intermixed *target\_name*  
24176 and *macro=value* operands on the command line, expecting that all of the macros are processed  
24177 before any of the targets are dealt with. Portable applications do not do this, but some  
24178 backwards-compatibility support may be warranted.
- 24179 Empty inference rules are specified with a semicolon command rather than omitting all  
24180 commands, as described in an early proposal. The latter case has no traditional meaning and is  
24181 reserved for implementation extensions, such as in GNU *make*.
- 24182 **FUTURE DIRECTIONS**
- 24183 None.
- 24184 **SEE ALSO**
- 24185 *ar*, *c99*, *get*, *lex*, *sh*, *yacc*, the System Interfaces volume of IEEE Std. 1003.1-200x, *system()*
- 24186 **CHANGE HISTORY**
- 24187 First released in Issue 2.
- 24188 **Issue 4**
- 24189 Aligned with the ISO/IEC 9945-2: 1993 standard.
- 24190 **Issue 4, Version 2**
- 24191 Under **Default Rules**, the string `"-G$@"` is deleted from the line referencing *sccs*.
- 24192 **Issue 5**
- 24193 **FUTURE DIRECTIONS** section added.
- 24194 **Issue 6**
- 24195 This utility is now marked as part of the Software Development Utilities option.
- 24196 The Open Group corrigenda item U029/1 has been applied, correcting a typographical error in  
24197 the SPECIAL TARGETS section.
- 24198 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from  
24199 “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of  
24200 *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.
- 24201 It is specified whether the command line is related to the makefile or to the *make* command, and  
24202 the macro processing rules are updated to align with the IEEE P1003.2b draft standard.
- 24203 The normative text is reworded to avoid use of the term “must” for application requirements.

24204 **NAME**

24205 man — display system documentation

24206 **SYNOPSIS**24207 man [-k] *name*...24208 **DESCRIPTION**

24209 The *man* utility shall write information about each of the *name* operands. If *name* is the name of a  
 24210 standard utility, *man* at a minimum shall write a message describing the syntax used by the  
 24211 standard utility, its options, and operands. If more information is available, the *man* utility shall  
 24212 provide it in an implementation-defined manner.

24213 An implementation may provide information for values of *name* other than the standard utilities.  
 24214 Standard utilities that are listed as optional and that are not supported by the implementation  
 24215 either shall cause a brief message indicating that fact to be displayed or shall cause a full display  
 24216 of information as described previously.

24217 **OPTIONS**

24218 The *man* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 24219 12.2, Utility Syntax Guidelines.

24220 The following option shall be supported:

24221 **-k** Interpret *name* operands as keywords to be used in searching a utilities summary  
 24222 database that contains a brief purpose entry for each standard utility and write lines  
 24223 from the summary database that match any of the keywords. The keyword search shall  
 24224 produce results that are the equivalent of the output of the following command:

```
24225 grep -Ei '
24226 name
24227 name
24228 ...
24229 ' summary-database
```

24230 This assumes that the *summary-database* is a text file with a single entry per line; this  
 24231 organization is not required and the example using *grep -Ei* is merely illustrative of the  
 24232 type of search intended. The purpose entry to be included in the database shall consist  
 24233 of a terse description of the purpose of the utility.

24234 **OPERANDS**

24235 The following operand shall be supported:

24236 *name* A keyword or the name of a standard utility. When **-k** is not specified and *name*  
 24237 does not represent one of the standard utilities, the results are unspecified.

24238 **STDIN**

24239 Not used.

24240 **INPUT FILES**

24241 None.

24242 **ENVIRONMENT VARIABLES**24243 The following environment variables shall affect the execution of *man*:

24244 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 24245 If *LANG* is unset or null, the corresponding value from the implementation-  
 24246 defined default locale shall be used. If any of the internationalization variables  
 24247 contains an invalid setting, the utility shall behave as if none of the variables had  
 24248 been defined.

- 24249 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
24250 internationalization variables.
- 24251 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
24252 characters (for example, single-byte as opposed to multi-byte characters in  
24253 arguments and in the summary database). The value of *LC\_CTYPE* need not affect  
24254 the format of the information written about the name operands.
- 24255 **LC\_MESSAGES**  
24256 Determine the locale that should be used to affect the format and contents of  
24257 diagnostic messages written to standard error and informative messages written to  
24258 standard output.
- 24259 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 24260 **PAGER** Determine an output filtering command for writing the output to a terminal. Any  
24261 string acceptable as a *command\_string* operand to the *sh -c* command shall be valid.  
24262 When standard output is a terminal device, the reference page output shall be  
24263 piped through the command. If the *PAGER* variable is null or not set, the  
24264 command shall be either *more* or another paginator utility documented in the  
24265 system documentation.
- 24266 **ASYNCHRONOUS EVENTS**  
24267 Default.
- 24268 **STDOUT**  
24269 The *man* utility shall write text describing the syntax of the utility *name*, its options and its  
24270 operands, or, when *-k* is specified, lines from the summary database. The format of this text is  
24271 implementation-defined.
- 24272 **STDERR**  
24273 Used only for diagnostic messages.
- 24274 **OUTPUT FILES**  
24275 None.
- 24276 **EXTENDED DESCRIPTION**  
24277 None.
- 24278 **EXIT STATUS**  
24279 The following exit values shall be returned:  
24280 0 Successful completion.  
24281 >0 An error occurred.
- 24282 **CONSEQUENCES OF ERRORS**  
24283 Default.
- 24284 **APPLICATION USAGE**  
24285 None.
- 24286 **EXAMPLES**  
24287 None.
- 24288 **RATIONALE**  
24289 It is recognized that the *man* utility is only of minimal usefulness as specified. The opinion of the  
24290 standard developers was strongly divided as to how much or how little information *man* should  
24291 be required to provide. They considered, however, that the provision of some portable way of  
24292 accessing documentation would aid user portability. The arguments against a fuller

- 24293 specification were:
- 24294 • Large quantities of documentation should not be required on a system that does not have
  - 24295 excess disk space.
  - 24296 • The current manual system does not present information in a manner that greatly aids user
  - 24297 portability.
  - 24298 • A “better help system” is currently an area in which vendors feel that they can add value to
  - 24299 their POSIX implementations.
- 24300 The `-f` option was considered, but due to implementation differences, it was not included in this
- 24301 volume of IEEE Std. 1003.1-200x.
- 24302 The description was changed to be more specific about what has to be displayed for a utility.
- 24303 The standard developers considered it insufficient to allow a display of only the synopsis
- 24304 without giving a short description of what each option and operand does.
- 24305 The “purpose” entry to be included in the database can be similar to the section title (less the
- 24306 numeric prefix) from this volume of IEEE Std. 1003.1-200x for each utility. These titles are
- 24307 similar to those used in historical systems for this purpose.
- 24308 See *mailx* for rationale concerning the default paginator.
- 24309 The caveat in the *LC\_CTYPE* description was added because it is not a requirement that an
- 24310 implementation provide reference pages for all of its supported locales on each system;
- 24311 changing *LC\_CTYPE* does not necessarily translate the reference page into another language.
- 24312 This is equivalent to the current state of *LC\_MESSAGES* in IEEE Std. 1003.1-200x—locale-specific
- 24313 messages are not yet a requirement.
- 24314 The historical *MANPATH* variable is not included in POSIX because no attempt is made to
- 24315 specify naming conventions for reference page files, nor even to mandate that they are files at
- 24316 all. In some systems they could be a true database, a hypertext file, or even fixed strings within
- 24317 the *man* executable. The standard developers considered the portability of reference pages to be
- 24318 outside their scope of work (and more appropriate to the POSIX.7 working group developing
- 24319 application-installation tools). However, users should be aware that *MANPATH* is implemented
- 24320 on a number of historical systems and that it can be used to tailor the search pattern for reference
- 24321 pages from the various categories (utilities, functions, file formats, and so on) when the system
- 24322 administrator reveals the location and conventions for reference pages on the system.
- 24323 The keyword search can rely on at least the text of the section titles from these utility
- 24324 descriptions, and the implementation may add more keywords. The term “section titles” refers
- 24325 to the strings such as:
- 24326 `man` – Display system documentation
- 24327 `ps` – Report process status
- 24328 **FUTURE DIRECTIONS**
- 24329 None.
- 24330 **SEE ALSO**
- 24331 *more*
- 24332 **CHANGE HISTORY**
- 24333 First released in Issue 4.

24334 **Issue 5**

24335 FUTURE DIRECTIONS section added.

24336 **NAME**

24337 mesg — permit or deny messages

24338 **SYNOPSIS**

24339 UP mesg [y|n]

24340

24341 **DESCRIPTION**

24342 The *mesg* utility shall control whether other users are allowed to send messages via *write*, *talk*, or  
 24343 other utilities to a terminal device. The terminal device affected shall be determined by searching  
 24344 for the first terminal in the sequence of devices associated with standard input, standard output,  
 24345 and standard error, respectively. With no arguments, *mesg* shall report the current state without  
 24346 changing it. Processes with appropriate privileges may be able to send messages to the terminal  
 24347 independent of the current state.

24348 **OPTIONS**

24349 None.

24350 **OPERANDS**

24351 The following operands shall be supported in the POSIX locale:

24352 *y* Grant permission to other users to send messages to the terminal device.24353 *n* Deny permission to other users to send messages to the terminal device.24354 **STDIN**

24355 Not used.

24356 **INPUT FILES**

24357 None.

24358 **ENVIRONMENT VARIABLES**24359 The following environment variables shall affect the execution of *mesg*:

24360 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 24361 If *LANG* is unset or null, the corresponding value from the implementation-  
 24362 defined default locale shall be used. If any of the internationalization variables  
 24363 contains an invalid setting, the utility shall behave as if none of the variables had  
 24364 been defined.

24365 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 24366 internationalization variables.

24367 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 24368 characters (for example, single-byte as opposed to multi-byte characters in  
 24369 arguments).

24370 *LC\_MESSAGES*

24371 Determine the locale that should be used to affect the format and contents of  
 24372 diagnostic messages written (by *mesg*) to standard error.

24373 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.24374 **ASYNCHRONOUS EVENTS**

24375 Default.

24376 **STDOUT**24377 If no operand is specified, *mesg* shall display the current terminal state in an unspecified format.



24378 **STDERR**

24379           Used only for diagnostic messages.

24380 **OUTPUT FILES**

24381           None.

24382 **EXTENDED DESCRIPTION**

24383           None.

24384 **EXIT STATUS**

24385           The following exit values shall be returned:

24386           0   Receiving messages is allowed.

24387           1   Receiving messages is not allowed.

24388           >1  An error occurred.

24389 **CONSEQUENCES OF ERRORS**

24390           Default.

24391 **APPLICATION USAGE**

24392           The mechanism by which the message status of the terminal is changed is unspecified.  
24393           Therefore, unspecified actions may cause the status of the terminal to change after *mesg* has  
24394           successfully completed. These actions may include, but are not limited to: another invocation of  
24395           the *mesg* utility, login procedures; invocation of the *stty* utility, invocation of the *chmod* utility or  
24396           *chmod()* function, and so on.

24397 **EXAMPLES**

24398           None.

24399 **RATIONALE**

24400           The terminal changed by *mesg* is that associated with the standard input, output, or error, rather  
24401           than the controlling terminal for the session. This is because users logged in more than once  
24402           should be able to change any of their login terminals without having to stop the job running in  
24403           those sessions. This is not a security problem involving the terminals of other users because  
24404           appropriate privileges would be required to affect the terminal of another user.

24405           The method of checking each of the first three file descriptors in sequence until a terminal is  
24406           found was adopted from System V.

24407           The file */dev/tty* is not specified for the terminal device because it was thought to be too  
24408           restrictive. Typical environment changes for the *n* operand are that write permissions are  
24409           removed for *others* and *group* from the appropriate device. It was decided to leave the actual  
24410           description of what is done as unspecified because of potential differences between  
24411           implementations.

24412           The format for standard output is unspecified because of differences between historical  
24413           implementations. This output is generally not useful to shell scripts (they can use the exit  
24414           status), so exact parsing of the output is unnecessary.

24415 **FUTURE DIRECTIONS**

24416           None.

24417 **SEE ALSO**

24418           *talk*, *write*

24419 **CHANGE HISTORY**

24420 First released in Issue 2.

24421 **Issue 4**

24422 Aligned with the ISO/IEC 9945-2: 1993 standard.

24423 **Issue 6**

24424 This utility is now marked as part of the User Portability Utilities option.

24425 **NAME**

24426           mkdir — make directories

24427 **SYNOPSIS**24428           mkdir [-p][-m *mode*] *dir*...24429 **DESCRIPTION**24430           The *mkdir* utility shall create the directories specified by the operands, in the order specified.24431           For each *dir* operand, the *mkdir* utility shall perform actions equivalent to the *mkdir()* function  
24432           defined in the System Interfaces volume of IEEE Std. 1003.1-200x, called with the following  
24433           arguments:

- 24434           1. The *dir* operand is used as the *path* argument.
- 24435           2. The value of the bitwise-inclusive OR of S\_IRWXU, S\_IRWXG, and S\_IRWXO is used as  
24436           the *mode* argument. (If the **-m** option is specified, the *mode* option-argument overrides this  
24437           default.)

24438 **OPTIONS**24439           The *mkdir* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
24440           12.2, Utility Syntax Guidelines.

24441           The following options shall be supported:

24442           **-m *mode***     Set the file permission bits of the newly-created directory to the specified *mode*  
24443           value. The *mode* option-argument shall be the same as the *mode* operand defined  
24444           for the *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-'  
24445           shall be interpreted relative to an assumed initial mode of *a=rwx*; '+' shall add  
24446           permissions to the default mode, '-' shall delete permissions from the default  
24447           mode.

24448           **-p**           Create any missing intermediate path name components.24449           For each *dir* operand that does not name an existing directory, effects equivalent to  
24450           those caused by the following command shall occur:

```
24451 mkdir -p -m $(umask -S),u+wX $(dirname dir) &&
24452 mkdir [-m mode] dir
```

24453           where the **-m *mode*** option represents that option supplied to the original  
24454           invocation of *mkdir*, if any.24455           Each *dir* operand that names an existing directory shall be ignored without error.24456 **OPERANDS**

24457           The following operand shall be supported:

24458           *dir*           A path name of a directory to be created.24459 **STDIN**

24460           Not used.

24461 **INPUT FILES**

24462           None.

24463 **ENVIRONMENT VARIABLES**24464           The following environment variables shall affect the execution of *mkdir*:

24465           *LANG*         Provide a default value for the internationalization variables that are unset or null.  
24466           If *LANG* is unset or null, the corresponding value from the implementation-  
24467           defined default locale shall be used. If any of the internationalization variables

- 24468 contains an invalid setting, the utility shall behave as if none of the variables had  
24469 been defined.
- 24470 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
24471 internationalization variables.
- 24472 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
24473 characters (for example, single-byte as opposed to multi-byte characters in  
24474 arguments).
- 24475 **LC\_MESSAGES**  
24476 Determine the locale that should be used to affect the format and contents of  
24477 diagnostic messages written to standard error.
- 24478 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 24479 **ASYNCHRONOUS EVENTS**
- 24480 Default.
- 24481 **STDOUT**
- 24482 Not used.
- 24483 **STDERR**
- 24484 Used only for diagnostic messages.
- 24485 **OUTPUT FILES**
- 24486 None.
- 24487 **EXTENDED DESCRIPTION**
- 24488 None.
- 24489 **EXIT STATUS**
- 24490 The following exit values shall be returned:
- 24491 0 All the specified directories were created successfully or the **-p** option was specified and all  
24492 the specified directories now exist.
- 24493 >0 An error occurred.
- 24494 **CONSEQUENCES OF ERRORS**
- 24495 Default.
- 24496 **APPLICATION USAGE**
- 24497 The default file mode for directories is *a=rwx* (777 on most systems) with selected permissions  
24498 removed in accordance with the file mode creation mask. For intermediate path name  
24499 components created by *mkdir*, the mode is the default modified by *u+wx* so that the  
24500 subdirectories can always be created regardless of the file mode creation mask; if different  
24501 ultimate permissions are desired for the intermediate directories, they can be changed  
24502 afterwards with *chmod*.
- 24503 Note that some of the requested directories may have been created even if an error occurs.
- 24504 **EXAMPLES**
- 24505 None.
- 24506 **RATIONALE**
- 24507 The System V **-m** option was included to control the file mode.
- 24508 The System V **-p** option was included to create any needed intermediate directories and to  
24509 complement the functionality provided by *rmdir* for removing directories in the path prefix as  
24510 they become empty. Because no error is produced if any path component already exists, the **-p**

- 24511 option is also useful to ensure that a particular directory exists.
- 24512 The functionality of *mkdir* is described substantially through a reference to the *mkdir()* function  
24513 in the System Interfaces volume of IEEE Std. 1003.1-200x. For example, by default, the mode of  
24514 the directory is affected by the file mode creation mask in accordance with the specified  
24515 behavior of the *mkdir()* function. In this way, there is less duplication of effort required for  
24516 describing details of the directory creation.
- 24517 **FUTURE DIRECTIONS**
- 24518 None.
- 24519 **SEE ALSO**
- 24520 *rm*, *rmdir*, *umask*, the System Interfaces volume of IEEE Std. 1003.1-200x, *mkdir()*
- 24521 **CHANGE HISTORY**
- 24522 First released in Issue 2.
- 24523 **Issue 4**
- 24524 Aligned with the ISO/IEC 9945-2:1993 standard.
- 24525 **Issue 5**
- 24526 FUTURE DIRECTIONS section added.

24527 **NAME**

24528           mkfifo — make FIFO special files

24529 **SYNOPSIS**24530           mkfifo [-m *mode*] *file*...24531 **DESCRIPTION**24532           The *mkfifo* utility shall create the FIFO special files specified by the operands, in the order  
24533           specified.24534           For each *file* operand, the *mkfifo* utility shall perform actions equivalent to the *mkfifo*() function  
24535           defined in the System Interfaces volume of IEEE Std. 1003.1-200x, called with the following  
24536           arguments:

- 24537           1. The *file* operand is used as the *path* argument.
- 24538           2. The value of the bitwise-inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP,  
24539           S\_IROTH, and S\_IWOTH is used as the *mode* argument. (If the **-m** option is specified, the  
24540           value of the *mkfifo*() *mode* argument is unspecified, but the FIFO shall at no time have  
24541           permissions less restrictive than the **-m mode** option-argument.)

24542 **OPTIONS**24543           The *mkfifo* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
24544           12.2, Utility Syntax Guidelines.

24545           The following option shall be supported:

24546           **-m mode**     Set the file permission bits of the newly-created FIFO to the specified *mode* value.  
24547                         The *mode* option-argument shall be the same as the *mode* operand defined for the  
24548                         *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-' shall be  
24549                         interpreted relative to an assumed initial mode of *a=rw*.

24550 **OPERANDS**

24551           The following operand shall be supported:

24552           *file*           A path name of the FIFO special file to be created.24553 **STDIN**

24554           Not used.

24555 **INPUT FILES**

24556           None.

24557 **ENVIRONMENT VARIABLES**24558           The following environment variables shall affect the execution of *mkfifo*:

24559           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
24560                         If *LANG* is unset or null, the corresponding value from the implementation-  
24561                         defined default locale shall be used. If any of the internationalization variables  
24562                         contains an invalid setting, the utility shall behave as if none of the variables had  
24563                         been defined.

24564           **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
24565                         internationalization variables.

24566           **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
24567                         characters (for example, single-byte as opposed to multi-byte characters in  
24568                         arguments).

24569           **LC\_MESSAGES**

24570                         Determine the locale that should be used to affect the format and contents of

- 24571 diagnostic messages written to standard error.
- 24572 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 24573 **ASYNCHRONOUS EVENTS**
- 24574 Default.
- 24575 **STDOUT**
- 24576 Not used.
- 24577 **STDERR**
- 24578 Used only for diagnostic messages.
- 24579 **OUTPUT FILES**
- 24580 None.
- 24581 **EXTENDED DESCRIPTION**
- 24582 None.
- 24583 **EXIT STATUS**
- 24584 The following exit values shall be returned:
- 24585 0 All the specified FIFO special files were created successfully.
- 24586 >0 An error occurred.
- 24587 **CONSEQUENCES OF ERRORS**
- 24588 Default.
- 24589 **APPLICATION USAGE**
- 24590 None.
- 24591 **EXAMPLES**
- 24592 None.
- 24593 **RATIONALE**
- 24594 This new utility was added to permit shell applications to create FIFO special files.
- 24595 The **-m** option was added to control the file mode, for consistency with the similar functionality provided the *mkdir* utility.
- 24596
- 24597 Early proposals included a **-p** option similar to the *mkdir -p* option that created intermediate directories leading up to the FIFO specified by the final component. This was removed because it is not commonly needed and is not common practice with similar utilities.
- 24598
- 24599
- 24600 The functionality of *mkfifo* is described substantially through a reference to the *mkfifo()* function in the System Interfaces volume of IEEE Std. 1003.1-200x. For example, by default, the mode of the FIFO file is affected by the file mode creation mask in accordance with the specified behavior of the *mkfifo()* function. In this way, there is less duplication of effort required for describing details of the file creation.
- 24601
- 24602
- 24603
- 24604
- 24605 **FUTURE DIRECTIONS**
- 24606 None.
- 24607 **SEE ALSO**
- 24608 *umask*, the System Interfaces volume of IEEE Std. 1003.1-200x, *mkfifo()*
- 24609 **CHANGE HISTORY**
- 24610 First released in Issue 3.

24611 **Issue 4**

24612 Aligned with the ISO/IEC 9945-2: 1993 standard.



## 24613 NAME

24614 more — display files on a page-by-page basis

## 24615 SYNOPSIS

24616 UP more [-ceisu][-n number][-p command][-t tagstring][file ...]

24617

## 24618 DESCRIPTION

24619 The *more* utility shall read files and either write them to the terminal on a page-by-page basis or  
 24620 filter them to standard output. If standard output is not a terminal device, all input files shall be  
 24621 copied to standard output in their entirety, without modification, except as specified for the *-s*  
 24622 option. If standard output is a terminal device, the files shall be written a number of lines (one  
 24623 screenful) at a time under the control of user commands. See the EXTENDED DESCRIPTION  
 24624 section.

24625 Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 24626 *more* definition; they are incapable of accepting commands that are not terminated with a  
 24627 <newline> character. Implementations that support such terminals shall provide an operating  
 24628 mode to *more* in which all commands can be terminated with a <newline> character on those  
 24629 terminals. This mode:

- 24630 • Shall be documented in the system documentation
- 24631 • Shall, at invocation, inform the user of the terminal deficiency that requires the <newline>  
 24632 character usage and provide instructions on how this warning can be suppressed in future  
 24633 invocations
- 24634 • Shall not be required for implementations supporting only fully capable terminals
- 24635 • Shall not affect commands already requiring <newline> characters
- 24636 • Shall not affect users on the capable terminals from using *more* as described in this volume of  
 24637 IEEE Std. 1003.1-200x

## 24638 OPTIONS

24639 The *more* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 24640 12.2, Utility Syntax Guidelines.

24641 The following options shall be supported:

- 24642 **-c** If a screen is to be written that has no lines in common with the current screen, or  
 24643 *more* is writing its first screen, *more* does not scroll the screen, but instead redraws  
 24644 each line of the screen in turn, from the top of the screen to the bottom. In addition,  
 24645 if *more* is writing its first screen, the screen is cleared. This option may be silently  
 24646 ignored on devices with insufficient terminal capabilities.
- 24647 **-e** By default, *more* shall exit immediately after writing the last line of the last file in  
 24648 the argument list. If the *-e* option is specified:
  - 24649 1. If there is only a single file in the argument list and that file was completely  
 24650 displayed on a single screen, *more* shall exit immediately after writing the last  
 24651 line of that file.
  - 24652 2. Otherwise, *more* shall exit only after reaching end-of-file on the last file in the  
 24653 argument list twice without an intervening operation. See the EXTENDED  
 24654 DESCRIPTION section.
- 24655 **-i** Perform pattern matching in searches without regard to case; see the Base  
 24656 Definitions volume of IEEE Std. 1003.1-200x, Section 9.2, Regular Expression  
 24657 General Requirements .

- 24658        **-n number**   Specify the number of lines per screenful. The *number* argument is a positive  
24659                    decimal integer. The **-n** option shall override any values obtained from any other  
24660                    source.
- 24661        **-p command** Each time a screen from a new file is displayed or redisplayed (including as a  
24662                    result of *more* commands; for example, **:p**), execute the *more* command(s) in the  
24663                    command arguments in the order specified, as if entered by the user after the first  
24664                    screen has been displayed. No intermediate results shall be displayed (that is, if the  
24665                    command is a movement to a screen different than the normal first screen, only the  
24666                    screen resulting from the command shall be displayed.) If any of the commands  
24667                    fail for any reason, an informational message to this effect shall be written, and no  
24668                    further commands specified using the **-p** option shall be executed for this file.
- 24669        **-s**                Behave as if consecutive empty lines were a single empty line.
- 24670        **-t tagstring** Write the screenful of the file containing the tag named by the *tagstring* argument.  
24671                    See the *ctags* utility. The tags feature represented by **-t tagstring** and the **:t**  
24672                    command is optional. It shall be provided on any system that also provides a  
24673                    conforming implementation of *ctags*; otherwise, the use of **-t** produces undefined  
24674                    results.
- 24675                    The file name resulting from the **-t** option shall be logically added as a prefix to the  
24676                    list of command line files, as if specified by the user. If the tag named by the  
24677                    *tagstring* argument is not found, it shall be an error, and *more* shall take no further  
24678                    action.
- 24679                    If the tag specifies a line number, the first line of the display shall contain the  
24680                    beginning of that line. If the tag specifies a pattern, the first line of the display shall  
24681                    contain the beginning of the matching text from the first line of the file that  
24682                    contains that pattern. If the line does not exist in the file or matching text is not  
24683                    found, an informational message to this effect shall be displayed, and *more* shall  
24684                    display the default screen as if **-t** had not been specified.
- 24685                    If both the **-t tagstring** and **-p command** options are given, the **-t tagstring** shall be  
24686                    processed first; that is, the file and starting line for the display shall be as specified  
24687                    by **-t**, and then the **-p more** command shall be executed. If the line (matching text)  
24688                    specified by the **-t** command does not exist (is not found), no **-p more** command  
24689                    shall be executed for this file at any time.
- 24690        **-u**                Treat a <backspace> character as a printable control character, displayed as an  
24691                    implementation-defined character sequence (see the EXTENDED DESCRIPTION  
24692                    section), suppressing backspacing and the special handling that produces  
24693                    underlined or standout mode text on some terminal types. Also, do not ignore a  
24694                    <carriage-return> character at the end of a line.

#### 24695 OPERANDS

24696        The following operand shall be supported:

- 24697        **file**                A path name of an input file. If no *file* operands are specified, the standard input  
24698                    shall be used. If a *file* is '-', the standard input shall be read at that point in the  
24699                    sequence.

#### 24700 STDIN

24701        The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.

24702 **INPUT FILES**

24703 The input files being examined shall be text files. If standard output is a terminal, standard error  
 24704 shall be used to read commands from the user. If standard output is a terminal, standard error is  
 24705 not readable, and command input is needed, *more* may attempt to obtain user commands from  
 24706 the controlling terminal (for example, */dev/tty*); otherwise, *more* shall terminate with an error  
 24707 indicating that it was unable to read user commands. If standard output is not a terminal, no  
 24708 error shall result if standard error cannot be opened for reading.

24709 **ENVIRONMENT VARIABLES**

24710 The following environment variables shall affect the execution of *more*:

24711 *COLUMNS* Override the system-selected horizontal screen size. See the Base Definitions  
 24712 volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid  
 24713 values and results when it is unset or null.

24714 *EDITOR* Used by the *v* command to select an editor. See the EXTENDED DESCRIPTION  
 24715 section.

24716 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 24717 If *LANG* is unset or null, the corresponding value from the implementation-  
 24718 defined default locale shall be used. If any of the internationalization variables  
 24719 contains an invalid setting, the utility shall behave as if none of the variables had  
 24720 been defined.

24721 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 24722 internationalization variables.

24723 *LC\_COLLATE*

24724 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 24725 character collating elements within regular expressions.

24726 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 24727 characters (for example, single-byte as opposed to multi-byte characters in  
 24728 arguments and input files) and the behavior of character classes within regular  
 24729 expressions.

24730 *LC\_MESSAGES*

24731 Determine the locale that should be used to affect the format and contents of  
 24732 diagnostic messages written to standard error and informative messages written to  
 24733 standard output.

24734 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

24735 *LINES* Override the system-selected vertical screen size, used as the number of lines in a  
 24736 screenful. See the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8,  
 24737 Environment Variables for valid values and results when it is unset or null. The *-n*  
 24738 option shall take precedence over the *LINES* variable for determining the number  
 24739 of lines in a screenful.

24740 *MORE* Determine a string containing options described in the OPTIONS section preceded  
 24741 with hyphens and <blank> character-separated as on the command line. Any  
 24742 command line options shall be processed after those in the *MORE* variable, as if  
 24743 the command line were:

24744 `more $MORE options operands`

24745 The *MORE* variable shall take precedence over the *TERM* and *LINES* variables for  
 24746 determining the number of lines in a screenful.

- 24747           **TERM**           Determine the name of the terminal type. If this variable is unset or null, an  
24748                               unspecified default terminal type is used.
- 24749 **ASYNCHRONOUS EVENTS**
- 24750           Default.
- 24751 **STDOUT**
- 24752           The standard output shall be used to write the contents of the input files.
- 24753 **STDERR**
- 24754           Used for diagnostic messages and user commands (see the INPUT FILES section), and, if  
24755           standard output is a terminal device, to write a prompting string. The prompting string shall  
24756           appear on the screen line below the last line of the file displayed in the current screenful. The  
24757           prompt shall contain the name of the file currently being examined and shall contain an end-of-  
24758           file indication and the name of the next file, if any, when prompting at the end-of-file. If an error  
24759           or informational message is displayed, it is unspecified whether it is contained in the prompt. If  
24760           it is not contained in the prompt, it shall be displayed and then the user shall be prompted for a  
24761           continuation character, at which point another message or the user prompt may be displayed.  
24762           The prompt is otherwise unspecified. It is unspecified whether informational messages are  
24763           written for other user commands.
- 24764 **OUTPUT FILES**
- 24765           None.
- 24766 **EXTENDED DESCRIPTION**
- 24767           The following subsection describes the behavior of *more* when the standard output is a terminal  
24768           device. If the standard output is not a terminal device, no options other than **-s** shall have any  
24769           effect, and all input files shall be copied to standard output otherwise unmodified, at which time  
24770           *more* shall exit without further action.
- 24771           The number of lines available per screen shall be determined by the **-n** option, if present, or by  
24772           examining values in the environment (see the ENVIRONMENT VARIABLES section). If neither  
24773           method yields a number, an unspecified number of lines shall be used.
- 24774           The maximum number of lines written shall be one less than this number, because the screen  
24775           line after the last line written shall be used to write a user prompt and user input. If the number  
24776           of lines in the screen is less than two, the results are undefined. It is unspecified whether user  
24777           input is permitted to be longer than the remainder of the single line where the prompt has been  
24778           written.
- 24779           The number of columns available per line shall be determined by examining values in the  
24780           environment (see the ENVIRONMENT VARIABLES section), with a default value as described  
24781           in Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.
- 24782           Lines that are longer than the display shall be folded; the length at which folding occurs is  
24783           unspecified, but should be appropriate for the output device. Folding may occur between glyphs  
24784           of single characters that take up multiple display columns.
- 24785           When standard output is a terminal and **-u** is not specified, *more* shall treat <backspace>  
24786           characters and <carriage-return> characters specially:
- 24787           • A character, followed first by a sequence of *n* <backspace> characters (where *n* is the same as  
24788           the number of column positions that the character occupies), then by *n* underscore characters  
24789           (' \_ '), shall cause that character to be written as underlined text, if the terminal type  
24790           supports that. The *n* underscore characters, followed first by *n* <backspace> characters, then  
24791           any character with *n* column positions, shall also cause that character to be written as  
24792           underlined text, if the terminal type supports that.

- 24793 • A sequence of *n* <backspace> characters (where *n* is the same as the number of column  
24794 positions that the previous character occupies) that appears between two identical printable  
24795 characters shall cause the first of those two characters to be written as emboldened text (that  
24796 is, visually brighter, standout mode, or inverse-video mode), if the terminal type supports  
24797 that, and the second to be discarded. Immediately subsequent occurrences of  
24798 <backspace>/character pairs for that same character also shall be discarded. (For example,  
24799 the sequence "a\ba\ba\ba" is interpreted as a single emboldened 'a'.)
- 24800 • The *more* utility shall logically discard all other <backspace> characters from the line as well  
24801 as the character which precedes them, if any.
- 24802 • A <carriage-return> character at the end of a line shall be ignored, rather than being written  
24803 as a non-printable character, as described in the next paragraph.
- 24804 It is implementation-defined how other non-printable characters are written. Implementations  
24805 should use the same format that they use for the *ex print* command; see the OPTIONS section  
24806 within the *ed* utility. It is unspecified whether a multi-column character shall be separated if it  
24807 crosses a logical line boundary; it shall not be discarded. The behavior is unspecified if the  
24808 number of columns on the display is less than the number of columns any single character in the  
24809 line being displayed would occupy.
- 24810 When each new file is displayed (or redisplayed), *more* shall write the first screen of the file.  
24811 Once the initial screen has been written, *more* shall prompt for a user command. If the execution  
24812 of the user command results in a screen that has lines in common with the current screen, and  
24813 the device has sufficient terminal capabilities, *more* shall scroll the screen; otherwise, it is  
24814 unspecified whether the screen is scrolled or redrawn.
- 24815 For all files but the last (including standard input if no file was specified, and for the last file as  
24816 well, if the *-e* option was not specified), when *more* has written the last line in the file, *more* shall  
24817 prompt for a user command. This prompt shall contain the name of the next file as well as an  
24818 indication that *more* has reached end-of-file. If the user command is *f*, <control>-F, <space>, *j*,  
24819 <newline>, *d*, <control>-D, or *s*, *more* shall display the next file. Otherwise, if displaying the last  
24820 file, *more* shall exit. Otherwise, *more* shall execute the user command specified.
- 24821 Several of the commands described in this section display a previous screen from the input  
24822 stream. In the case that text is being taken from a non-rewindable stream, such as a pipe, it is  
24823 implementation-defined how much backwards motion is supported. If a command cannot be  
24824 executed because of a limitation on backwards motion, an error message to this effect shall be  
24825 displayed, the current screen shall not change, and the user shall be prompted for another  
24826 command.
- 24827 If a command cannot be performed because there are insufficient lines to display, *more* shall alert  
24828 the terminal. If a command cannot be performed because there are insufficient lines to display or  
24829 a / command fails: if the input is the standard input, the last screen in the file may be displayed;  
24830 otherwise, the current file and screen shall not change, and the user shall be prompted for  
24831 another command.
- 24832 The interactive commands in the following sections shall be supported. Some commands can be  
24833 preceded by a decimal integer, called *count* in the following descriptions. If not specified with  
24834 the command, *count* shall default to 1. In the following descriptions, *pattern* is a basic regular  
24835 expression, as described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3,  
24836 Basic Regular Expressions. The term “examine” is historical usage meaning “open the file for  
24837 viewing”; for example, *more foo* would be expressed as examining file *foo*. In the following  
24838 descriptions, unless otherwise specified, *line* is a logical line in the *more* display, not a line from  
24839 the file being examined.

24840 In the following descriptions, the *current position* refers to two things:

- 24841 1. The position of the current line on the screen
- 24842 2. The line number (in the file) of the current line on the screen

24843 Usually, the line on the screen corresponding to the current position is the third line on the  
 24844 screen. If this is not possible (there are fewer than three lines to display or this is the first page of  
 24845 the file, or it is the last page of the file), then the current position is either the first or last line on  
 24846 the screen as described later.

## 24847 **Help**

24848 *Synopsis:*     h

24849 Write a summary of these commands and other implementation-defined commands. The  
 24850 behavior shall be as if the *more* utility were executed with the *-e* option on a file that contained  
 24851 the summary information. The user shall be prompted as described earlier in this section when  
 24852 end-of-file is reached. If the user command is one of those specified to continue to the next file,  
 24853 *more* shall return to the file and screen state from which the **h** command was executed.

## 24854 **Scroll Forward One Screenful**

24855 *Synopsis:*     [ *count* ]f  
 24856                 [ *count* ]<control>-F

24857 Scroll forward *count* lines, with a default of one screenful. If *count* is more than the screen size,  
 24858 only the final screenful shall be written.

## 24859 **Scroll Backward One Screenful**

24860 *Synopsis:*     [ *count* ]b  
 24861                 [ *count* ]<control>-B

24862 Scroll backward *count* lines, with a default of one screenful (see the *-n* option). If *count* is more  
 24863 than the screen size, only the final screenful shall be written.

## 24864 **Scroll Forward One Line**

24865 *Synopsis:*     [ *count* ]<space>  
 24866                 [ *count* ]j  
 24867                 [ *count* ]<newline>

24868 Scroll forward *count* lines. The default *count* for the <space> character shall be one screenful; for **j**  
 24869 and <newline> character, one line. The entire *count* lines shall be written, even if *count* is more  
 24870 than the screen size.

## 24871 **Scroll Backward One Line**

24872 *Synopsis:*     [ *count* ]k

24873 Scroll backward *count* lines. The entire *count* lines shall be written, even if *count* is more than the  
 24874 screen size.

**24875 Scroll Forward One Half Screenful**

24876 *Synopsis:* [count]d  
24877 [count]<control>-D

24878 Scroll forward *count* lines, with a default of one half of the screen size. If *count* is specified, it  
24879 shall become the new default for subsequent **d**, <control>-D, and **u** commands.

**24880 Skip Forward One Line**

24881 *Synopsis:* [count]s

24882 Display the screenful beginning with the line *count* lines after the last line on the current screen.  
24883 If *count* would cause the current position to be such that less than one screenful would be  
24884 written, the last screenful in the file shall be written.

**24885 Scroll Backward One Half Screenful**

24886 *Synopsis:* [count]u  
24887 [count]<control>-U

24888 Scroll backward *count* lines, with a default of one half of the screen size. If *count* is specified, it  
24889 shall become the new default for subsequent **d**, <control>-D, **u**, and <control>-U commands.  
24890 The entire *count* lines shall be written, even if *count* is more than the screen size.

**24891 Go to Beginning of File**

24892 *Synopsis:* [count]g

24893 Display the screenful beginning with line *count*.

**24894 Go to End-of-File**

24895 *Synopsis:* [count]G

24896 If *count* is specified, display the screenful beginning with the line *count*. Otherwise, display the  
24897 last screenful of the file.

**24898 Refresh the Screen**

24899 *Synopsis:* r  
24900 <control>-L

24901 Refresh the screen.

**24902 Discard and Refresh**

24903 *Synopsis:* R

24904 Refresh the screen, discarding any buffered input. If the current file is non-seekable, buffered  
24905 input shall not be discarded and the **R** command is equivalent to the **r** command.

**24906 Mark Position**

24907 *Synopsis:* `mletter`

24908 Mark the current position with the letter named by *letter*, where *letter* represents the name of one  
24909 of the lowercase letters of the portable character set. When a new file is examined, all marks may  
24910 be lost.

**24911 Return to Mark**

24912 *Synopsis:* `'letter`

24913 Return to the position that was previously marked with the letter named by *letter*, making that  
24914 line the current position.

**24915 Return to Previous Position**

24916 *Synopsis:* `''`

24917 Return to the position from which the last large movement command was executed (where a  
24918 "large movement" is defined as any movement of more than a screenful of lines). If no such  
24919 movements have been made, return to the beginning of the file.

**24920 Search Forward for Pattern**

24921 *Synopsis:* `[count]/[!]pattern<newline>`

24922 Display the screenful beginning with the *count*th line containing the pattern. The search shall  
24923 start after the first line currently displayed. The null regular expression (`'/'` followed by a  
24924 `<newline>` character) shall repeat the search using the previous regular expression, with a  
24925 default *count*. If the character `'!'` is included, the matching lines shall be those that do not  
24926 contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be  
24927 displayed.

**24928 Search Backward for Pattern**

24929 *Synopsis:* `[count]?[!]pattern<newline>`

24930 Display the screenful beginning with the *count*th previous line containing the pattern. The  
24931 search shall start on the last line before the first line currently displayed. The null regular  
24932 expression (`'?'` followed by a `<newline>` character) shall repeat the search using the previous  
24933 regular expression, with a default *count*. If the character `'!'` is included, matching lines shall be  
24934 those that do not contain the *pattern*.

24935 If no match is found for the *pattern*, a message to that effect shall be displayed.

**24936 Repeat Search**

24937 *Synopsis:* `[count]n`

24938 Repeat the previous search for *count*th line containing the last *pattern* (or not containing the last  
24939 *pattern*, if the previous search was `"/!"` or `"?!"`).



24940 **Repeat Search in Reverse**24941 *Synopsis:* [count]N24942 Repeat the search in the opposite direction of the previous search for the *count*th line containing  
24943 the last *pattern* (or not containing the last *pattern*, if the previous search was `"/!"` or `"?!"`).24944 **Examine New File**24945 *Synopsis:* :e [*filename*]<newline>24946 Examine a new file. If the *filename* argument is not specified, the current file (see the `:n` and `:p`  
24947 commands below) shall be re-examined. The *filename* shall be subjected to the process of shell  
24948 word expansions (see Section 2.6 (on page 2244)); if more than a single path name results, the  
24949 effects are unspecified. If *filename* is a number sign (`'#'`), the previously examined file shall be  
24950 re-examined. If *filename* is not accessible for any reason (including that it is a non-seekable file),  
24951 an error message to this effect shall be displayed and the current file and screen shall not change.24952 **Examine Next File**24953 *Synopsis:* [count]:n24954 Examine the next file. If a number *count* is specified, the *count*th next file shall be examined. If  
24955 *filename* refers to a non-seekable file, the results are unspecified.24956 **Examine Previous File**24957 *Synopsis:* [count]:p24958 Examine the previous file. If a number *count* is specified, the *count*th previous file shall be  
24959 examined. If *filename* refers to a non-seekable file, the results are unspecified.24960 **Go to Tag**24961 *Synopsis:* :t *tagstring*<newline>24962 If the file containing the tag named by the *tagstring* argument is not the current file, examine the  
24963 file, as if the `:e` command was executed with that file as the argument. Otherwise, or in addition,  
24964 display the screenful beginning with the tag, as described for the `-t` option (see the OPTIONS  
24965 section). If the *ctags* utility is not supported by the system, the use of `:t` produces undefined  
24966 results.24967 **Invoke Editor**24968 *Synopsis:* v24969 Invoke an editor to edit the current file being examined. If standard input is being examined, the  
24970 results are unspecified. The name of the editor shall be taken from the environment variable  
24971 *EDITOR*, or shall default to *vi*. If the last path name component in *EDITOR* is either *vi* or *ex*, the  
24972 editor shall be invoked with a `-c linenumber` command line argument, where *linenumber* is the  
24973 line number of the physical line containing the logical line currently displayed as the first line of  
24974 the screen. It is implementation-defined whether line-setting options are passed to editors other  
24975 than *vi* and *ex*.24976 When the editor exits, *more* shall resume with the same file and screen as when the editor was  
24977 invoked.

24978 **Display Position**

24979 *Synopsis:* =  
 24980 <control>-G

24981 Write a message for which the information references the first byte of the line after the last line of  
 24982 the file on the screen. This message shall include the name of the file currently being examined,  
 24983 its number relative to the total number of files there are to examine, the physical line number,  
 24984 the byte number and the total bytes in the file, and what percentage of the file precedes the  
 24985 current position. If *more* is reading from standard input, or the file is shorter than a single screen,  
 24986 the line number, the byte number, the total bytes, and the percentage need not be written.

24987 **Quit**

24988 *Synopsis:* q  
 24989 :q  
 24990 ZZ

24991 Exit *more*.

24992 **EXIT STATUS**

24993 The following exit values shall be returned:

24994 0 Successful completion.  
 24995 >0 An error occurred.

24996 **CONSEQUENCES OF ERRORS**

24997 If an error is encountered accessing a file when using the **:n** command, *more* shall attempt to  
 24998 examine the next file in the argument list, but the final exit status shall be affected. If an error is  
 24999 encountered accessing a file via the **:p** command, *more* shall attempt to examine the previous file  
 25000 in the argument list, but the final exit status shall be affected. If an error is encountered accessing  
 25001 a file via the **:e** command, *more* shall remain in the current file and the final exit status shall not  
 25002 be affected.

25003 **APPLICATION USAGE**

25004 When the standard output is not a terminal, only the **-s** filter-modification option is effective.  
 25005 This is based on historical practice. For example, a typical implementation of *man* pipes its  
 25006 output through *more -s* to squeeze excess white space for terminal users. When *man* is piped to  
 25007 *lp*, however, it is undesirable for this squeezing to happen.

25008 **EXAMPLES**

25009 The **-p** allows arbitrary commands to be executed at the start of each file. Examples are:

25010 *more -p G file1 file2*  
 25011 Examine each file starting with its last screenful.

25012 *more -p 100 file1 file2*  
 25013 Examine each file starting with line 100 in the current position (usually the third line, so line  
 25014 98 would be the first line written).

25015 *more -p /100 file1 file2*  
 25016 Examine each file starting with the first line containing the string "100" in the current  
 25017 position

25018 **RATIONALE**

25019 The *more* utility, available in BSD and BSD-derived systems, was chosen as the prototype for the  
 25020 POSIX file display program since it is more widely available than either the public-domain  
 25021 program *less* or than *pg*, a pager provided in System V. The 4.4 BSD *more* is the model for the

25022 features selected; it is almost fully upward-compatible from the 4.3 BSD version in wide use and  
 25023 has become more amenable for *vi* users. Several features originally derived from various file  
 25024 editors, found in both *less* and *pg*, have been added to this volume of IEEE Std. 1003.1-200x as  
 25025 they have proved extremely popular with users.

25026 There are inconsistencies between *more* and *vi* that result from historical practice. For example,  
 25027 the single-character commands **h**, **f**, **b**, and <space> are screen movers in *more*, but cursor  
 25028 movers in *vi*. These inconsistencies were maintained because the cursor movements are not  
 25029 applicable to *more* and the powerful functionality achieved without the use of the control key  
 25030 justifies the differences.

25031 The tags interface has been included in a program that is not a text editor because it promotes  
 25032 another degree of consistent operation with *vi*. It is conceivable that the paging environment of  
 25033 *more* would be superior for browsing source code files in some circumstances.

25034 The operating mode referred to for block-mode terminals effectively adds a <newline> to each  
 25035 Synopsis line that currently has none. So, for example, **d**<newline> would page one screenful.  
 25036 The mode could be triggered by a command line option, environment variable, or some other  
 25037 method. The details are not imposed by this volume of IEEE Std. 1003.1-200x because there are  
 25038 so few systems known to support such terminals. Nevertheless, it was considered that all  
 25039 systems should be able to support *more* given the exception cited for this small community of  
 25040 terminals because, in comparison to *vi*, the cursor movements are few and the command set  
 25041 relatively amenable to the optional <newline>s.

25042 Some versions of *more* provide a shell escaping mechanism similar to the *ex* ! command. The  
 25043 standard developers did not consider that this was necessary in a paginator, particularly given  
 25044 the wide acceptance of multiple window terminals and job control features. (They chose to  
 25045 retain such features in the editors and *mailx* because the shell interaction also gives an  
 25046 opportunity to modify the editing buffer, which is not applicable to *more*).

25047 The **-p** (position) option replaces the **+** command because of the Utility Syntax Guidelines. In  
 25048 early proposals, it took a *pattern* argument, but historical *less* provided the *more* general facility of  
 25049 a command. It would have been desirable to use the same **-c** as *ex* and *vi*, but the letter was  
 25050 already in use.

25051 The text stating “from a non-rewindable stream ... implementations may limit the amount of  
 25052 backwards motion supported” would allow an implementation that permitted no backwards  
 25053 motion beyond text already on the screen. It was not possible to require a minimum amount of  
 25054 backwards motion that would be effective for all conceivable device types. The implementation  
 25055 should allow the user to back up as far as possible, within device and reasonable memory  
 25056 allocation constraints.

25057 Historically, non-printable characters were displayed using the ARPA standard mappings,  
 25058 which are as follows:

- 25059 1. Printable characters are left alone.
- 25060 2. Control characters less than \177 are represented as followed by the character offset from  
 25061 the '@' character in the ASCII map; for example, \007 is represented as 'G'.
- 25062 3. \177 is represented as followed by '? '.

25063 The display of characters having their eighth bit set was less standard. Existing implementations  
 25064 use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed characters with their  
 25065 eighth bit set as the two characters "M-, " followed by the seven bit display as described  
 25066 previously.) The latter probably has the best claim to historical practice because it was used with  
 25067 the **-v** option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

- 25068 No specific display format is required by IEEE Std. 1003.1-200x. Implementations are  
25069 encouraged to conform to historic practice in the absence of any strong reason to diverge.
- 25070 **FUTURE DIRECTIONS**
- 25071 None.
- 25072 **SEE ALSO**
- 25073 *ctags, ed, ex, vi*
- 25074 **CHANGE HISTORY**
- 25075 First released in Issue 4.
- 25076 **Issue 5**
- 25077 FUTURE DIRECTIONS section added.
- 25078 **Issue 6**
- 25079 This utility is now marked as part of the User Portability Utilities option.
- 25080 The obsolescent SYNOPSIS is removed.
- 25081 The utility has been extensively reworked for alignment with the IEEE P1003.2b draft standard:
- 25082
- Changes have been made as result of IEEE PASC Interpretations 1003.2 #37 and #109.
  - The *more* utility should be able to handle underlined and emboldened displays of characters  
25083 that are wider than a single column position.  
25084

25085 **NAME**

25086 mv — move files

25087 **SYNOPSIS**25088 mv [-fi] *source\_file target\_file*25089 mv [-fi] *source\_file... target\_file*25090 **DESCRIPTION**

25091 In the first synopsis form, the *mv* utility shall move the file named by the *source\_file* operand to  
 25092 the *destination* specified by the *target\_file*. This first synopsis form is assumed when the final  
 25093 operand does not name an existing directory and is not a symbolic link referring to an existing  
 25094 directory.

25095 In the second synopsis form, *mv* shall move each file named by a *source\_file* operand to a  
 25096 *destination* file in the existing directory named by the *target\_dir* operand, or referenced if  
 25097 *target\_dir* is a symbolic link referring to an existing directory. The *destination* path for each  
 25098 *source\_file* shall be the concatenation of the target directory, a single slash character, and the last  
 25099 path name component of the *source\_file*. This second form is assumed when the final operand  
 25100 names an existing directory.

25101 If any operand specifies an existing file of a type not specified by the System Interfaces volume  
 25102 of IEEE Std. 1003.1-200x, the behavior is implementation-defined.

25103 For each *source\_file* the following steps shall be taken:

25104 1. If the destination path exists, the *-f* option is not specified, and either of the following  
 25105 conditions is true:

25106 a. The permissions of the destination path do not permit writing and the standard input  
 25107 is a terminal.

25108 b. The *-i* option is specified.

25109 the *mv* utility shall write a prompt to standard error and read a line from standard input. If  
 25110 the response is not affirmative, *mv* shall do nothing more with the current *source\_file* and  
 25111 go on to any remaining *source\_files*.

25112 2. The *mv* utility shall perform actions equivalent to the *rename()* function defined in the  
 25113 System Interfaces volume of IEEE Std. 1003.1-200x, called with the following arguments:

25114 a. The *source\_file* operand is used as the *old* argument.

25115 b. The destination path is used as the *new* argument.

25116 If this succeeds, *mv* shall do nothing more with the current *source\_file* and go on to any  
 25117 remaining *source\_files*. If this fails for any reasons other than those described for the *errno*  
 25118 [EXDEV] in the System Interfaces volume of IEEE Std. 1003.1-200x, *mv* shall write a  
 25119 diagnostic message to standard error, do nothing more with the current *source\_file*, and go  
 25120 on to any remaining *source\_files*.

25121 3. If the destination path exists, and it is a file of type directory and *source\_file* is not a file of  
 25122 type directory, or it is a file not of type directory and *source\_file* is a file of type directory,  
 25123 *mv* shall write a diagnostic message to standard error, do nothing more with the current  
 25124 *source\_file*, and go on to any remaining *source\_files*.

25125 4. If the destination path exists, *mv* shall attempt to remove it. If this fails for any reason, *mv*  
 25126 shall write a diagnostic message to standard error, do nothing more with the current  
 25127 *source\_file*, and go on to any remaining *source\_files*.

25128 5. The file hierarchy rooted in *source\_file* shall be duplicated as a file hierarchy rooted in the  
 25129 *destination* path. If *source\_file* or any of the files below it in the hierarchy are symbolic links,  
 25130 the links themselves shall be duplicated, including their contents, rather than any files to  
 25131 which they refer. The following characteristics of each file in the file hierarchy shall be  
 25132 duplicated:

- 25133 • The time of last data modification and time of last access
- 25134 • The user ID and group ID
- 25135 • The file mode

25136 If the user ID, group ID, or file mode of a regular file cannot be duplicated, the file mode  
 25137 bits S\_ISUID and S\_ISGID shall not be duplicated.

25138 When files are duplicated to another file system, the implementation may require that the  
 25139 process invoking *mv* has read access to each file being duplicated.

25140 If the duplication of the file hierarchy fails for any reason, *mv* shall write a diagnostic  
 25141 message to standard error, do nothing more with the current *source\_file*, and go on to any  
 25142 remaining *source\_files*.

25143 If the duplication of the file characteristics fails for any reason, *mv* shall write a diagnostic  
 25144 message to standard error, but this failure shall not cause *mv* to modify its exit status.

25145 6. The file hierarchy rooted in *source\_file* shall be removed. If this fails for any reason, *mv* shall  
 25146 write a diagnostic message to the standard error, do nothing more with the current  
 25147 *source\_file*, and go on to any remaining *source\_files*.

#### 25148 OPTIONS

25149 The *mv* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 25150 12.2, Utility Syntax Guidelines.

25151 The following options shall be supported:

- 25152 **-f** Do not prompt for confirmation if the destination path exists. Any previous  
 25153 occurrences of the **-i** option is ignored.
- 25154 **-i** Prompt for confirmation if the destination path exists. Any previous occurrences of  
 25155 the **-f** option is ignored.

25156 Specifying more than one of the **-f** or **-i** options shall not be considered an error. The last option  
 25157 specified shall determine the behavior of *mv*.

#### 25158 OPERANDS

25159 The following operands shall be supported:

- 25160 *source\_file* A path name of a file or directory to be moved.
- 25161 *target\_file* A new path name for the file or directory being moved.
- 25162 *target\_dir* A path name of an existing directory into which to move the input files.

#### 25163 STDIN

25164 Used to read an input line in response to each prompt specified in the STDERR section.  
 25165 Otherwise, the standard input shall not be used.

#### 25166 INPUT FILES

25167 The input files specified by each *source\_file* operand can be of any file type.

25168 **ENVIRONMENT VARIABLES**

25169 The following environment variables shall affect the execution of *mv*:

25170 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 25171 If *LANG* is unset or null, the corresponding value from the implementation-  
 25172 defined default locale shall be used. If any of the internationalization variables  
 25173 contains an invalid setting, the utility shall behave as if none of the variables had  
 25174 been defined.

25175 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 25176 internationalization variables.

25177 *LC\_COLLATE*

25178 Determine the locale for the behavior of ranges, equivalence classes and multi-  
 25179 character collating elements used in the extended regular expression defined for  
 25180 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

25181 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 25182 characters (for example, single-byte as opposed to multi-byte characters in  
 25183 arguments and input files), the behavior of character classes used in the extended  
 25184 regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
 25185 category.

25186 *LC\_MESSAGES*

25187 Determine the locale for the processing of affirmative responses that should be  
 25188 used to affect the format and contents of diagnostic messages written to standard  
 25189 error.

25190 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

25191 **ASYNCHRONOUS EVENTS**

25192 Default.

25193 **STDOUT**

25194 Not used.

25195 **STDERR**

25196 Prompts shall be written to the standard error under the conditions specified in the  
 25197 *DESCRIPTION* section. The prompts shall contain the *destination* path name, but their format is  
 25198 otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic messages.

25199 **OUTPUT FILES**

25200 The output files may be of any file type.

25201 **EXTENDED DESCRIPTION**

25202 None.

25203 **EXIT STATUS**

25204 The following exit values shall be returned:

25205 0 All input files were moved successfully.

25206 >0 An error occurred.

25207 **CONSEQUENCES OF ERRORS**

25208 If the copying or removal of *source\_file* is prematurely terminated by a signal or error, *mv* may  
 25209 leave a partial copy of *source\_file* at the source or destination. The *mv* utility shall not modify  
 25210 both *source\_file* and the destination path simultaneously; termination at any point shall leave  
 25211 either *source\_file* or the destination path complete.

## 25212 APPLICATION USAGE

25213 None.

## 25214 EXAMPLES

25215 If the current directory contains only files **a** (of any type defined by the System Interfaces  
25216 volume of IEEE Std. 1003.1-200x), **b** (also of any type), and a directory **c**:

25217 mv a b c

25218 mv c d

25219 results with the original files **a** and **b** residing in the directory **d** in the current directory.

## 25220 RATIONALE

25221 Early proposals diverged from the SVID and BSD historical practice in that they required that  
25222 when the destination path exists, the **-f** option is not specified, and input is not a terminal, *mv*  
25223 fails. This was done for compatibility with *cp*. The current text returns to historical practice. It  
25224 should be noted that this is consistent with the *rename()* function defined in the System  
25225 Interfaces volume of IEEE Std. 1003.1-200x, which does not require write permission on the  
25226 target.

25227 For absolute clarity, paragraph (1), describing the behavior of *mv* when prompting for  
25228 confirmation, should be interpreted in the following manner:

```
25229 if (exists AND (NOT f_option) AND
25230 ((not_writable AND input_is_terminal) OR i_option))
```

25231 The **-i** option exists on BSD systems, giving applications and users a way to avoid accidentally  
25232 unlinking files when moving others. When the standard input is not a terminal, the 4.3 BSD *mv*  
25233 deletes all existing destination paths without prompting, even when **-i** is specified; this is  
25234 inconsistent with the behavior of the 4.3 BSD *cp* utility, which always generates an error when  
25235 the file is unwritable and the standard input is not a terminal. The standard developers decided  
25236 that use of **-i** is a request for interaction, so when the *destination* path exists, the utility takes  
25237 instructions from whatever responds to standard input.

25238 The *rename()* function is able to move directories within the same file system. Some historical  
25239 versions of *mv* have been able to move directories, but not to a different file system. The  
25240 standard developers considered that this was an annoying inconsistency, so this volume of  
25241 IEEE Std. 1003.1-200x requires directories to be able to be moved even across file systems. There  
25242 is no **-R** option to confirm that moving a directory is actually intended, since such an option was  
25243 not required for moving directories in historical practice. Requiring the application to specify it  
25244 sometimes, depending on the destination, seemed just as inconsistent. The semantics of the  
25245 *rename()* function were preserved as much as possible. For example, *mv* is not permitted to  
25246 “rename” files to or from directories, even though they might be empty and removable.

25247 Historic implementations of *mv* did not exit with a non-zero exit status if they were unable to  
25248 duplicate any file characteristics when moving a file across file systems, nor did they write a  
25249 diagnostic message for the user. The former behavior has been preserved to prevent scripts from  
25250 breaking; a diagnostic message is now required, however, so that users are alerted that the file  
25251 characteristics have changed.

25252 The exact format of the interactive prompts is unspecified. Only the general nature of the  
25253 contents of prompts are specified because implementations may desire more descriptive  
25254 prompts than those used on historical implementations. Therefore, an application not using the  
25255 **-f** option or using the **-i** option relies on the system to provide the most suitable dialog directly  
25256 with the user, based on the behavior specified.

25257 When *mv* is dealing with a single file system and *source\_file* is a symbolic link, the link itself is  
25258 moved as a consequence of the dependence on the *rename()* functionality, per the



- 25259           DESCRIPTION. Across file systems, this has to be made explicit.
- 25260 **FUTURE DIRECTIONS**
- 25261           None.
- 25262 **SEE ALSO**
- 25263           *cp, ln*
- 25264 **CHANGE HISTORY**
- 25265           First released in Issue 2.
- 25266 **Issue 4**
- 25267           Aligned with the ISO/IEC 9945-2:1993 standard.
- 25268 **Issue 6**
- 25269           The *mv* utility is changed to describe processing of symbolic links as specified in the
- 25270           IEEE P1003.2b draft standard.

## 25271 NAME

25272 newgrp — change to a new group

## 25273 SYNOPSIS

25274 UP newgrp [-l][group]

25275

## 25276 DESCRIPTION

25277 The *newgrp* utility shall create a new shell execution environment with a new real and effective  
 25278 group identification. Of the attributes listed in Section 2.13 (on page 2273), the new shell  
 25279 execution environment shall retain the working directory, file creation mask, and exported  
 25280 variables from the previous environment (that is, open files, traps, unexported variables, alias  
 25281 definitions, shell functions, and *set* options may be lost). All other aspects of the process  
 25282 environment that are preserved by the *exec* family of functions defined in the System Interfaces  
 25283 volume of IEEE Std. 1003.1-200x shall also be preserved by *newgrp*; whether other aspects are  
 25284 preserved is unspecified.

25285 A failure to assign the new group identifications (for example, for security or password-related  
 25286 reasons) shall not prevent the new shell execution environment from being created.

25287 The *newgrp* utility shall affect the supplemental groups for the process as follows:

- 25288 • On systems where the effective group ID is normally in the supplementary group list (or  
 25289 whenever the old effective group ID actually is in the supplementary group list):
  - 25290 — If the new effective group ID is also in the supplementary group list, *newgrp* shall change  
 25291 the effective group ID.
  - 25292 — If the new effective group ID is not in the supplementary group list, *newgrp* shall add the  
 25293 new effective group ID to the list, if there is room to add it.
- 25294 • On systems where the effective group ID is not normally in the supplementary group list (or  
 25295 whenever the old effective group ID is not in the supplementary group list):
  - 25296 — If the new effective group ID is in the supplementary group list, *newgrp* shall delete it.
  - 25297 — If the old effective group ID is not in the supplementary list, *newgrp* shall add it if there is  
 25298 room.

25299 **Note:** The System Interfaces volume of IEEE Std. 1003.1-200x does not specify whether the  
 25300 effective group ID of a process is included in its supplementary group list.

25301 With no operands, *newgrp* shall change the effective group back to the groups identified in the  
 25302 user's user entry, and shall set the list of supplementary groups to that set in the user's group  
 25303 database entries.

25304 If a password is required for the specified group, and the user is not listed as a member of that  
 25305 group in the group database, the user shall be prompted to enter the correct password for that  
 25306 group. If the user is listed as a member of that group, no password is requested. If no password  
 25307 is required for the specified group, it is implementation-defined whether users not listed as  
 25308 members of that group can change to that group. Whether or not a password is required,  
 25309 implementation-defined system accounting or security mechanisms may impose additional  
 25310 authorization restrictions that may cause *newgrp* to write a diagnostic message and suppress the  
 25311 changing of the group identification.

## 25312 OPTIONS

25313 The *newgrp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 25314 12.2, Utility Syntax Guidelines.

25315 The following option shall be supported:

25316 **-l** (The letter ell.) Change the environment to what would be expected if the user  
25317 actually logged in again.

#### 25318 OPERANDS

25319 The following operand shall be supported:

25320 *group* A group name from the group database or a non-negative numeric group ID.  
25321 Specifies the group ID to which the real and effective group IDs shall be set. If  
25322 *group* is a non-negative numeric string and exists in the group database as a group  
25323 name (see *getgrnam()*), the numeric group ID associated with that group name  
25324 shall be used as the group ID.

#### 25325 STDIN

25326 Not used.

#### 25327 INPUT FILES

25328 The file */dev/tty* shall be used to read a single line of text for password checking, when one is  
25329 required.

#### 25330 ENVIRONMENT VARIABLES

25331 The following environment variables shall affect the execution of *newgrp*:

25332 *LANG* Provide a default value for the internationalization variables that are unset or null.  
25333 If *LANG* is unset or null, the corresponding value from the implementation-  
25334 defined default locale shall be used. If any of the internationalization variables  
25335 contains an invalid setting, the utility shall behave as if none of the variables had  
25336 been defined.

25337 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
25338 internationalization variables.

25339 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
25340 characters (for example, single-byte as opposed to multi-byte characters in  
25341 arguments).

#### 25342 LC\_MESSAGES

25343 Determine the locale that should be used to affect the format and contents of  
25344 diagnostic messages written to standard error.

25345 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 25346 ASYNCHRONOUS EVENTS

25347 Default.

#### 25348 STDOUT

25349 Not used.

#### 25350 STDERR

25351 Used for diagnostic messages and a prompt string for a password, if one is required. Diagnostic  
25352 messages may be written in cases where the exit status is not available. See the EXIT STATUS  
25353 section.

#### 25354 OUTPUT FILES

25355 None.

25356 **EXTENDED DESCRIPTION**

25357 None.

25358 **EXIT STATUS**

25359 If *newgrp* succeeds in creating a new shell execution environment, whether or not the group  
25360 identification was changed successfully, the exit status shall be the exit status of the shell.  
25361 Otherwise, the following exit value shall be returned:

25362 &gt;0 An error occurred.

25363 **CONSEQUENCES OF ERRORS**

25364 The invoking shell may terminate.

25365 **APPLICATION USAGE**

25366 There is no convenient way to enter a password into the Group Database. Use of group  
25367 passwords is not encouraged, because by their very nature they encourage poor security  
25368 practices. Group passwords may disappear in the future.

25369 A common implementation of *newgrp* is that the current shell uses *exec* to overlay itself with  
25370 *newgrp*, which in turn overlays itself with a new shell after changing group. On some systems,  
25371 however, this may not occur and *newgrp* may be invoked as a subprocess.

25372 The *newgrp* command is intended only for use from an interactive terminal. It does not offer a  
25373 useful interface for the support of applications.

25374 The exit status of *newgrp* is generally inapplicable. If *newgrp* is used in a script, in most cases it  
25375 successfully invokes a new shell and the rest of the original shell script is bypassed when the  
25376 new shell exits. Used interactively, *newgrp* displays diagnostic messages to indicate problems.  
25377 But usage such as:

25378 `newgrp foo`25379 `echo $?`

25380 is not useful because the new shell might not have access to any status *newgrp* may have  
25381 generated (and most historical systems do not provide this status). A zero status echoed here  
25382 does not necessarily indicate that the user has changed to the new group successfully. Following  
25383 *newgrp* with the *id* command provides a portable means of determining whether the group  
25384 change was successful or not.

25385 **EXAMPLES**

25386 None.

25387 **RATIONALE**

25388 Most historical implementations use one of the *exec* functions to implement the behavior of  
25389 *newgrp*. Errors detected before the *exec* leave the environment unchanged, while errors detected  
25390 after the *exec* leave the user in a changed environment. While it would be useful to have *newgrp*  
25391 issue a diagnostic message to tell the user that the environment changed, it would be  
25392 inappropriate to require this change to some historical implementations.

25393 The password mechanism is allowed in the group database, but how this would be  
25394 implemented is not specified.

25395 The *newgrp* utility was retained in this volume of IEEE Std. 1003.1-200x, even given the existence  
25396 of the multiple group permissions feature in the System Interfaces volume of  
25397 IEEE Std. 1003.1-200x, for several reasons. First, in some systems, the group ownership of a  
25398 newly created file is determined by the group of the directory in which the file is created, as  
25399 allowed by the System Interfaces volume of IEEE Std. 1003.1-200x; on other systems, the group  
25400 ownership of a newly created file is determined by the effective group ID. On systems of the  
25401 latter type, *newgrp* allows files to be created with a specific group ownership. Finally, many

25402 systems use the real group ID in accounting, and on such systems, *newgrp* allows the accounting  
25403 identity of the user to be changed.

25404 **FUTURE DIRECTIONS**

25405 None.

25406 **SEE ALSO**

25407 *sh*, the System Interfaces volume of IEEE Std. 1003.1-200x, *exec*

25408 **CHANGE HISTORY**

25409 First released in Issue 2.

25410 **Issue 4**

25411 Aligned with the ISO/IEC 9945-2:1993 standard.

25412 The *newgrp* utility is now mandatory; it is optional in Issue 3.

25413 **Issue 6**

25414 This utility is now marked as part of the User Portability Utilities option.

25415 The obsolescent SYNOPSIS is removed.

25416 The text describing supplemental groups is no longer conditional on {NGROUPS\_MAX} being  
25417 greater than 1. This is because {NGROUPS\_MAX} now has a minimum value of 8. This is a FIPS  
25418 requirement.

25419 **NAME**

25420 nice — invoke a utility with an altered nice value

25421 **SYNOPSIS**25422 UP nice [-n *increment*] *utility* [*argument...*]

25423

25424 **DESCRIPTION**

25425 The *nice* utility shall invoke a utility, requesting that it be run with a different nice value (see the  
 25426 Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.241, Nice Value). With no options  
 25427 and only if the user has appropriate privileges, the executed utility shall be run with a nice value  
 25428 that is some implementation-defined quantity less than or equal to the nice value of the current  
 25429 process. If the user lacks appropriate privileges to affect the nice value in the requested manner,  
 25430 the *nice* utility shall not affect the nice value; in this case, a warning message may be written to  
 25431 standard error, but this shall not prevent the invocation of *utility* or affect the exit status.

25432 **OPTIONS**

25433 The *nice* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 25434 12.2, Utility Syntax Guidelines.

25435 The following option is supported:

25436 **-n *increment*** Specify how the nice value of the executed utility shall be adjusted. The *increment*  
 25437 option-argument shall be a positive or negative decimal integer that shall be used  
 25438 to modify the nice value of the executed utility in an implementation-defined  
 25439 manner.

25440 Positive *increment* values shall cause a lower or unchanged nice value. Negative  
 25441 *increment* values may require appropriate privileges and shall cause a higher or  
 25442 unchanged nice value.

25443 The nice value shall be bounded in an implementation-defined manner. If the  
 25444 requested *increment* would raise or lower the nice value of the executed utility  
 25445 beyond implementation-defined limits, then the limit whose value was exceeded  
 25446 shall be used.

25447 **OPERANDS**

25448 The following operands shall be supported:

25449 ***utility*** The name of a utility that is to be invoked. If the *utility* operand names any of the  
 25450 special built-in utilities in Section 2.15 (on page 2276), the results are undefined.

25451 ***argument*** Any string to be supplied as an argument when invoking the utility named by the  
 25452 *utility* operand.

25453 **STDIN**

25454 Not used.

25455 **INPUT FILES**

25456 None.

25457 **ENVIRONMENT VARIABLES**25458 The following environment variables shall affect the execution of *nice*:

25459 ***LANG*** Provide a default value for the internationalization variables that are unset or null.  
 25460 If *LANG* is unset or null, the corresponding value from the implementation-  
 25461 defined default locale shall be used. If any of the internationalization variables  
 25462 contains an invalid setting, the utility shall behave as if none of the variables had  
 25463 been defined.

- 25464 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
25465 internationalization variables.
- 25466 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
25467 characters (for example, single-byte as opposed to multi-byte characters in  
25468 arguments).
- 25469 *LC\_MESSAGES*  
25470 Determine the locale that should be used to affect the format and contents of  
25471 diagnostic messages written to standard error.
- 25472 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 25473 *PATH* Determine the search path used to locate the utility to be invoked. See the Base  
25474 Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.
- 25475 **ASYNCHRONOUS EVENTS**
- 25476 Default.
- 25477 **STDOUT**
- 25478 Not used.
- 25479 **STDERR**
- 25480 Used only for diagnostic messages.
- 25481 **OUTPUT FILES**
- 25482 None.
- 25483 **EXTENDED DESCRIPTION**
- 25484 None.
- 25485 **EXIT STATUS**
- 25486 If the *utility* utility is invoked, the exit status of *nice* shall be the exit status of *utility*; otherwise,  
25487 the *nice* utility shall exit with one of the following values:
- 25488 1-125 An error occurred in the *nice* utility.
- 25489 126 The utility specified by *utility* was found but could not be invoked.
- 25490 127 The utility specified by *utility* could not be found.
- 25491 **CONSEQUENCES OF ERRORS**
- 25492 Default.
- 25493 **APPLICATION USAGE**
- 25494 The only guaranteed portable uses of this utility are:
- 25495 *nice utility*  
25496 Run *utility* with the default lower nice value.
- 25497 *nice -n <positive integer> utility*  
25498 Run *utility* with a lower nice value.
- 25499 On some systems they have no discernible effect on the invoked utility and on some others they  
25500 are exactly equivalent.
- 25501 Historical systems have frequently supported the *<positive integer>* up to 20. Since there is no  
25502 error penalty associated with guessing a number that is too high, users without access to the  
25503 system conformance document (to see what limits are actually in place) could use the historical  
25504 1 to 20 range or attempt to use very large numbers if the job should be truly low priority.
- 25505 The nice value value of a process can be displayed using the command:

25506 ps -o nice

25507 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
25508 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
25509 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
25510 used for other meanings; most utilities use small values for “normal error conditions” and the  
25511 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
25512 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
25513 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
25514 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
25515 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
25516 any other reason.

#### 25517 EXAMPLES

25518 None.

#### 25519 RATIONALE

25520 Due to the text about the limits of the nice value being implementation-defined, *nice* is not  
25521 actually required to change the nice value of the executed command; the limits could be zero  
25522 differences from the system default, although the implementor is required to document this fact  
25523 in the conformance document.

25524 The 4.3 BSD version of *nice* does not check if *increment* is a valid decimal integer. The command  
25525 *nice -x utility*, for example, would be treated the same as the command *nice --1 utility*. If the  
25526 user does not have appropriate privileges, this results in a “permission denied” error. This is  
25527 considered a bug.

25528 When a user without appropriate privileges gives a negative *increment*, System V treats it like  
25529 the command *nice -0 utility*, while 4.3 BSD writes a “permission denied” message and does not  
25530 run the utility. Neither was considered clearly superior, so the behavior was left unspecified.

25531 The C shell has a built-in version of *nice* that has a different interface from the one described in  
25532 this volume of IEEE Std. 1003.1-200x.

25533 The term “utility” is used, rather than “command”, to highlight the fact that shell compound  
25534 commands, pipelines, and so on, cannot be used. Special built-ins also cannot be used.  
25535 However, “utility” includes user application programs and shell scripts, not just utilities defined  
25536 in this volume of IEEE Std. 1003.1-200x.

25537 Historical implementations of *nice* provide a nice value range of 40 or 41 discrete steps, with the  
25538 default nice value being the midpoint of that range. By default, they lower the nice value of the  
25539 executed utility by 10.

25540 Some historical documentation states that the *increment* value must be within a fixed range. This  
25541 is misleading; the valid *increment* values on any invocation are determined by the current  
25542 process nice value, which is not always the default.

25543 The definition of nice value is not intended to suggest that all processes in a system have  
25544 priorities that are comparable. Scheduling policy extensions such as the realtime priorities in  
25545 POSIX.4 make the notion of a single underlying priority for all scheduling policies problematic.  
25546 Some systems may implement the *nice*-related features to affect all processes on the system,  
25547 others to affect just the general time-sharing activities implied by this volume of  
25548 IEEE Std. 1003.1-200x, and others may have no effect at all. Because of the use of  
25549 “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are  
25550 possible.



25551 **FUTURE DIRECTIONS**

25552           None.

25553 **SEE ALSO**25554           *renice*25555 **CHANGE HISTORY**

25556           First released in Issue 4.

25557 **Issue 6**

25558           This utility is now marked as part of the User Portability Utilities option.

25559           The obsolescent SYNOPSIS is removed.

## 25560 NAME

25561 nl — line numbering filter

## 25562 SYNOPSIS

```
25563 xSI nl [-p][-b type][-d delim][-f type][-h type][-i incr][-l num][-n format]
25564 [-s sep][-v startnum][-w width][file]
25565
```

## 25566 DESCRIPTION

25567 The *nl* utility shall read lines from the named *file* or the standard input if no *file* is named and  
 25568 shall reproduce the lines to standard output. Lines shall be numbered on the left. Additional  
 25569 functionality may be provided in accordance with the command options in effect.

25570 The *nl* utility views the text it reads in terms of logical pages. Line numbering is reset at the start  
 25571 of each logical page. A logical page consists of a header, a body, and a footer section. Empty  
 25572 sections are valid. Different line numbering options are independently available for header,  
 25573 body, and footer (for example, no numbering of header and footer lines while numbering blank  
 25574 lines only in the body).

25575 The starts of logical page sections are signaled by input lines containing nothing but the  
 25576 following delimiter characters:

25577

| Line     | Start of |
|----------|----------|
| \: \: \: | Header   |
| \: \:    | Body     |
| \:       | Footer   |

25578

25579

25580

25581 Unless otherwise specified, *nl* assumes the text being read is in a single logical page body.

## 25582 OPTIONS

25583 The *nl* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 25584 Utility Syntax Guidelines. Only one file can be named.

25585 The following options shall be supported:

25586 **-b type** Specify which logical page body lines shall be numbered. Recognized *types* and  
 25587 their meaning are:

25588 **a** Number all lines.

25589 **t** Number only non-empty lines.

25590 **n** No line numbering.

25591 **pstring** Number only lines that contain the basic regular expression specified in  
 25592 *string*.

25593 The default *type* for logical page body is **t** (text lines numbered).

25594 **-d delim** Specify the delimiter characters that indicate the start of a logical page section.  
 25595 These can be changed from the default characters "\:" to two user-specified  
 25596 characters. If only one character is entered, the second character remains the  
 25597 default character ' : '.

25598 **-f type** Specify the same as **b type** except for footer. The default for logical page footer is **n**  
 25599 (no lines numbered).

25600 **-h type** Specify the same as **b type** except for header. The default *type* for logical page  
 25601 header is **n** (no lines numbered).

- 25602        **-i incr**        Specify the increment value used to number logical page lines. The default is 1.
- 25603        **-l num**        Specify the number of blank lines to be considered as one. For example, **-l 2** results  
25604        in only the second adjacent blank line being numbered (if the appropriate **-h a**,  
25605        **-b a**, or **-f a** option is set). The default is 1.
- 25606        **-n format**       Specify the line numbering format. Recognized values are: **ln**, left justified, leading  
25607        zeros suppressed; **rn**, right justified, leading zeros suppressed; **rz**, right justified,  
25608        leading zeros kept. The default *format* is **rn** (right justified).
- 25609        **-p**                Specify that numbering should not be restarted at logical page delimiters.
- 25610        **-s sep**           Specify the characters used in separating the line number and the corresponding  
25611        text line. The default *sep* is a <tab>.
- 25612        **-v startnum**     Specify the initial value used to number logical page lines. The default is 1.
- 25613        **-w width**        Specify the number of characters to be used for the line number. The default *width*  
25614        is 6.

#### 25615 OPERANDS

25616        The following operand shall be supported:

- 25617        *file*            A path name of a text file to be line-numbered.

#### 25618 STDIN

25619        The standard input is a text file that is used if no *file* operand is given.

#### 25620 INPUT FILES

25621        The input file named by the *file* operand is a text file.

#### 25622 ENVIRONMENT VARIABLES

25623        The following environment variables shall affect the execution of *nl*:

- 25624        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
25625        If *LANG* is unset or null, the corresponding value from the implementation-  
25626        defined default locale shall be used. If any of the internationalization variables  
25627        contains an invalid setting, the utility shall behave as if none of the variables had  
25628        been defined.
- 25629        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
25630        internationalization variables.
- 25631        *LC\_COLLATE*     Determine the locale for the behavior of ranges, equivalence classes and multi-  
25632        character collating elements within regular expressions.
- 25633        *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
25634        characters (for example, single-byte as opposed to multi-byte characters in  
25635        arguments and input files), the behavior of character classes within regular  
25636        expressions, and for deciding which characters are in character class **graph** (for the  
25637        **-b t**, **-f t**, and **-h t** options).  
25638        **-b t**, **-f t**, and **-h t** options).
- 25639        *LC\_MESSAGES*   Determine the locale that should be used to affect the format and contents of  
25640        diagnostic messages written to standard error.  
25641        *NLSPATH*       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

25643 **ASYNCHRONOUS EVENTS**

25644 Default.

25645 **STDOUT**

25646 The standard output shall be a text file in the following format:

25647 "%s%s%s", &lt;line number&gt;, &lt;separator&gt;, &lt;input line&gt;

25648 where &lt;line number&gt; is one of the following numeric formats:

25649 %6d When the **rn** format is used (the default; see **-n**).25650 %06d When the **rz** format is used.25651 %-6d When the **ln** format is used.25652 <empty> When line numbers are suppressed for a portion of the page; the <separator> is also  
25653 suppressed.25654 In the preceding list, the number 6 is the default width; the **-w** option can change this value.25655 **STDERR**

25656 Used only for diagnostic messages.

25657 **OUTPUT FILES**

25658 None.

25659 **EXTENDED DESCRIPTION**

25660 None.

25661 **EXIT STATUS**

25662 The following exit values shall be returned:

25663 0 Successful completion.

25664 &gt;0 An error occurred.

25665 **CONSEQUENCES OF ERRORS**

25666 Default.

25667 **APPLICATION USAGE**25668 In using the **-d delim** option, care should be taken to escape characters that have special meaning  
25669 to the command interpreter.25670 **EXAMPLES**

25671 The command:

25672 nl -v 10 -i 10 -d \!+ file1

25673 numbers *file1* starting at line number 10 with an increment of 10. The logical page delimiter is  
25674 " !+". Note that the ' ! ' has to be escaped when using *csh* as a command interpreter because of  
25675 its history substitution syntax. For *ksh* and *sh* the escape is not necessary, but does not do any  
25676 harm.25677 **RATIONALE**

25678 None.

25679 **FUTURE DIRECTIONS**

25680 None.

25681 **SEE ALSO**25682 *pr*25683 **CHANGE HISTORY**

25684 First released in Issue 2.

25685 **Issue 4**

25686 Format reorganized.

25687 Utility Syntax Guideline support mandated.

25688 Internationalized environment variable support mandated.

25689 **Issue 5**25690 The option `[-f type]` is added to the SYNOPSIS. The option descriptions are presented in  
25691 alphabetic order. The description of `-bt` is changed to “Number only non-empty lines”.25692 **Issue 6**25693 The obsolescent behavior allowing the options to be intermingled with the optional *file* operand  
25694 is removed.

## 25695 NAME

25696 nm — write the name list of an object file (DEVELOPMENT)

## 25697 SYNOPSIS

25698 UP SD XSI nm [-APv][-efox][ -g | -u][-t *format*] *file...*

25699

## 25700 DESCRIPTION

25701 This utility shall be provided on systems that support both the User Portability Utilities option  
25702 and the Software Development Utilities option. On other systems it is optional. Certain options  
25703 are only available on XSI-conformant systems.

25704 The *nm* utility shall display symbolic information appearing in the object file, executable file or  
25705 object-file library named by *file*. If no symbolic information is available for a valid input file, the  
25706 *nm* utility shall report that fact, but not consider it an error condition.

25707 XSI The default base used when numeric values are written is unspecified. On XSI-conformant  
25708 systems, it shall be decimal.

## 25709 OPTIONS

25710 The *nm* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
25711 12.2, Utility Syntax Guidelines.

25712 The following options shall be supported:

25713 **-A** Write the full path name or library name of an object on each line.

25714 XSI **-e** Write only external (global) and static symbol information.

25715 XSI **-f** Produce full output. Write redundant symbols (**.text**, **.data**, and **.bss**), normally  
25716 suppressed.

25717 **-g** Write only external (global) symbol information.

25718 XSI **-o** Write numeric values in octal (equivalent to **-t o**).

25719 **-P** Write information in a portable output format, as specified in the STDOUT section.

25720 **-t format** Write each numeric value in the specified format. The format shall be dependent  
25721 on the single character used as the *format* option-argument:

25722 XSI d The offset is written in decimal (default).

25723 o The offset is written in octal.

25724 x The offset is written in hexadecimal.

25725 **-u** Write only undefined symbols.

25726 **-v** Sort output by value instead of alphabetically.

25727 XSI **-x** Write numeric values in hexadecimal (equivalent to **-t x**).

## 25728 OPERANDS

25729 The following operand shall be supported:

25730 *file* A path name of an object file, executable file, or object-file library.

## 25731 STDIN

25732 See the INPUT FILES section.

25733 **INPUT FILES**

25734 The input file shall be an object file, an object-file library whose format is the same as those  
 25735 produced by the *ar* utility for link editing, or an executable file. The *nm* utility may accept  
 25736 additional implementation-defined object library formats for the input file.

25737 **ENVIRONMENT VARIABLES**

25738 The following environment variables shall affect the execution of *nm*:

25739 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 25740 If *LANG* is unset or null, the corresponding value from the implementation-  
 25741 defined default locale shall be used. If any of the internationalization variables  
 25742 contains an invalid setting, the utility shall behave as if none of the variables had  
 25743 been defined.

25744 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 25745 internationalization variables.

25746 *LC\_COLLATE*  
 25747 Determine the locale for character collation information for the symbol-name and  
 25748 symbol-value collation sequences.

25749 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 25750 characters (for example, single-byte as opposed to multi-byte characters in  
 25751 arguments).

25752 *LC\_MESSAGES*  
 25753 Determine the locale that should be used to affect the format and contents of  
 25754 diagnostic messages written to standard error.

25755 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

25756 **ASYNCHRONOUS EVENTS**

25757 Default.

25758 **STDOUT**

25759 If symbolic information is present in the input files, then for each file or for each member of an  
 25760 archive, the *nm* utility shall write the following information to standard output. By default, the  
 25761 format is unspecified, but the output shall be sorted alphabetically by symbol name:

- 25762 • Library or object name, if *-A* is specified
- 25763 • Symbol name
- 25764 • Symbol type, which shall either be one of the following single characters or an  
 25765 implementation-defined type represented by a single character:
  - 25766 A Global absolute symbol.
  - 25767 a Local absolute symbol.
  - 25768 B Global “bss” (that is, uninitialized data space) symbol.
  - 25769 b Local bss symbol.
  - 25770 D Global data symbol.
  - 25771 d Local data symbol.
  - 25772 T Global text symbol.
  - 25773 t Local text symbol.

25774           U    Undefined symbol.

25775           • Value of the symbol

25776           • The size associated with the symbol, if applicable

25777           This information may be supplemented by additional information specific to the  
25778           implementation.

25779           If the **-P** option is specified, the previous information shall be displayed using the following  
25780           portable format. The three versions differ depending on whether **-t d**, **-t o**, or **-t x** was specified,  
25781           respectively:

25782           "%s%s %s %d %d\n", <library/object name>, <name>, <type>,  
25783           <value>, <size>

25784           "%s%s %s %o %o\n", <library/object name>, <name>, <type>,  
25785           <value>, <size>

25786           "%s%s %s %x %x\n", <library/object name>, <name>, <type>,  
25787           <value>, <size>

25788           where

25789           <library/object name> shall be formatted as follows:

25790           • If **-A** is not specified, <library/object name> shall be an empty string.

25791           • If **-A** is specified and the corresponding *file* operand does not name a library:

25792            "%s: ", <file>

25793           • If **-A** is specified and the corresponding *file* operand names a library. In this case, <object file>  
25794           shall name the object file in the library containing the symbol being described:

25795           "%s[%s]: ", <file>, <object file>

25796           If **-A** is not specified, then if more than one *file* operand is specified or if only one *file* operand is  
25797           specified and it names a library, *nm* shall write a line identifying the object containing the  
25798           following symbols before the lines containing those symbols, in the form:

25799           • If the corresponding *file* operand does not name a library:

25800            "%s:\n", <file>

25801           • If the corresponding *file* operand names a library; in this case, <object file> shall be the name  
25802           of the file in the library containing the following symbols:

25803            "%s[%s]:\n", <file>, <object file>

25804           If **-P** is specified, but **-t** is not, the format shall be as if **-t x** had been specified.

25805   **STDERR**

25806           Used only for diagnostic messages.

25807   **OUTPUT FILES**

25808           None.

25809   **EXTENDED DESCRIPTION**

25810           None.



25811 **EXIT STATUS**

25812           The following exit values shall be returned:

25813           0   Successful completion.

25814           >0  An error occurred.

25815 **CONSEQUENCES OF ERRORS**

25816           Default.

25817 **APPLICATION USAGE**

25818           Mechanisms for dynamic linking make this utility less meaningful when applied to an executable file because a dynamically linked executable may omit numerous library routines that would be found in a statically linked executable.

25821 **EXAMPLES**

25822           None.

25823 **RATIONALE**

25824           Historical implementations of *nm* have used different bases for numeric output and supplied different default types of symbols that were reported. The *-t format* option, similar to that used in *od* and *strings*, can be used to specify the numeric base; *-g* and *-u* can be used to restrict the amount of output or the types of symbols included in the output.

25828           The option list was significantly reduced from that provided by historical implementations.

25829           The *nm* description is a subset of both the System V and BSD *nm* utilities with no specified default output.

25831           It was recognized that mechanisms for dynamic linking make this utility less meaningful when applied to an executable file (because a dynamically linked executable file may omit numerous library routines that would be found in a statically linked executable file), but the value of *nm* during software development was judged to outweigh other limitations.

25835           The compromise of using *-t format versus* using *-d*, *-o*, and other similar options was necessary because of differences in the meaning of *-o* between implementations. The *-o* option from BSD has been provided here as *-A* to avoid confusion with the *-o* from System V (which has been provided here as *-t* and as *-o* on XSI-conformant systems).

25839           The default output format of *nm* is not specified because of differences in historical implementations. The *-P* option was added to allow some type of portable output format. After a comparison of the different formats used in SunOS, BSD, SVR3, and SVR4, it was decided to create one that did not match the current format of any of these four systems. The format devised is easy to parse by humans, easy to parse in shell scripts, and does not need to vary depending on locale (because no English descriptions are included). All of the systems currently have the information available to use this format.

25846           The format given in *nm* STDOUT uses spaces between the fields, which may be any number of <blank>s required to align the columns. The single-character types were selected to match historical practice, and the requirement that implementation additions also be single characters made parsing the information easier for shell scripts.

25850 **FUTURE DIRECTIONS**

25851           None.

25852 **SEE ALSO**

25853           *ar*, *c99*

25854 **CHANGE HISTORY**

25855 First released in Issue 2.

25856 **Issue 4**

25857 Aligned with the ISO/IEC 9945-2:1993 standard.

25858 **Issue 6**

25859 This utility is now marked as supported when both the User Portability Utilities option and the  
25860 Software Development Utilities option are supported.

25861 **NAME**

25862           nohup — invoke a utility immune to hangups

25863 **SYNOPSIS**25864           nohup *utility* [*argument...*]25865 **DESCRIPTION**

25866           The *nohup* utility shall invoke the utility named by the *utility* operand with arguments supplied  
 25867 as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal shall be  
 25868 set to be ignored.

25869           If the standard output is a terminal, all output written by the named *utility* to its standard output  
 25870 shall be appended to the end of the file **nohup.out** in the current directory. If **nohup.out** cannot  
 25871 be created or opened for appending, the output shall be appended to the end of the file  
 25872 **nohup.out** in the directory specified by the *HOME* environment variable. If neither file can be  
 25873 created or opened for appending, *utility* shall not be invoked. If a file is created, the file's  
 25874 permission bits shall be set to S\_IRUSR | S\_IWUSR.

25875           If the standard error is a terminal, all output written by the named *utility* to its standard error  
 25876 shall be redirected to the same file descriptor as the standard output.

25877 **OPTIONS**

25878           None.

25879 **OPERANDS**

25880           The following operands shall be supported:

25881           *utility*       The name of a utility that is to be invoked. If the *utility* operand names any of the  
 25882 special built-in utilities in Section 2.15 (on page 2276), the results are undefined.

25883           *argument*     Any string to be supplied as an argument when invoking the utility named by the  
 25884 *utility* operand.

25885 **STDIN**

25886           Not used.

25887 **INPUT FILES**

25888           None.

25889 **ENVIRONMENT VARIABLES**25890           The following environment variables shall affect the execution of *nohup*:

25891           *HOME*         Determine the path name of the user's home directory: if the output file **nohup.out**  
 25892 cannot be created in the current directory, the *nohup* utility shall use the directory  
 25893 named by *HOME* to create the file.

25894           *LANG*         Provide a default value for the internationalization variables that are unset or null.  
 25895 If *LANG* is unset or null, the corresponding value from the implementation-  
 25896 defined default locale shall be used. If any of the internationalization variables  
 25897 contains an invalid setting, the utility behave as if none of the variables had been  
 25898 defined.

25899           *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 25900 internationalization variables.

25901           *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
 25902 characters (for example, single-byte as opposed to multi-byte characters in  
 25903 arguments).

- 25904 *LC\_MESSAGES*  
25905 Determine the locale that should be used to affect the format and contents of  
25906 diagnostic messages written to standard error.
- 25907 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 25908 *PATH* Determine the search path that is used to locate the utility to be invoked. See the  
25909 Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment  
25910 Variables.
- 25911 **ASYNCHRONOUS EVENTS**  
25912 The *nohup* utility shall take the standard action for all signals except that SIGHUP shall be  
25913 ignored.
- 25914 **STDOUT**  
25915 If the standard output is not a terminal, the standard output of *nohup* shall be the standard  
25916 output generated by the execution of the *utility* specified by the operands. Otherwise, nothing  
25917 shall be written to the standard output.
- 25918 **STDERR**  
25919 If the standard output is a terminal, a message shall be written to the standard error, indicating  
25920 the name of the file to which the output is being appended. The name of the file shall be either  
25921 **nohup.out** or **\$HOME/nohup.out**.
- 25922 **OUTPUT FILES**  
25923 If the standard output is a terminal, all output written by the named *utility* to the standard  
25924 output and standard error is appended to the file **nohup.out**, which is created if it does not  
25925 already exist.
- 25926 **EXTENDED DESCRIPTION**  
25927 None.
- 25928 **EXIT STATUS**  
25929 The following exit values shall be returned:  
25930 126 The utility specified by *utility* was found but could not be invoked.  
25931 127 An error occurred in the *nohup* utility or the utility specified by *utility* could not be  
25932 found.  
25933 Otherwise, the exit status of *nohup* shall be that of the utility specified by the *utility* operand.
- 25934 **CONSEQUENCES OF ERRORS**  
25935 Default.
- 25936 **APPLICATION USAGE**  
25937 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
25938 an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
25939 utility exited with an error indication”. The value 127 was chosen because it is not commonly  
25940 used for other meanings; most utilities use small values for “normal error conditions” and the  
25941 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
25942 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
25943 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
25944 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
25945 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
25946 any other reason.

25947 **EXAMPLES**

25948 It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done by  
25949 placing pipelines and command lists in a single file; this file can then be invoked as a utility, and  
25950 the *nohup* applies to everything in the file.

25951 Alternatively, the following command can be used to apply *nohup* to a complex command:

```
25952 nohup sh -c 'complex-command-line'
```

25953 **RATIONALE**

25954 The 4.3 BSD version ignores SIGTERM and SIGHUP, and if *./nohup.out* cannot be used, it fails  
25955 instead of trying to use *\$HOME/nohup.out*.

25956 The *cs* utility has a built-in version of *nohup* that acts differently from the POSIX Shell and  
25957 Utilities *nohup*.

25958 The term *utility* is used, rather than *command*, to highlight the fact that shell compound  
25959 commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility*  
25960 includes user application programs and shell scripts, not just the standard utilities.

25961 Historical versions of the *nohup* utility use default file creation semantics. Some more recent  
25962 versions use the permissions specified here as an added security precaution.

25963 Some historical implementations ignore SIGQUIT in addition to SIGHUP; others ignore  
25964 SIGTERM. An early proposal allowed, but did not require, SIGQUIT to be ignored. Several  
25965 reviewers objected that *nohup* should only modify the handling of SIGHUP as required by this  
25966 volume of IEEE Std. 1003.1-200x.

25967 **FUTURE DIRECTIONS**

25968 None.

25969 **SEE ALSO**

25970 *sh*, the System Interfaces volume of IEEE Std. 1003.1-200x, *signal()*

25971 **CHANGE HISTORY**

25972 First released in Issue 2.

25973 **Issue 4**

25974 Aligned with the ISO/IEC 9945-2:1993 standard.

## 25975 NAME

25976 od — dump files in various formats

## 25977 SYNOPSIS

25978 od [-v][-A *address\_base*][-j *skip*][-N *count*][-t *type\_string*].  
 25979 [*file*...]

25980 XSI od [-bcdosx][*file*] [[+]offset[.][b]]

25981

## 25982 DESCRIPTION

25983 The *od* utility shall write the contents of its input files to standard output in a user-specified  
 25984 format.

## 25985 OPTIONS

25986 The *od* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 25987 XSI Utility Syntax Guidelines, except that the order of presentation of the **-t** options and the  
 25988 **-bcdosx** options is significant.

25989 The following options shall be supported:

25990 **-A** *address\_base*

25991 Specify the input offset base. See the EXTENDED DESCRIPTION section. The  
 25992 application shall ensure that the *address\_base* option-argument is a character. The  
 25993 characters 'd', 'o', and 'x' specify that the offset base shall be written in  
 25994 decimal, octal, or hexadecimal, respectively. The character 'n' specifies that the  
 25995 offset shall not be written.

25996 XSI **-b** Interpret bytes in octal. This is equivalent to **-t o1**.

25997 XSI **-c** Interpret bytes as characters specified by the current setting of the *LC\_CTYPE*  
 25998 category. Certain non-graphic characters appear as C escapes: "NUL=\0",  
 25999 "BS=\b", "FF=\f", "NL=\n", "CR=\r", "HT=\t"; others appear as 3-digit octal  
 26000 numbers.

26001 XSI **-d** Interpret *words* (two-byte units) in unsigned decimal. This is equivalent to **-t u2**.

26002 **-j** *skip* Jump over *skip* bytes from the beginning of the input. The *od* utility shall read or  
 26003 seek past the first *skip* bytes in the concatenated input files. If the combined input  
 26004 is not at least *skip* bytes long, the *od* utility shall write a diagnostic message to  
 26005 standard error and exit with a non-zero exit status.

26006 By default, the *skip* option-argument shall be interpreted as a decimal number.  
 26007 With a leading "0x" or "0X", the offset shall be interpreted as a hexadecimal  
 26008 number; otherwise, with a leading '0', the offset shall be interpreted as an octal  
 26009 number. Appending the character 'b', 'k', or 'm' to offset shall cause it to be  
 26010 MAN interpreted as a multiple of 512, 1024, or 1048576 bytes, respectively. If the *skip*  
 26011 number is hexadecimal, any appended 'b' shall be considered to be the final  
 26012 hexadecimal digit.

26013 **-N** *count* Format no more than *count* bytes of input. By default, *count* shall be interpreted as  
 26014 a decimal number. With a leading "0x" or "0X", *count* shall be interpreted as a  
 26015 hexadecimal number; otherwise, with a leading '0', it shall be interpreted as an  
 26016 octal number. If *count* bytes of input (after successfully skipping, if **-j** *skip*  
 26017 is specified) are not available, it shall not be considered an error; the *od* utility shall  
 26018 format the input that is available.

26019 XSI **-o** Interpret *words* (two-byte units) in octal. This is equivalent to **-t o2**.

- 26020 XSI **-s** Interpret *words* (two-byte units) in signed decimal. This is equivalent to **-t d2**.
- 26021 **-t *type\_string***
- 26022 Specify one or more output types. See the EXTENDED DESCRIPTION section. The
- 26023 application shall ensure that the *type\_string* option-argument is a string specifying
- 26024 the types to be used when writing the input data. The string shall consist of the
- 26025 type specification characters **a**, **c**, **d**, **f**, **o**, **u**, and **x**, specifying named character,
- 26026 character, signed decimal, floating point, octal, unsigned decimal, and
- 26027 hexadecimal, respectively. The type specification characters **d**, **f**, **o**, **u**, and **x** can be
- 26028 followed by an optional unsigned decimal integer that specifies the number of
- 26029 bytes to be transformed by each instance of the output type. The type specification
- 26030 character **f** can be followed by an optional **F**, **D**, or **L** indicating that the conversion
- 26031 should be applied to an item of type **float**, **double**, or **long double**, respectively.
- 26032 The type specification characters **d**, **o**, **u** and **x** can be followed by an optional **C**, **S**,
- 26033 **I**, or **L** indicating that the conversion should be applied to an item of type **char**,
- 26034 **short**, **int**, or **long**, respectively. Multiple types can be concatenated within the
- 26035 same *type\_string* and multiple **-t** options can be specified. Output lines shall be
- 26036 written for each type specified in the order in which the type specification
- 26037 characters are specified.
- 26038 **-v** Write all input data. Without the **-v** option, any number of groups of output lines,
- 26039 which would be identical to the immediately preceding group of output lines
- 26040 (except for the byte offsets), shall be replaced with a line containing only an
- 26041 asterisk (**' \* '**).
- 26042 XSI **-x** Interpret *words* (two-byte units) in hexadecimal. This is equivalent to **-t x2**.
- 26043 XSI Multiple types can be specified by using multiple **-bcdostx** options. Output lines are written for
- 26044 each type specified in the order in which the types are specified.
- 26045 **OPERANDS**
- 26046 The following operands shall be supported:
- 26047 **file** A path name of a file to be read. If no *file* operands are specified, the standard
- 26048 input shall be used. If the first character of *file* is a plus sign (**' + '**) or the first
- 26049 character of the first *file* operand is numeric, no more than two operands are given,
- 26050 XSI and none of the **-A**, **-j**, **-N**, or **-t** options is specified, the results are unspecified.
- 26051 On XSI-conformant systems, the operand shall be assumed to be an *offset*.
- 26052 XSI **[+]*offset*[.][*b*]**
- 26053 The *offset* operand specifies the offset in the file where dumping is to commence.
- 26054 This operand is normally interpreted as octal bytes. If **' . '** is appended, the offset
- 26055 shall be interpreted in decimal. If **' b '** is appended, the offset shall be interpreted
- 26056 in units of 512 bytes. If the *file* argument is omitted, and none of the **-A**, **-j**, **-N**, or
- 26057 **-t** options is specified, the application shall ensure that the *offset* argument is
- 26058 preceded by **' + '**.
- 26059 **STDIN**
- 26060 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES
- 26061 section.
- 26062 **INPUT FILES**
- 26063 The input files can be any file type.

26064 **ENVIRONMENT VARIABLES**

26065 The following environment variables shall affect the execution of *od*:

26066 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 26067 If *LANG* is unset or null, the corresponding value from the implementation-  
 26068 defined default locale shall be used. If any of the internationalization variables  
 26069 contains an invalid setting, the utility shall behave as if none of the variables had  
 26070 been defined.

26071 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 26072 internationalization variables.

26073 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 26074 characters (for example, single-byte as opposed to multi-byte characters in  
 26075 arguments and input files).

26076 *LC\_MESSAGES*  
 26077 Determine the locale that should be used to affect the format and contents of  
 26078 diagnostic messages written to standard error.

26079 *LC\_NUMERIC*  
 26080 Determine the locale for selecting the radix character used when writing floating-  
 26081 point formatted output.

26082 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

26083 **ASYNCHRONOUS EVENTS**

26084 Default.

26085 **STDOUT**

26086 See the EXTENDED DESCRIPTION section.

26087 **STDERR**

26088 Used only for diagnostic messages.

26089 **OUTPUT FILES**

26090 None.

26091 **EXTENDED DESCRIPTION**

26092 The *od* utility shall copy sequentially each input file to standard output, transforming the input  
 26093 XSI data according to the output types specified by the **-t** options or the **-bcdosx** options. If no  
 26094 output type is specified, the default output shall be as if **-t oS** had been specified.

26095 The number of bytes transformed by the output type specifier *c* may be variable depending on  
 26096 the *LC\_CTYPE* category.

26097 The default number of bytes transformed by output type specifiers **d**, **f**, **o**, **u**, and **x** corresponds  
 26098 to the various C-language types as follows. If the *c99* compiler is present on the system, these  
 26099 specifiers shall correspond to the sizes used by default in that compiler. Otherwise, these sizes  
 26100 may vary among systems that conform to IEEE Std. 1003.1-200x.

- 26101 • For the type specifier characters **d**, **o**, **u**, and **x**, the default number of bytes shall correspond
- 26102 to the size of the underlying implementation's basic integer type. For these specifier
- 26103 characters, the implementation shall support values of the optional number of bytes to be
- 26104 converted corresponding to the number of bytes in the C-language types **char**, **short**, **int**, and
- 26105 **long**. These numbers can also be specified by an application as the characters **'C'**, **'S'**, **'I'**,
- 26106 and **'L'**, respectively.



26107 **Notes to Reviewers**26108 *This section with side shading will not appear in the final copy. - Ed.*26109 D3, XCU, ERN 99: We need to address long long, which usually uses the notation LL;  
26110 however, that is 2 characters. Do we need to invent a new single character notation for long  
26111 long?26112 The implementation shall also support the values 1, 2, and 4, even if it provides no C-  
26113 Language types of those sizes. The byte order used when interpreting numeric values is  
26114 implementation-defined, but shall correspond to the order in which a constant of the  
26115 corresponding type is stored in memory on the system.

- 26116 • For the type specifier character
- f**
- , the default number of bytes shall correspond to the number
- 
- 26117 of bytes in the underlying implementation's basic double precision floating-point data type.
- 
- 26118 The implementation shall support values of the optional number of bytes to be converted
- 
- 26119 corresponding to the number of bytes in the C-language types
- float**
- ,
- double**
- , and
- long**
- 
- 26120
- double**
- . These numbers can also be specified by an application as the characters 'F', 'D',
- 
- 26121 and 'L', respectively.

26122 The type specifier character **a** specifies that bytes are interpreted as named characters from the  
26123 International Reference Version (IRV) of the ISO/IEC 646:1991 standard. Only the least  
26124 significant seven bits of each byte shall be used for this type specification. Bytes with the values  
26125 listed in the following table shall be written using the corresponding names for those characters.26126 **Table 4-12** Named Characters in *od*

| Value | Name | Value | Name | Value | Name      | Value | Name |
|-------|------|-------|------|-------|-----------|-------|------|
| \000  | nul  | \001  | soh  | \002  | stx       | \003  | etx  |
| \004  | eot  | \005  | enq  | \006  | ack       | \007  | bel  |
| \010  | bs   | \011  | ht   | \012  | lf or nl* | \013  | vt   |
| \014  | ff   | \015  | cr   | \016  | so        | \017  | si   |
| \020  | dle  | \021  | dc1  | \022  | dc2       | \023  | dc3  |
| \024  | dc4  | \025  | nak  | \026  | syn       | \027  | etb  |
| \030  | can  | \031  | em   | \032  | sub       | \033  | esc  |
| \034  | fs   | \035  | gs   | \036  | rs        | \037  | us   |
| \040  | sp   | \177  | del  |       |           |       |      |

26138 **Note:** The "\012" value may be written either as **lf** or **nl**.26139 The type specifier character **c** specifies that bytes are interpreted as characters specified by the  
26140 current setting of the *LC\_CTYPE* locale category. Characters listed in the table in the Base  
26141 Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation ("\\", '\a',  
26142 '\b', '\f', '\n', '\r', '\t', '\v') shall be written as the corresponding escape sequences,  
26143 except that backslash shall be written as a single backslash and a NUL shall be written as '\0'.  
26144 Other non-printable characters shall be written as one three-digit octal number for each byte in  
26145 the character. If the size of a byte on the system is greater than nine bits, the format used for  
26146 non-printable characters is implementation-defined. Printable multi-byte characters shall be  
26147 written in the area corresponding to the first byte of the character; the two-character sequence  
26148 "\*\*\*" shall be written in the area corresponding to each remaining byte in the character, as an  
26149 indication that the character is continued. When either the **-j skip** or **-N count** option is specified  
26150 along with the **c** type specifier, and this results in an attempt to start or finish in the middle of a  
26151 multi-byte character, the result is implementation-defined.26152 The input data shall be manipulated in blocks, where a block is defined as a multiple of the least  
26153 common multiple of the number of bytes transformed by the specified output types. If the least

26154 common multiple is greater than 16, the results are unspecified. Each input block shall be  
 26155 written as transformed by each output type, one per written line, in the order that the output  
 26156 types were specified. If the input block size is larger than the number of bytes transformed by  
 26157 the output type, the output type shall sequentially transform the parts of the input block, and  
 26158 the output from each of the transformations shall be separated by one or more <blank>  
 26159 characters.

26160 If, as a result of the specification of the `-N` option or end-of-file being reached on the last input  
 26161 file, input data only partially satisfies an output type, the input shall be extended sufficiently  
 26162 with null bytes to write the last byte of the input.

26163 Unless `-A n` is specified, the first output line produced for each input block shall be preceded by  
 26164 the input offset, cumulative across input files, of the next byte to be written. The format of the  
 26165 input offset is unspecified; however, it shall not contain any <blank> characters, shall start at the  
 26166 first character of the output line, and shall be followed by one or more <blank> characters. In  
 26167 addition, the offset of the byte following the last byte written shall be written after all the input  
 26168 data has been processed, but shall not be followed by any <blank> characters.

26169 If no `-A` option is specified, the input offset base is unspecified.

#### 26170 EXIT STATUS

26171 The following exit values shall be returned:

26172     0 All input files were processed successfully.

26173     >0 An error occurred.

#### 26174 CONSEQUENCES OF ERRORS

26175 Default.

#### 26176 APPLICATION USAGE

26177 Applications are warned not to use file names starting with `'+'` or a first operand starting with  
 26178 a numeric character so that the old functionality can be maintained by implementations, unless  
 26179 they specify one of the new options specified by the ISO/IEC 9945-2:1993 standard. To  
 26180 guarantee that one of these file names is always interpreted as a file name, an application could  
 26181 always specify the address base format with the `-A` option.

#### 26182 EXAMPLES

26183 If a file containing 128 bytes with decimal values zero to 127, in increasing order, is supplied as  
 26184 standard input to the command:

```
26185 od -A d -t a
```

26186 on an implementation using an input block size of 16 bytes, the standard output, independent of  
 26187 the current locale setting, would be similar to:

```
26188 0000000 nul soh stx etx eot enq ack bel bs ht nl vt ff cr so si
26189 0000016 dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
26190 0000032 sp ! " # $ % & ' () * + , - . /
26191 0000048 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
26192 0000064 @ A B C D E F G H I J K L M N O
26193 0000080 P Q R S T U V W X Y Z [\] ^ _
26194 0000096 ` a b c d e f g h i j k l m n o
26195 0000112 p q r s t u v w x y z { | } ~ del
26196 0000128
```

26197 Note that this volume of IEEE Std. 1003.1-200x allows `nl` or `lf` to be used as the name for the  
 26198 ISO/IEC 646:1991 standard IRV character with decimal value 10. The IRV names this character  
 26199 `lf` (line feed), but traditional implementations have referred to this character as newline (`nl`) and

26200 the POSIX locale character set symbolic name for the corresponding character is a <newline>  
26201 character.

26202 The command:

```
26203 od -A o -t o2x2x -n 18
```

26204 on a system with 32-bit words and an implementation using an input block size of 16 bytes  
26205 could write 18 bytes in approximately the following format:

```
26206 0000000 032056 031440 041123 042040 052516 044530 020043 031464
26207 342e 3320 4253 4420 554e 4958 2023 3334
26208 342e3320 42534420 554e4958 20233334
```

```
26209 0000020 032472
```

```
26210 353a
```

```
26211 353a0000
```

```
26212 0000022
```

26213 The command:

```
26214 od -A d -t f -t o4 -t x4 -n 24 -j 0x15
```

26215 on a system with 64-bit doubles (for example, IEEE Std. 754-1985 double precision floating-point  
26216 format) would skip 21 bytes of input data and then write 24 bytes in approximately the  
26217 following format:

```
26218 0000000 1.0000000000000000e+00 1.5735000000000000e+01
26219 07774000000 00000000000 10013674121 35341217270
26220 3ff00000 00000000 402f3851 eb851eb8
```

```
26221 0000016 1.4066823000000000e+02
```

```
26222 10030312542 04370303230
```

```
26223 40619562 23e18698
```

```
26224 0000024
```

## 26225 RATIONALE

26226 The *od* utility went through several names in early proposals, including *hd*, *xd*, and most recently  
26227 *hexdump*. There were several objections to all of these based on the following reasons:

- 26228 • The *hd* and *xd* names conflicted with historical utilities that behaved differently.
- 26229 • The *hexdump* description was much more complex than needed for a simple dump utility.
- 26230 • The *od* utility has been available on all historical implementations and there was no need to  
26231 create a new name for a utility so similar to the historical *od* utility.

26232 The original reasons for not standardizing historical *od* were also fairly widespread. Those  
26233 reasons are given below along with rationale explaining why the standard developers believe  
26234 that this version does not suffer from the indicated problem:

- 26235 • The BSD and System V versions of *od* have diverged, and the intersection of features  
26236 provided by both does not meet the needs of the user community. In fact, the System V  
26237 version only provides a mechanism for dumping octal bytes and **shorts**, signed and unsigned  
26238 decimal **shorts**, hexadecimal **shorts**, and ASCII characters. BSD added the ability to dump  
26239 **floats**, **doubles**, named ASCII characters, and octal, signed decimal, unsigned decimal, and  
26240 hexadecimal **longs**. The version presented here provides more normalized forms for  
26241 dumping bytes, **shorts**, **ints**, and **longs** in octal, signed decimal, unsigned decimal, and  
26242 hexadecimal; **float**, **double**, and **long double**; and named ASCII as well as current locale  
26243 characters.

- 26244  
26245  
26246  
26247
- It would not be possible to come up with a compatible superset of the BSD and System V flags that met the requirements of the standard developers. The historical default *od* output is the specified default output of this utility. None of the option letters chosen for this version of *od* conflict with any of the options to historical versions of *od*.
- 26248  
26249  
26250  
26251  
26252  
26253  
26254  
26255  
26256  
26257  
26258  
26259  
26260  
26261  
26262
- On systems with different sizes for **short**, **int**, and **long**, there was no way to ask for dumps of **ints**, even in the BSD version. Because of the way options are named, the name space could not be extended to solve these problems. This is why the **-t** option was added (with type specifiers more closely matched to the *printf()* formats used in the rest of this volume of IEEE Std. 1003.1-200x) and the optional field sizes were added to the **d**, **f**, **o**, **u**, and **x** type specifiers. It is also one of the reasons why the historical practice was not mandated as a required obsolescent form of *od*. (Although the old versions of *od* are not listed as an obsolescent form, implementations are urged to continue to recognize the older forms for several more years.) The **a**, **c**, **f**, **o**, and **x** types match the meaning of the corresponding format characters in the historical implementations of *od* except for the default sizes of the fields converted. The **d** format is signed in this volume of IEEE Std. 1003.1-200x to match the *printf()* notation. (Historical versions of *od* used **d** as a synonym for **u** in this version. The System V implementation uses **s** for signed decimal; BSD uses **i** for signed decimal and **s** for null-terminated strings.) Other than **d** and **u**, all of the type specifiers match format characters in the historical BSD version of *od*.
- 26263  
26264  
26265  
26266  
26267  
26268  
26269  
26270  
26271  
26272  
26273  
26274  
26275  
26276  
26277  
26278  
26279
- The sizes of the C-language types **char**, **short**, **int**, **long**, **float**, **double**, and **long double** are used even though it is recognized that there may be zero or more than one compiler for the C language on an implementation and that they may use different sizes for some of these types. (For example, one compiler might use 2 bytes **shorts**, 2 bytes **ints**, and 4 bytes **longs**, while another compiler (or an option to the same compiler) uses 2 bytes **shorts**, 4 bytes **ints**, and 4 bytes **longs**.) Nonetheless, there has to be a basic size known by the implementation for these types, corresponding to the values reported by invocations of the *getconf* utility when called with *system\_var* operands {UCHAR\_MAX}, {USHORT\_MAX}, {UINT\_MAX}, and {ULONG\_MAX} for the types **char**, **short**, **int**, and **long**, respectively. There are similar constants required by the ISO C standard, but not required by the System Interfaces volume of IEEE Std. 1003.1-200x or this volume of IEEE Std. 1003.1-200x. They are {FLT\_MANT\_DIG}, {DBL\_MANT\_DIG}, and {LDBL\_MANT\_DIG} for the types **float**, **double**, and **long double**, respectively. If the optional *c99* utility is provided by the implementation and used as specified by this volume of IEEE Std. 1003.1-200x, these are the sizes that would be provided. If an option is used that specifies different sizes for these types, there is no guarantee that the *od* utility is able to interpret binary data output by such a program correctly.
- 26280  
26281  
26282
- This volume of IEEE Std. 1003.1-200x requires that the numeric values of these lengths be recognized by the *od* utility and that symbolic forms also be recognized. Thus, a portable application can always look at an array of **unsigned long** data elements using *od -t uL*.
- 26283  
26284  
26285
- The method of specifying the format for the address field based on specifying a starting offset in a file unnecessarily tied the two together. The **-A** option now specifies the address base and the **-S** option specifies a starting offset.
- 26286  
26287  
26288  
26289  
26290  
26291  
26292  
26293
- It would be difficult to break the dependence on U.S. ASCII to achieve an internationalized utility. It does not seem to be any harder for *od* to dump characters in the current locale than it is for the *ed* or *sed* **l** commands. The **c** type specifier does this without difficulty and is completely compatible with the historical implementations of the **c** format character when the current locale uses a superset of the ISO/IEC 646:1991 standard as a codeset. The **a** type specifier (from the BSD **a** format character) was left as a portable means to dump ASCII (or more correctly ISO/IEC 646:1991 standard (IRV)) so that headers produced by *pax* could be deciphered even on systems that do not use the ISO/IEC 646:1991 standard as a subset of

- 26294 their base codeset.
- 26295 The use of "\*\*\*" as an indication of continuation of a multi-byte character in `c` specifier output  
26296 was chosen based on seeing an implementation that uses this method. The continuation bytes  
26297 have to be marked in a way that is not ambiguous with another single-byte or multi-byte  
26298 character.
- 26299 An early proposal used `-S` and `-n`, respectively, for the `-j` and `-N` options eventually selected.  
26300 These were changed to avoid conflicts with historical implementations.
- 26301 The original standard specified `-t o2` as the default when no output type was given. This was  
26302 changed to `-t oS` (the length of a **short**) to accommodate a supercomputer implementation that  
26303 historically used 64 bits as its default (and that defined shorts as 64 bits). This change should not  
26304 affect portable applications. The requirement to support lengths of 1, 2, and 4 was added at the  
26305 same time to address an historical implementation that had no two-byte data types in its C  
26306 compiler.
- 26307 The use of a basic integer data type is intended to allow the implementation to choose a word  
26308 size commonly used by applications on that architecture.
- 26309 **FUTURE DIRECTIONS**
- 26310 All option and operand interfaces marked as extensions may be withdrawn in a future issue.
- 26311 **SEE ALSO**
- 26312 *c99, sed*
- 26313 **CHANGE HISTORY**
- 26314 First released in Issue 2.
- 26315 **Issue 4**
- 26316 Aligned with the ISO/IEC 9945-2:1993 standard.
- 26317 **Issue 4, Version 2**
- 26318 The description of the `-c` option is made dependent on the current setting of the `LC_CTYPE`  
26319 category, and a reference to the POSIX locale is deleted.
- 26320 **Issue 5**
- 26321 In the description of the `-c` option, the phrase "This is equivalent to `-t c`." is deleted.  
26322 The FUTURE DIRECTIONS section has been modified.
- 26323 **Issue 6**
- 26324 The `od` utility is changed to remove the assumption that **short** was a two-byte entity, as per the  
26325 revisions in the IEEE P1003.2b draft standard.
- 26326 The normative text is reworded to avoid use of the term "must" for application requirements.

## 26327 NAME

26328 paste — merge corresponding or subsequent lines of files

## 26329 SYNOPSIS

26330 paste [-s][-d *list*] *file...*

## 26331 DESCRIPTION

26332 The *paste* utility shall concatenate the corresponding lines of the given input files, and writes the  
26333 resulting lines to standard output.

26334 The default operation of *paste* shall concatenate the corresponding lines of the input files. The  
26335 <newline> character of every line except the line from the last input file shall be replaced with a  
26336 <tab> character.

26337 If an end-of-file condition is detected on one or more input files, but not all input files, *paste* shall  
26338 behave as though empty lines were read from the files on which end-of-file was detected, unless  
26339 the *-s* option is specified.

## 26340 OPTIONS

26341 The *paste* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
26342 12.2, Utility Syntax Guidelines.

26343 The following options shall be supported:

26344 *-d list* Unless a backslash character appears in *list*, each character in *list* is an element  
26345 specifying a delimiter character. If a backslash character appears in *list*, the  
26346 backslash character and one or more characters following it are an element  
26347 specifying a delimiter character as described below. These elements specify one or  
26348 more delimiters to use, instead of the default <tab> character, to replace the  
26349 <newline> character of the input lines. The elements in *list* shall be used circularly;  
26350 that is, when the list is exhausted the first element from the list is reused. When the  
26351 *-s* option is specified:

- 26352 • The last <newline> character in a file shall not be modified.
- 26353 • The delimiter shall be reset to the first element of list after each *file* operand is  
26354 processed.

26355 When the *-s* option is not specified:

- 26356 • The <newline> characters in the file specified by the last *file* operand shall not  
26357 be modified.
- 26358 • The delimiter shall be reset to the first element of list each time a line is  
26359 processed from each file.

26360 If a backslash character appears in *list*, it and the character following it shall be  
26361 used to represent the following delimiter characters:

26362 \n <newline> character.

26363 \t <tab> character.

26364 \\ Backslash character.

26365 \0 Empty string (not a null character). If '\0' is immediately followed by the  
26366 character 'x', the character 'X', or any character defined by the *LC\_CTYPE*  
26367 **digit** keyword (see the Base Definitions volume of IEEE Std. 1003.1-200x,  
26368 Chapter 7, Locale), the results are unspecified.

- 26369 If any other characters follow the backslash, the results are unspecified.
- 26370 **-s** Concatenate all of the lines of each separate input file in command line order. The  
26371 <newline> character of every line except the last line in each input file shall be  
26372 replaced with the <tab> character, unless otherwise specified by the **-d** option.
- 26373 **OPERANDS**
- 26374 The following operand shall be supported:
- 26375 **file** A path name of an input file. If **'-'** is specified for one or more of the *files*, the  
26376 standard input shall be used; the standard input shall be read one line at a time,  
26377 circularly, for each instance of **'-'**. Implementations shall support pasting of at  
26378 least 12 *file* operands.
- 26379 **STDIN**
- 26380 The standard input shall be used only if one or more *file* operands is **'-'**. See the INPUT FILES  
26381 section.
- 26382 **INPUT FILES**
- 26383 The input files shall be text files, except that line lengths shall be unlimited.
- 26384 **ENVIRONMENT VARIABLES**
- 26385 The following environment variables shall affect the execution of *paste*:
- 26386 **LANG** Provide a default value for the internationalization variables that are unset or null.  
26387 If *LANG* is unset or null, the corresponding value from the implementation-  
26388 defined default locale shall be used. If any of the internationalization variables  
26389 contains an invalid setting, the utility shall behave as if none of the variables had  
26390 been defined.
- 26391 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
26392 internationalization variables.
- 26393 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
26394 characters (for example, single-byte as opposed to multi-byte characters in  
26395 arguments and input files).
- 26396 **LC\_MESSAGES**
- 26397 Determine the locale that should be used to affect the format and contents of  
26398 diagnostic messages written to standard error.
- 26399 **XSI** **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 26400 **ASYNCHRONOUS EVENTS**
- 26401 Default.
- 26402 **STDOUT**
- 26403 Concatenated lines of input files shall be separated by the <tab> character (or other characters  
26404 under the control of the **-d** option) and terminated by a <newline> character.
- 26405 **STDERR**
- 26406 Used only for diagnostic messages.
- 26407 **OUTPUT FILES**
- 26408 None.
- 26409 **EXTENDED DESCRIPTION**
- 26410 None.

26411 **EXIT STATUS**

26412           The following exit values shall be returned:

26413           0   Successful completion.

26414           >0  An error occurred.

26415 **CONSEQUENCES OF ERRORS**

26416           If one or more input files cannot be opened when the `-s` option is not specified, a diagnostic message shall be written to standard error, but no output is written to standard output. If the `-s` option is specified, the *paste* utility shall provide the default behavior described in Section 1.11 (on page 2224).

26420 **APPLICATION USAGE**

26421           When the escape sequences of the *list* option-argument are used in a shell script, they must be quoted; otherwise, the shell treats the `'\'` as a special character.

26423           Portable applications should only use the specific backslash escaped delimiters presented in this volume of IEEE Std. 1003.1-200x. Historical implementations treat `'\x'`, where `'x'` is not in this list, as `'x'`, but future implementations are free to expand this list to recognize other common escapes similar to those accepted by *printf* and other standard utilities.

26427           Most of the standard utilities work on text files. The *cut* utility can be used to turn files with arbitrary line lengths into a set of text files containing the same data. The *paste* utility can be used to create (or recreate) files with arbitrary line lengths. For example, if *file* contains long lines:

```
26430 cut -b 1-500 -n file > file1
```

```
26431 cut -b 501- -n file > file2
```

26432           creates **file1** (a text file) with lines no longer than 500 bytes (plus the <newline> character) and **file2** that contains the remainder of the data from *file*. Note that **file2** is not a text file if there are lines in *file* that are longer than 500 + {LINE\_MAX} bytes. The original file can be recreated from **file1** and **file2** using the command:

```
26436 paste -d "\0" file1 file2 > file
```

26437           The commands:

```
26438 paste -d "\0" ...
```

```
26439 paste -d " " ...
```

26440           are not necessarily equivalent; the latter is not specified by this volume of IEEE Std. 1003.1-200x and may result in an error. The construct `'\0'` is used to mean “no separator” because historical versions of *paste* did not follow the syntax guidelines, and the command:

```
26443 paste -d" " ...
```

26444           could not be handled properly by *getopt*().

26445 **EXAMPLES**

26446           1. Write out a directory in four columns:

```
26447 ls | paste - - - -
```

26448           2. Combine pairs of lines from a file into single lines:

```
26449 paste -s -d "\t\n" file
```



26450 **RATIONALE**

26451 None.

26452 **FUTURE DIRECTIONS**

26453 None.

26454 **SEE ALSO**26455 *cut, grep, pr*26456 **CHANGE HISTORY**

26457 First released in Issue 2.

26458 **Issue 4**

26459 Aligned with the ISO/IEC 9945-2:1993 standard.

26460 **Issue 6**

26461 The normative text is reworded to avoid use of the term “must” for application requirements.

## 26462 NAME

26463 patch — apply changes to files

## 26464 SYNOPSIS

```
26465 UP patch [-blNR][-c | -e | -n][-d dir][-D define][-i patchfile]
26466 [-o outfile][-p num][-r rejectfile][file]
```

26467

## 26468 DESCRIPTION

26469 The *patch* utility shall read a source (patch) file containing any of the three forms of difference (diff) listings produced by the *diff* utility (normal, context or in the style of *ed*) and apply those differences to a file. By default, *patch* shall read from the standard input.

26472 The *patch* utility shall attempt to determine the type of the *diff* listing, unless overruled by a *-c*, *-e*, or *-n* option.

26474 If the patch file contains more than one patch, *patch* shall attempt to apply each of them as if they came from separate patch files. (In this case, the application shall ensure that the name of the patch file is determinable for each *diff* listing.)

## 26477 OPTIONS

26478 The *patch* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

26480 The following options shall be supported:

26481 **-b** Save a copy of the original contents of each modified file, before the differences are applied, in a file of the same name with the suffix **.orig** appended to it. If the file already exists, it shall be overwritten; if multiple patches are applied to the same file, the **.orig** file shall be written only for the first patch. When the *-o outfile* option is also specified, *file.orig* shall not be created but, if *outfile* already exists, *outfile.orig* shall be created.

26487 **-c** Interpret the patch file as a context difference (the output of the utility *diff* when the *-c* or *-C* options are specified).

26489 **-d dir** Change the current directory to *dir* before processing as described in the EXTENDED DESCRIPTION section.

26491 **-D define** Mark changes with one of the following C preprocessor constructs:

```
26492 #ifdef define
26493 ...
26494 #endif
26495 #ifndef define
26496 ...
26497 #endif
```

26498 optionally combined with the C preprocessor construct **#else**.

26499 **-e** Interpret the patch file as an *ed* script, rather than a *diff* script.

26500 **-i patchfile** Read the patch information from the file named by the path name *patchfile*, rather than the standard input.

26502 **-l** (The letter ell.) Cause any sequence of <blank> characters in the difference script to match any sequence of <blank> characters in the input file. Other characters shall be matched exactly.

26505        **-n**           Interpret the script as a normal difference.

26506        **-N**           Ignore patches where the differences have already been applied to the file; by  
26507           default, already-applied patches shall be rejected.

26508        **-o outfile**   Instead of modifying the files (specified by the *file* operand or the difference  
26509           listings) directly, write a copy of the file referenced by each patch, with the  
26510           appropriate differences applied, to *outfile*. Multiple patches for a single file shall  
26511           be applied to the intermediate versions of the file created by any previous patches,  
26512           and shall result in multiple, concatenated versions of the file being written to  
26513           *outfile*.

26514        **-p num**       For all path names in the patch file that indicate the names of files to be patched,  
26515           delete *num* path name components from the beginning of each path name. If the  
26516           path name in the patch file is absolute, any leading slashes shall be considered the  
26517           first component (that is, **-p 1** shall remove the leading slashes). Specifying **-p 0**  
26518           shall cause the full path name to be used. If **-p** is not specified, only the basename  
26519           (the final path name component) shall be used.

26520        **-R**           Reverse the sense of the patch script; that is, assume that the difference script was  
26521           created from the new version to the old version. The **-R** option cannot be used  
26522           with *ed* scripts. The *patch* utility shall attempt to reverse each portion of the script  
26523           before applying it. Rejected differences shall be saved in swapped format. If this  
26524           option is not specified, and until a portion of the patch file is successfully applied,  
26525           *patch* attempts to apply each portion in its reversed sense as well as in its normal  
26526           sense. If the attempt is successful, the user shall be prompted to determine if the  
26527           **-R** option should be set.

26528        **-r rejectfile**   Override the default reject file name. In the default case, the reject file shall have  
26529           the same name as the output file, with the suffix **.rej** appended to it; see **Patch**  
26530           **Application** (on page 2903).

26531 **OPERANDS**

26532        The following operand shall be supported:

26533        *file*           A path name of a file to patch.

26534 **STDIN**

26535        See the INPUT FILES section.

26536 **INPUT FILES**

26537        Input files shall be text files.

26538 **ENVIRONMENT VARIABLES**

26539        The following environment variables shall affect the execution of *patch*:

26540        **LANG**           Provide a default value for the internationalization variables that are unset or null.  
26541           If **LANG** is unset or null, the corresponding value from the implementation-  
26542           defined default locale shall be used. If any of the internationalization variables  
26543           contains an invalid setting, the utility shall behave as if none of the variables had  
26544           been defined.

26545        **LC\_ALL**        If set to a non-empty string value, override the values of all the other  
26546           internationalization variables.

26547        **LC\_CTYPE**     Determine the locale for the interpretation of sequences of bytes of text data as  
26548           characters (for example, single-byte as opposed to multi-byte characters in  
26549           arguments and input files).

- 26550 **LC\_MESSAGES**
- 26551 Determine the locale that should be used to affect the format and contents of
- 26552 diagnostic messages written to standard error and informative messages written to
- 26553 standard output.
- 26554 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 26555 **LC\_TIME** Determine the locale for recognizing the format of file timestamps written by the
- 26556 *diff* utility in a context-difference input file.
- 26557 **ASYNCHRONOUS EVENTS**
- 26558 Default.
- 26559 **STDOUT**
- 26560 Not used.
- 26561 **STDERR**
- 26562 Used for diagnostic and informational messages.
- 26563 **OUTPUT FILES**
- 26564 The output of the *patch* utility, the save files (**.orig** suffixes) and the reject files (**.rej** suffixes) shall
- 26565 be text files.
- 26566 **EXTENDED DESCRIPTION**
- 26567 A patchfile may contain patching instructions for more than one file; file names shall be
- 26568 determined as specified in **File Name Determination** (on page 2903). When the **-b** option is
- 26569 specified, for each patched file, the original shall be saved in a file of the same name with the
- 26570 suffix **.orig** appended to it.
- 26571 For each patched file, a reject file may also be created as noted in **Patch Application** (on page
- 26572 2903). In the absence of a **-r** option, the name of this file shall be formed by appending the suffix
- 26573 **.rej** to the original file name.
- 26574 **Patchfile Format**
- 26575 The patch file shall contain zero or more lines of header information followed by one or more
- 26576 patches. Each patch shall contain zero or more lines of file name identification in the format
- 26577 produced by *diff -c*, and one or more sets of *diff* output, which are customarily called *hunks*.
- 26578 The *patch* utility shall recognize the following expression in the header information:
- 26579 **Index:** *pathname*
- 26580 The file to be patched is named *pathname*.
- 26581 If all lines (including headers) within a patch begin with the same leading sequence of <blank>
- 26582 characters, the *patch* utility shall remove this sequence before proceeding. Within each patch, if
- 26583 the type of difference is context, the *patch* utility shall recognize the following expressions:
- 26584 **\*\*\*** *filename timestamp*
- 26585 The patches arose from *filename*.
- 26586 **---** *filename timestamp*
- 26587 The patches should be applied to *filename*.
- 26588 Each hunk within a patch shall be the *diff* output to change a line range within the original file.
- 26589 The line numbers for successive hunks within a patch shall occur in ascending order.

26590 **File Name Determination**

26591 If no *file* operand is specified, *patch* shall perform the following steps to determine the file name  
26592 to use:

- 26593 1. If the type of *diff* is context, the *patch* utility shall delete path name components (as  
26594 specified by the **-p** option) from the file name on the line beginning with "\*\*\*\*", then test  
26595 for the existence of this file relative to the current directory (or the directory specified with  
26596 the **-d** option). If the file exists, the *patch* utility shall use this file name.
- 26597 2. If the type of *diff* is context, the *patch* utility shall delete the path name components (as  
26598 specified by the **-p** option) from the file name on the line beginning with "——", then test  
26599 for the existence of this file relative to the current directory (or the directory specified with  
26600 the **-d** option). If the file exists, the *patch* utility shall use this file name.
- 26601 3. If the header information contains a line beginning with the string **Index:**, the *patch* utility  
26602 shall delete path name components (as specified by the **-p** option) from this line, then test  
26603 for the existence of this file relative to the current directory (or the directory specified with  
26604 the **-d** option). If the file exists, the *patch* utility shall use this file name.
- 26605 XSI 4. If an **SCCS** directory exists in the current directory, *patch* shall attempt to perform a *get -e*  
26606 **SCCS/s.filename** command to retrieve an editable version of the file. If the file exists, the  
26607 *patch* utility shall use this file name.
- 26608 5. The *patch* utility shall write a prompt to standard output and request a file name  
26609 interactively from the controlling terminal (for example, **/dev/tty**).

26610 **Patch Application**

26611 If the **-c**, **-e**, or **-n** option is present, the *patch* utility shall interpret information within each hunk  
26612 as a context difference, an *ed* difference or a normal difference, respectively. In the absence of  
26613 any of these options, the *patch* utility shall determine the type of difference based on the format  
26614 of information within the hunk.

26615 For each hunk, the *patch* utility shall begin to search for the place to apply the patch at the line  
26616 number at the beginning of the hunk, plus or minus any offset used in applying the previous  
26617 hunk. If lines matching the hunk context are not found, *patch* shall scan both forwards and  
26618 backwards at least 1 000 bytes for a set of lines that match the hunk context.

26619 If no such place is found and it is a context difference, then another scan shall take place,  
26620 ignoring the first and last line of context. If that fails, the first two and last two lines of context  
26621 shall be ignored and another scan shall be made. Implementations may search more extensively  
26622 for installation locations.

26623 If no location can be found, the *patch* utility shall append the hunk to the reject file. The rejected  
26624 hunk shall be written in context-difference format regardless of the format of the patch file. If the  
26625 input was a normal or *ed-style* difference, the reject file may contain differences with zero lines  
26626 of context. The line numbers on the hunks in the reject file may be different from the line  
26627 numbers in the patch file since they shall reflect the approximate locations for the failed hunks in  
26628 the new file rather than the old one.

26629 If the type of patch is an *ed* diff, the implementation may accomplish the patching by invoking  
26630 the *ed* utility.

26631 **EXIT STATUS**

26632 The following exit values shall be returned:

- 26633 0 Successful completion.

26634 1 One or more lines were written to a reject file.

26635 >1 An error occurred.

#### 26636 CONSEQUENCES OF ERRORS

26637 Patches that cannot be correctly placed in the file shall be written to a reject file.

#### 26638 APPLICATION USAGE

26639 The **-R** option does not work with *ed* scripts because there is too little information to reconstruct  
26640 the reverse operation.

26641 The **-p** option makes it possible to customize a patchfile to local user directory structures  
26642 without manually editing the patchfile. For example, if the file name in the patch file was:

26643 /curds/whey/src/blurfl/blurfl.c

26644 Setting **-p 0** gives the entire path name unmodified; **-p 1** gives:

26645 curds/whey/src/blurfl/blurfl.c

26646 without the leading slash, **-p 4** gives:

26647 blurfl/blurfl.c

26648 and not specifying **-p** at all gives:

26649 blurfl.c .

#### 26650 EXAMPLES

26651 None.

#### 26652 RATIONALE

26653 Some of the functionality in historical *patch* implementations was not specified. The following  
26654 documents those features present in historical implementations that have not been specified.

26655 A deleted piece of functionality was the '+' pseudo-option allowing an additional set of options  
26656 and a patch file operand to be given. This was seen as being insufficiently useful to standardize.

26657 In historical implementations, if the string "Prereq:" appeared in the header, the *patch* utility  
26658 would search for the corresponding version information (the string specified in the header,  
26659 delimited by <blank>s or the beginning or end of a line or the file) anywhere in the original file.  
26660 This was deleted as too simplistic and insufficiently trustworthy a mechanism to standardize.  
26661 For example, if:

26662 Prereq: 1.2

26663 were in the header, the presence of a delimited 1.2 anywhere in the file would satisfy the  
26664 prerequisite.

26665 The following options were dropped from historical implementations of *patch* as insufficiently  
26666 useful to standardize:

26667 **-b** The **-b** option historically provided a method for changing the name extension of  
26668 the backup file from the default **.orig**. This option has been modified and retained  
26669 in this volume of IEEE Std. 1003.1-200x.

26670 **-F** The **-F** option specified the number of lines of a context diff to ignore when  
26671 searching for a place to install a patch.

26672 **-f** The **-f** option historically caused *patch* not to request additional information from  
26673 the user.

- 26674        **-r**            The **-r** option historically provided a method of overriding the extension of the  
26675                    reject file from the default **.rej**.
- 26676        **-s**            The **-s** option historically caused *patch* to work silently unless an error occurred.
- 26677        **-x**            The **-x** option historically set internal debugging flags.
- 26678                    In some file system implementations, the saving of a **.orig** file may produce unwanted results. In  
26679                    the case of 12, 13, or 14-character file names (on file systems supporting 14-character maximum  
26680                    file names), the **.orig** file overwrites the new file. The reject file may also exceed this file name  
26681                    limit. It was suggested, due to some historical practice, that a tilde ('~') suffix be used instead  
26682                    of **.orig** and some other character instead of the **.rej** suffix. This was rejected because it is not  
26683                    obvious to the user which file is which. The suffixes **.orig** and **.rej** are clearer and more  
26684                    understandable.
- 26685                    The **-b** option has the opposite sense in some historical implementations—do not save the **.orig**  
26686                    file. The default case here is not to save the files, making *patch* behave more consistently with the  
26687                    other standard utilities.
- 26688                    The **-w** option in early proposals was changed to **-I** to match historical practice.
- 26689                    The **-N** option was included because without it, a non-interactive application cannot reject  
26690                    previously applied patches. For example, if a user is piping the output of *diff* into the *patch*  
26691                    utility, and the user only wants to patch a file to a newer version non-interactively, the **-N**  
26692                    option is required.
- 26693                    Changes to the **-I** option description were proposed to allow matching across <newline>s in  
26694                    addition to just <blank>s. Since this is not historical practice, and since some ambiguities could  
26695                    result, it is suggested that future developments in this area utilize another option letter, such as  
26696                    **-L**.
- 26697   **FUTURE DIRECTIONS**
- 26698                    None.
- 26699   **SEE ALSO**
- 26700                    *ed*, *diff*
- 26701   **CHANGE HISTORY**
- 26702                    First released in Issue 4.
- 26703   **Issue 5**
- 26704                    **FUTURE DIRECTIONS** section added.
- 26705   **Issue 6**
- 26706                    This utility is now marked as part of the User Portability Utilities option.
- 26707                    The description of the **-D** option and the steps in **File Name Determination** (on page 2903) are  
26708                    changed to match historical practice as defined in the IEEE P1003.2b draft standard.
- 26709                    The normative text is reworded to avoid use of the term “must” for application requirements.

## 26710 NAME

26711 pathchk — check path names

## 26712 SYNOPSIS

26713 pathchk [-p] *pathname...*

## 26714 DESCRIPTION

26715 The *pathchk* utility shall check that one or more path names are valid (that is, they could be used  
 26716 to access or create a file without causing syntax errors) and portable (that is, no file name  
 26717 truncation results). More extensive portability checks are provided by the **-p** option.

26718 By default, the *pathchk* utility shall check each component of each *pathname* operand based on the  
 26719 underlying file system. A diagnostic shall be written for each *pathname* operand that:

- 26720 • Is longer than {PATH\_MAX} bytes (see **Path Name Variable Values** in the Base Definitions  
 26721 volume of IEEE Std. 1003.1-200x, Chapter 13, Headers, <limits.h>)
- 26722 • Contains any component longer than {NAME\_MAX} bytes in its containing directory
- 26723 • Contains any component in a directory that is not searchable
- 26724 • Contains any character in any component that is not valid in its containing directory

26725 The format of the diagnostic message is not specified, but shall indicate the error detected and  
 26726 the corresponding *pathname* operand.

26727 It shall not be considered an error if one or more components of a *pathname* operand do not exist  
 26728 as long as a file matching the path name specified by the missing components could be created  
 26729 that does not violate any of the checks specified above.

## 26730 OPTIONS

26731 The *pathchk* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 26732 12.2, Utility Syntax Guidelines.

26733 The following option shall be supported:

- 26734 **-p** Instead of performing checks based on the underlying file system, write a  
 26735 diagnostic for each *pathname* operand that:
  - 26736 • Is longer than {\_POSIX\_PATH\_MAX} bytes (see **Minimum Values** in the Base  
 26737 Definitions volume of IEEE Std. 1003.1-200x, Chapter 13, Headers, <limits.h>)
  - 26738 • Contains any component longer than {\_POSIX\_NAME\_MAX} bytes
  - 26739 • Contains any character in any component that is not in the portable file name  
 26740 character set

## 26741 OPERANDS

26742 The following operand shall be supported:

26743 *pathname* A path name to be checked.

## 26744 STDIN

26745 Not used.

## 26746 INPUT FILES

26747 None.

## 26748 ENVIRONMENT VARIABLES

26749 The following environment variables shall affect the execution of *pathchk*:

26750 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 26751 If *LANG* is unset or null, the corresponding value from the implementation-



26752 defined default locale shall be used. If any of the internationalization variables  
 26753 contains an invalid setting, the utility shall behave as if none of the variables had  
 26754 been defined.

26755 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 26756 internationalization variables.

26757 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 26758 characters (for example, single-byte as opposed to multi-byte characters in  
 26759 arguments).

26760 **LC\_MESSAGES**  
 26761 Determine the locale that should be used to affect the format and contents of  
 26762 diagnostic messages written to standard error.

26763 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

26764 **ASYNCHRONOUS EVENTS**  
 26765 Default.

26766 **STDOUT**  
 26767 Not used.

26768 **STDERR**  
 26769 Used only for diagnostic messages.

26770 **OUTPUT FILES**  
 26771 None.

26772 **EXTENDED DESCRIPTION**  
 26773 None.

26774 **EXIT STATUS**  
 26775 The following exit values shall be returned:  
 26776 0 All *pathname* operands passed all of the checks.  
 26777 >0 An error occurred.

26778 **CONSEQUENCES OF ERRORS**  
 26779 Default.

26780 **APPLICATION USAGE**  
 26781 The *test* utility can be used to determine whether a given path name names an existing file; it  
 26782 does not, however, give any indication of whether or not any component of the path name was  
 26783 truncated in a directory where the **\_POSIX\_NO\_TRUNC** feature is not in effect. The *pathchk*  
 26784 utility does not check for file existence; it performs checks to determine if a path name does exist  
 26785 or could be created with no path name component truncation.

26786 The *noclobber* option in the shell (see the *set* (on page 2297) special built-in) can be used to  
 26787 atomically create a file. As with all file creation semantics in the System Interfaces volume of  
 26788 IEEE Std. 1003.1-200x, it guarantees atomic creation, but still depends on applications to agree  
 26789 on conventions and cooperate on the use of files after they have been created.

26790 **EXAMPLES**  
 26791 To verify that all path names in an imported data interchange archive are legitimate and  
 26792 unambiguous on the current system:  
 26793 

```
pax -f archive | sed -e '/ == ./s///' | xargs pathchk
```

  
 26794 

```
if [$? -eq 0]
```

  
 26795 

```
then
```

```

26796 pax -r -f archive
26797 else
26798 echo Investigate problems before importing files.
26799 exit 1
26800 fi

26801 To verify that all files in the current directory hierarchy could be moved to any system
26802 conforming to the System Interfaces volume of IEEE Std. 1003.1-200x that also supports the pax
26803 utility:

26804 find . -print | xargs pathchk -p
26805 if [$? -eq 0]
26806 then
26807 pax -w -f archive .
26808 else
26809 echo Portable archive cannot be created.
26810 exit 1
26811 fi

26812 To verify that a user-supplied path name names a readable file and that the application can
26813 create a file extending the given path without truncation and without overwriting any existing
26814 file:

26815 case $- in
26816 *C*) reset="";;
26817 *) reset="set +C"
26818 set -C;;
26819 esac
26820 test -r "$path" && pathchk "$path.out" &&
26821 rm "$path.out" > "$path.out"
26822 if [$? -ne 0]; then
26823 printf "%s: %s not found or %s.out fails \
26824 creation checks.\n" $0 "$path" "$path"
26825 $reset # Reset the noclobber option in case a trap
26826 # on EXIT depends on it.
26827 exit 1
26828 fi
26829 $reset
26830 PROCESSING < "$path" > "$path.out"

26831 The following assumptions are made in this example:

26832 1. PROCESSING represents the code that is used by the application to use $path once it is
26833 verified that $path.out works as intended.

26834 2. The state of the noclobber option is unknown when this code is invoked and should be set
26835 on exit to the state it was in when this code was invoked. (The reset variable is used in this
26836 example to restore the initial state.)

26837 3. Note the usage of:

26838 rm "$path.out" > "$path.out"

26839 a. The pathchk command has already verified, at this point, that $path.out is not
26840 truncated.

26841 b. With the noclobber option set, the shell verifies that $path.out does not already exist
26842 before invoking rm.

```

26843 c. If the shell succeeded in creating **\$path.out**, *rm* removes it so that the application can  
26844 create the file again in the **PROCESSING** step.

26845 d. If the **PROCESSING** step wants the file to exist already when it is invoked, the:

26846 `rm "$path.out" > "$path.out"`

26847 should be replaced with:

26848 `> "$path.out"`

26849 which verifies that the file did not already exist, but leaves **\$path.out** in place for use  
26850 by **PROCESSING**.

#### 26851 RATIONALE

26852 The *pathchk* utility is new, commissioned for this volume of IEEE Std. 1003.1-200x. It, along with  
26853 the *set -C(noclobber)* option added to the shell, replaces the *mktemp*, *validnam*, and *create* utilities  
26854 that appeared in early proposals. All of these utilities were attempts to solve several common  
26855 problems:

26856 • Verify the validity (for several different definitions of “valid”) of a path name supplied by a  
26857 user, generated by an application, or imported from an external source.

26858 • Atomically create a file.

26859 • Perform various string handling functions to generate a temporary file name.

26860 The *create* utility, included in an early proposal, provided checking and atomic creation in a  
26861 single invocation of the utility; these are orthogonal issues and need not be grouped into a single  
26862 utility. Note that the *noclobber* option also provides a way of creating a lock for process  
26863 synchronization; since it provides an atomic *create*, there is no race between a test for existence  
26864 and the following creation if it did not exist.

26865 Having a function like *tmpnam()* in the ISO C standard is important in many high-level  
26866 languages. The shell programming language, however, has built-in string manipulation  
26867 facilities, making it very easy to construct temporary file names. The names needed obviously  
26868 depend on the application, but are frequently of a form similar to:

26869 `$TMPDIR/application_abbreviation$$ .suffix`

26870 In cases where there is likely to be contention for a given suffix, a simple shell *for* or *while* loop  
26871 can be used with the shell *noclobber* option to create a file without risk of collisions, as long as  
26872 applications trying to use the same file name name space are cooperating on the use of files after  
26873 they have been created.

#### 26874 FUTURE DIRECTIONS

26875 None.

#### 26876 SEE ALSO

26877 *test*, Section 2.7 (on page 2251)

#### 26878 CHANGE HISTORY

26879 First released in Issue 4.

## 26880 NAME

26881 pax — portable archive interchange

## 26882 SYNOPSIS

26883 pax [-cdnv][[-H|-L][[-f *archive*][[-s *replstr*]]...[*pattern*...]26884 pax -r[-cdiknuv][[-H|-L][[-f *archive*][[-o *options*]]...[-p *string*]]...  
26885 [-s *replstr*]]...[*pattern*...]26886 pax -w[-dituvX][[-H|-L][[-b *blocksize*][[-a][[-f *archive*][[-o *options*]]...]  
26887 [-s *replstr*]]...[-x *format*][*file*...]26888 pax -r -w[-diklntuvX][[-H|-L][[-p *string*]]...[-s *replstr*]]...  
26889 [*file*...] *directory*

## 26890 DESCRIPTION

26891 The *pax* utility shall read, write, and write lists of the members of archive files and copy  
26892 directory hierarchies. A variety of archive formats shall be supported; see the *-x format* option.26893 The action to be taken depends on the presence of the *-r* and *-w* options. The four combinations  
26894 of *-r* and *-w* are referred to as the four modes of operation: **list**, **read**, **write**, and **copy** modes,  
26895 corresponding respectively to the four forms shown in the SYNOPSIS section.26896 **list** In **list** mode (when neither *-r* nor *-w* are specified), *pax* shall write the names of  
26897 the members of the archive file read from the standard input, with path names  
26898 matching the specified patterns, to standard output. If a named file is of type  
26899 directory, the file hierarchy rooted at that file shall be listed as well.26900 **read** In **read** mode (when *-r* is specified, but *-w* is not), *pax* shall extract the members of  
26901 the archive file read from the standard input, with path names matching the  
26902 specified patterns. If an extracted file is of type directory, the file hierarchy rooted  
26903 at that file shall be extracted as well. The extracted files shall be created relative to  
26904 the current file hierarchy.26905 If an attempt is made to extract a directory when the directory already exists, this  
26906 shall not be considered to be an error. If an attempt is made to extract a FIFO when  
26907 the FIFO already exists, this shall not be considered to be an error.26908 The ownership, access, and modification times, and file mode of the restored files  
26909 are discussed under the *-p* option.26910 **write** In **write** mode (when *-w* is specified, but *-r* is not), *pax* shall write the contents of  
26911 the *file* operands to the standard output in an archive format. If no *file* operands are  
26912 specified, a list of files to copy, one per line, shall be read from the standard input.  
26913 A file of type directory shall include all of the files in the file hierarchy rooted at the  
26914 file.26915 **copy** In **copy** mode (when both *-r* and *-w* are specified), *pax* shall copy the *file* operands  
26916 to the destination directory.26917 If no *file* operands are specified, a list of files to copy, one per line, shall be read  
26918 from the standard input. A file of type directory shall include all of the files in the  
26919 file hierarchy rooted at the file.26920 The effect of the **copy** shall be as if the copied files were written to an archive file  
26921 and then subsequently extracted, except that there may be hard links between the  
26922 original and the copied files. If the destination directory is a subdirectory of one of  
26923 the files to be copied, the results are unspecified. If the destination directory is a  
26924 file of a type not defined by the System Interfaces volume of IEEE Std. 1003.1-200x,

26925 the results are implementation-defined; otherwise, it shall be an error for the file  
 26926 named by the *directory* operand not to exist, not be writable by the user, or not be a  
 26927 file of type directory.

26928 In **read** or **copy** modes, if intermediate directories are necessary to extract an archive member,  
 26929 *pax* shall perform actions equivalent to the *mkdir()* function defined in the System Interfaces  
 26930 volume of IEEE Std. 1003.1-200x, called with the following arguments:

- 26931 • The intermediate directory used as the *path* argument
- 26932 • The value of the bitwise-inclusive OR of S\_IRWXU, S\_IRWXG, and S\_IRWXO as the *mode*  
 26933 argument

26934 If any specified *pattern* or *file* operands are not matched by at least one file or archive member,  
 26935 *pax* shall write a diagnostic message to standard error for each one that did not match and exit  
 26936 with a non-zero exit status.

26937 The archive formats described in the EXTENDED DESCRIPTION section shall be automatically  
 26938 detected on input. The default output archive format shall be implementation-defined.

26939 A single archive can span multiple files. The *pax* utility shall determine, in an implementation-  
 26940 defined manner, what file to read or write as the next file.

26941 If the selected archive format supports the specification of linked files, it shall be an error if these  
 26942 files cannot be linked when the archive is extracted. For archive formats that do not store file  
 26943 contents with each name that causes a hard link, if the file that contains the data is not extracted  
 26944 during this *pax* session, either the data shall be restored from the original file, or a diagnostic  
 26945 message shall be displayed with the name of a file that can be used to extract the data. In  
 26946 traversing directories, *pax* shall detect infinite loops; that is, entering a previously visited  
 26947 directory that is an ancestor of the last file visited. When it detects an infinite loop, *pax* shall  
 26948 write a diagnostic message to standard error and shall terminate.

#### 26949 OPTIONS

26950 The *pax* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 26951 12.2, Utility Syntax Guidelines, except that the order of presentation of the **-o**, **-p**, and **-s** options  
 26952 is significant.

26953 The following options shall be supported:

- 26954 **-r** Read an archive file from standard input.
- 26955 **-w** Write files to the standard output in the specified archive format.
- 26956 **-a** Append files to the end of the archive. It is implementation-defined which devices  
 26957 on the system support appending. Additional file formats unspecified by this  
 26958 volume of IEEE Std. 1003.1-200x may impose restrictions on appending.
- 26959 **-b *blocksize*** Block the output at a positive decimal integer number of bytes per write to the  
 26960 archive file. Devices and archive formats may impose restrictions on blocking.  
 26961 Blocking shall be automatically determined on input. Portable applications shall  
 26962 not specify a *blocksize* value larger than 32 256. Default blocking when creating  
 26963 archives depends on the archive format. (See the **-x** option below.)
- 26964 **-c** Match all file or archive members except those specified by the *pattern* or *file*  
 26965 operands.
- 26966 **-d** Cause files of type directory being copied or archived or archive members of type  
 26967 directory being extracted or listed to match only the file or archive member itself  
 26968 and not the file hierarchy rooted at the file.

|       |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26969 | <b>-f</b> <i>archive</i> | Specify the path name of the input or output archive, overriding the default standard input (in <b>list</b> or <b>read</b> modes) or standard output ( <b>write</b> mode).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 26970 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26971 | <b>-H</b>                | If a symbolic link referencing a file of type directory is specified on the command line, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. The default behavior shall be to archive the symbolic link itself.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 26972 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26973 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26974 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26975 | <b>-i</b>                | Interactively rename files or archive members. For each archive member matching a <i>pattern</i> operand or file matching a <i>file</i> operand, a prompt shall be written to the file <b>/dev/tty</b> . The prompt shall contain the name of the file or archive member, but the format is otherwise unspecified. A line shall then be read from <b>/dev/tty</b> . If this line is blank, the file or archive member shall be skipped. If this line consists of a single period, the file or archive member shall be processed with no modification to its name. Otherwise, its name shall be replaced with the contents of the line. The <i>pax</i> utility shall immediately exit with a non-zero exit status if end-of-file is encountered when reading a response or if <b>/dev/tty</b> cannot be opened for reading and writing. |
| 26976 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26977 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26978 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26979 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26980 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26981 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26982 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26983 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26984 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26985 |                          | The results of extracting a hard link to a file that has been renamed during extraction are unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 26986 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26987 | <b>-k</b>                | Prevent the overwriting of existing files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 26988 | <b>-l</b>                | (The letter ell.) In <b>copy</b> mode, hard links shall be made between the source and destination file hierarchies whenever possible.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 26989 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26990 | <b>-L</b>                | If a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. The default behavior shall be to archive the symbolic link itself.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 26991 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26992 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26993 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26994 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26995 | <b>-n</b>                | Select the first archive member that matches each <i>pattern</i> operand. No more than one archive member shall be matched for each pattern (although members of type directory shall still match the file hierarchy rooted at that file).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 26996 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26997 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26998 | <b>-o</b> <i>options</i> | Provide information to the implementation to modify the algorithm for extracting or writing files. The value of <i>options</i> shall consist of one or more comma-separated keywords of the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 26999 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27000 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27001 |                          | <i>keyword</i> [[:]= <i>value</i> ][, <i>keyword</i> [[:]= <i>value</i> ], ...]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27002 |                          | Some keywords apply only to certain file formats, as indicated with each description. Use of keywords that are inapplicable to the file format being processed produces undefined results.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 27003 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27004 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27005 |                          | Keywords in the <i>options</i> argument shall be a string that would be a valid portable file name as described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.282, Portable File Name Character Set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 27006 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27007 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27008 | <b>Note:</b>             | Keywords are not expected to be file names, merely to follow the same character composition rules as portable file names.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 27009 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27010 |                          | Keywords can be preceded with white space. The <i>value</i> field shall consist of zero or more characters; within <i>value</i> , the application shall precede any literal comma with a backslash, which shall be ignored, but preserves the comma as part of <i>value</i> . A comma as the final character, or a comma followed solely by white space as the final characters, in <i>options</i> shall be ignored. Multiple <b>-o</b> options can be specified; if                                                                                                                                                                                                                                                                                                                                                                   |
| 27011 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27012 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27013 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27014 |                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

27015 keywords given to these multiple `-o` options conflict, the keywords and values  
 27016 appearing later in command line sequence shall take precedence and the earlier  
 27017 shall be silently ignored. The following keyword values of *options* shall be  
 27018 supported for the file formats as indicated:

27019 **delete=pattern**

27020 (Applicable only to the `-x pax` format.) When used in **write** or **copy** mode, *pax*  
 27021 shall omit from extended header records that it produces any keywords  
 27022 matching the string pattern. When used in **read** or **list** mode, *pax* shall ignore  
 27023 any keywords matching the string pattern in the extended header records. In  
 27024 both cases, matching shall be performed using the pattern matching notation  
 27025 described in Section 2.14.1 (on page 2274) and Section 2.14.2 (on page 2274).  
 27026 For example:

27027 `-o delete=security.*`

27028 would suppress security-related information. See **pax Extended Header** (on  
 27029 page 2923) for extended header record keyword usage.

27030 **exthdr.name=string**

27031 (Applicable only to the `-x pax` format.) This keyword allows user control over  
 27032 the name that is written into the **ustar** header blocks for the extended header  
 27033 produced under the circumstances described in **pax Header Block** (on page  
 27034 2922). The name shall be the contents of *string*, after the following character  
 27035 substitutions have been made:

| <i>string</i><br>Includes: | Replaced By:                                                                                                        |
|----------------------------|---------------------------------------------------------------------------------------------------------------------|
| %d                         | The directory name of the file, equivalent to the result of the <i>dirname</i> utility on the translated path name. |
| %f                         | The file name of the file, equivalent to the result of the <i>basename</i> utility on the translated path name.     |
| %%                         | A '%' character.                                                                                                    |

27043 Any other '%' characters in *string* produce undefined results.

27044 If no `-o exthdr.name=string` is specified, *pax* shall use the following default  
 27045 value:

27046 `%d/PaxHeaders/%f`

27047 **globexthdr.name=string**

27048 (Applicable only to the `-x pax` format.) When used in **write** or **copy** mode with  
 27049 the appropriate options, *pax* creates global extended header records with **ustar**  
 27050 header blocks that will be treated as regular files by previous versions of *pax*.  
 27051 This keyword allows user control over the name that is written into the **ustar**  
 27052 header blocks for global extended header records. The name shall be the  
 27053 contents of *string*, after the following character substitutions have been made:

| <i>string</i><br>Includes: | Replaced By:                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| %n                         | An integer that represents the sequence number of the global extended header record in the archive, starting at 1. |
| %%                         | A '%' character.                                                                                                   |

27054  
 27055  
 27056  
 27057  
 27058

27059 Any other ‘%’ characters in *string* produce undefined results.

27060 If no **-o globexthdr.name=string** is specified, *pax* shall use the following  
27061 default value:

27062 \$TMPDIR/GlobalHead.%n

27063 where \$TMPDIR represents the value of the *TMPDIR* environment variable. If  
27064 *TMPDIR* is not set, *pax* shall use **/tmp**.

27065 **invalid=action**

27066 (Applicable only to the **-x pax** format.) This keyword allows user control over  
27067 the action *pax* takes upon encountering values in an extended header record  
27068 that, in **read** or **copy** mode, are invalid in the destination hierarchy or, in **list**  
27069 mode, cannot be written in the codeset and current locale of the  
27070 implementation. The following are invalid values that shall be recognized by  
27071 *pax*:

- 27072 — In **read** or **copy** mode, a file name or link name that contains character  
27073 encodings invalid in the destination hierarchy. (For example, the name  
27074 may contain embedded NULs.)
- 27075 — In **read** or **copy** mode, a file name or link name that is longer than the  
27076 maximum allowed in the destination hierarchy (for either a path name  
27077 component or the entire path name).
- 27078 — In **list** mode, any character string value (file name, link name, user name,  
27079 and so on) that cannot be written in the codeset and current locale of the  
27080 implementation.

27081 The following mutually-exclusive values of the *action* argument are  
27082 supported:

27083 **bypass** In **read** or **copy** mode, *pax* shall bypass the file, causing no  
27084 change to the destination hierarchy. In **list** mode, *pax* shall write  
27085 all requested valid values for the file, but its method for writing  
27086 invalid values is unspecified.

27087 **rename** In **read** or **copy** mode, *pax* shall act as if the **-i** option were in  
27088 effect for each file with invalid file name or link name values,  
27089 allowing the user to provide a replacement name interactively.  
27090 In **list** mode, *pax* shall behave identically to the **bypass** action.

27091 **UTF-8** When used in **read**, **copy**, or **list** mode and a file name, link  
27092 name, owner name, or any other field in an extended header  
27093 record cannot be translated from the *pax* UTF-8 codeset format  
27094 to the codeset and current locale of the implementation, *pax*  
27095 shall use the actual UTF-8 encoding for the name.

27096 **write** In **read** or **copy** mode, *pax* shall write the file, translating or  
27097 truncating the name, regardless of whether this may overwrite  
27098 an existing file with a valid name. In **list** mode, *pax* shall behave  
27099 identically to the **bypass** action.

27100 If no **-o invalid=** option is specified, *pax* shall act as if **-o invalid=bypass** were  
27101 specified. Any overwriting of existing files that may be allowed by the  
27102 **-o invalid=** actions shall be subject to permission (**-p**) and modification time  
27103 (**-u**) restrictions, and shall be suppressed if the **-k** option is also specified.



27104 **linkdata** (Applicable only to the **-x pax** format.) In **write** mode, *pax* shall write the  
 27105 contents of a file to the archive even when that file is merely a hard link to a  
 27106 file whose contents have already been written to the archive.

27107 **listopt=format**

27108 This keyword specifies the output format of the table of contents produced  
 27109 when the **-v** option is specified in **list** mode. See **List Mode Format**  
 27110 **Specifications** (on page 2918). To avoid ambiguity, the **listopt=format** shall be  
 27111 the only or final **keyword=value** pair in a **-o** option-argument; all characters in  
 27112 the remainder of the option-argument shall be considered part of the format  
 27113 string. When multiple **-olistopt=format** options are specified, the format  
 27114 strings shall be considered a single, concatenated string, evaluated in  
 27115 command line order.

27116 **times**

27117 (Applicable only to the **-x pax** format.) When used in **write** or **copy** mode, *pax*  
 27118 shall include **atime**, **ctime**, and **mtime** extended header records for each file.  
 27119 See **pax Extended Header File Times** (on page 2926).

27120 In addition to these keywords, if the **-x pax** format is specified, any of the  
 27121 keywords and values defined in **pax Extended Header** (on page 2923), including  
 27122 implementation extensions, can be used in **-o** option-arguments, in either of two  
 27123 modes:

27124 **keyword=value**

27125 When used in **write** or **copy** mode, these keyword/value pairs shall be  
 27126 included at the beginning of the archive as **typeflag g** global extended header  
 27127 records. When used in **read** or **list** mode, these keyword/value pairs shall act  
 27128 as if they had been at the beginning of the archive as **typeflag g** global  
 27129 extended header records.

27130 **keyword:=value**

27131 When used in **write** or **copy** mode, these keyword/value pairs shall be  
 27132 included as records at the beginning of a **typeflag x** extended header for each  
 27133 file. (This is equivalent to the equal-sign form except that it creates no  
 27134 **typeflag g** global extended header records.) When used in **read** or **list** mode,  
 27135 these keyword/value pairs shall act as if they were included as records at the  
 27136 end of each extended header; thus, they shall override any global or file-  
 27137 specific extended header record keywords of the same names. For example, in  
 27138 the command:

```
27139 pax -r -o "
27140 gname:=mygroup,
27141 " <archive
```

27142 the group name will be forced to a new value for all files read from the  
 27143 archive.

27144 The precedences of **-o** keywords over various fields in the archive are described in  
 27145 **pax Extended Header Keyword Precedence** (on page 2925).

27146 **-p string**

27147 Specify one or more file characteristic options (privileges). The *string* option-  
 27148 argument shall be a string specifying file characteristics to be retained or discarded  
 27149 on extraction. The string shall consist of the specification characters **a**, **e**, **m**, **o**, and  
 27150 **p**. Other implementation-defined characters can be included. Multiple  
 27151 characteristics can be concatenated within the same string and multiple **-p** options  
 can be specified. The meaning of the specification characters are as follows:

- 27152           **a**   Do not preserve file access times.
- 27153           **e**   Preserve the user ID, group ID, file mode bits (see the Base Definitions volume  
27154           of IEEE Std. 1003.1-200x, Section 3.170, File Mode Bits), access time,  
27155           modification time, and any other implementation-defined file characteristics.
- 27156           **m**   Do not preserve file modification times.
- 27157           **o**   Preserve the user ID and group ID.
- 27158           **p**   Preserve the file mode bits. Other implementation-defined file mode attributes  
27159           may be preserved.
- 27160           In the preceding list, “preserve” indicates that an attribute stored in the archive  
27161           shall be given to the extracted file, subject to the permissions of the invoking  
27162           process. The access and modification times of the file shall be preserved unless  
27163           otherwise specified with the **-p** option or not stored in the archive. All attributes  
27164           that are not preserved shall be determined as part of the normal file creation action  
27165           (see Section 1.7.1.4 (on page 2209)).
- 27166           If neither the **e** nor the **o** specification character is specified, or the user ID and  
27167           group ID are not preserved for any reason, *pax* shall not set the S\_ISUID and  
27168           S\_ISGID bits of the file mode.
- 27169           If the preservation of any of these items fails for any reason, *pax* shall write a  
27170           diagnostic message to standard error. Failure to preserve these items shall affect  
27171           the final exit status, but shall not cause the extracted file to be deleted.
- 27172           If file characteristic letters in any of the *string* option-arguments are duplicated or  
27173           conflict with each other, the ones given last shall take precedence. For example, if  
27174           **-p eme** is specified, file modification times are preserved.
- 27175           **-s replstr**   Modify file or archive member names named by *pattern* or *file* operands according  
27176           to the substitution expression *replstr*, using the syntax of the *ed* utility. The  
27177           concepts of “address” and “line” are meaningless in the context of the *pax* utility,  
27178           and shall not be supplied. The format shall be:
- 27179           **-s /old/new/[gp]**
- 27180           where as in *ed*, *old* is a basic regular expression and *new* can contain an ampersand,  
27181           ‘\n’ (where *n* is a digit) backreferences, or subexpression matching. The *old* string  
27182           also shall be permitted to contain <newline> characters.
- 27183           Any non-null character can be used as a delimiter (‘/’ shown here). Multiple **-s**  
27184           expressions can be specified; the expressions shall be applied in the order  
27185           specified, terminating with the first successful substitution. The optional trailing  
27186           ‘g’ is as defined in the *ed* utility. The optional trailing ‘p’ shall cause successful  
27187           substitutions to be written to standard error. File or archive member names that  
27188           substitute to the empty string shall be ignored when reading and writing archives.
- 27189           **-t**   Cause the access times of the archived files to be the same as they were before  
27190           being read by *pax*.
- 27191           **-u**   Ignore files that are older (having a less recent file modification time) than a pre-  
27192           existing file or archive member with the same name. In **read** mode, an archive  
27193           member with the same name as a file in the file system shall be extracted if the  
27194           archive member is newer than the file. In **write** mode, an archive file member with  
27195           the same name as a file in the file system shall be superseded if the file is newer  
27196           than the archive member. If **-a** is also specified, this is accomplished by appending

27197 to the archive; otherwise, it is unspecified whether this is accomplished by actual  
 27198 replacement in the archive or by appending to the archive. In **copy** mode, the file in  
 27199 the destination hierarchy shall be replaced by the file in the source hierarchy or by  
 27200 a link to the file in the source hierarchy if the file in the source hierarchy is newer.

27201 **-v** In **list** mode, produce a verbose table of contents (see the STDOUT section).  
 27202 Otherwise, write archive member path names to standard error (see the STDERR  
 27203 section).

27204 **-x format** Specify the output archive format. The *pax* utility shall support the following  
 27205 formats:

27206 **cpio** The *cpio* interchange format; see the EXTENDED DESCRIPTION  
 27207 section. The default *blocksize* for this format for character special  
 27208 archive files shall be 5120. Implementations shall support all  
 27209 *blocksize* values less than or equal to 32 256 that are multiples of 512.

27210 **pax** The *pax* interchange format; see the EXTENDED DESCRIPTION  
 27211 section. The default *blocksize* for this format for character special  
 27212 archive files shall be 5120. Implementations shall support all  
 27213 *blocksize* values less than or equal to 32 256 that are multiples of 512.

27214 **ustar** The *tar* interchange format; see the EXTENDED DESCRIPTION  
 27215 section. The default *blocksize* for this format for character special  
 27216 archive files shall be 10 240. Implementations shall support all  
 27217 *blocksize* values less than or equal to 32 256 that are multiples of 512.

27218 Implementation-defined formats shall specify a default block size as well as any  
 27219 other block sizes supported for character special archive files.

27220 Any attempt to append to an archive file in a format different from the existing  
 27221 archive format shall cause *pax* to exit immediately with a non-zero exit status.

27222 In **copy** mode, if no **-x** format is specified, *pax* shall behave as if **-xpax** were  
 27223 specified.

27224 **-X** When traversing the file hierarchy specified by a path name, *pax* shall not descend  
 27225 into directories that have a different device ID (*st\_dev*; see the System Interfaces  
 27226 volume of IEEE Std. 1003.1-200x, *stat()*).

27227 The options that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u**, and **-v**)  
 27228 shall interact as follows. In **read** mode, the archive members shall be selected based on the user-  
 27229 specified *pattern* operands as modified by the **-c**, **-n**, and **-u** options. Then, any **-s** and **-i** options  
 27230 shall modify, in that order, the names of the selected files. The **-v** option shall write names  
 27231 resulting from these modifications.

27232 In **write** mode, the files shall be selected based on the user-specified path names as modified by  
 27233 the **-n** and **-u** options. Then, any **-s** and **-i** options shall modify, in that order, the names of  
 27234 these selected files. The **-v** option shall write names resulting from these modifications.

27235 If both the **-u** and **-n** options are specified, *pax* shall not consider a file selected unless it is newer  
 27236 than the file to which it is compared.

27237 **List Mode Format Specifications**

27238 In **list** mode with the `-o listopt=format` option, the *format* argument shall be applied for each  
 27239 selected file. The *pax* utility shall append a <newline> character to the **listopt** output for each  
 27240 selected file. The format argument shall be used as the *format* string described in the Base  
 27241 Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation, with the  
 27242 exceptions 1. through 5. defined in the EXTENDED DESCRIPTION section of *printf*, plus the  
 27243 following exceptions:

27244 6. The sequence (*keyword*) can occur before a format conversion specifier. The conversion  
 27245 argument is defined by the value of *keyword*. The implementation shall support the  
 27246 following keywords:

27247 — Any of the Field Name entries in Table 4-13 (on page 2927) and Table 4-15 (on page  
 27248 2930). The implementation may support the *cpio* keywords without the leading `c_` in  
 27249 addition to the form required by Table 4-16 (on page 2931).

27250 — Any keyword defined for the extended header in **pax Extended Header** (on page 2923).

27251 — Any keyword provided as an implementation-defined extension within the extended  
 27252 header defined in **pax Extended Header** (on page 2923).

27253 For example, the sequence "`%(charset)s`" is the string value of the name of the character  
 27254 set in the extended header.

27255 The result of the keyword conversion argument shall be the value from the applicable  
 27256 header field or extended header, without any trailing NULs.

27257 All keyword values used as conversion arguments shall be translated from the UTF-8  
 27258 encoding to the character set appropriate for the local file system, user database, and so on,  
 27259 as applicable.

27260 7. An additional conversion character, **T**, shall be used to specify time formats. The **T**  
 27261 conversion character can be preceded by the sequence (*keyword=subformat*), where *subformat*  
 27262 is a date format as defined by *date* operands. The default *keyword* shall be **mtime** and the  
 27263 default subformat shall be:

27264 `%b %e %H:%M %Y`

27265 8. An additional conversion character, **M**, shall be used to specify the file mode string as  
 27266 defined in *ls* Standard Output. If (*keyword*) is omitted, the **mode** keyword shall be used. For  
 27267 example, `%.1M` writes the single character corresponding to the <entry type> field of the *ls*  
 27268 `-l` command.

27269 9. An additional conversion character, **D**, shall be used to specify the device for block or  
 27270 special files, if applicable, in an implementation-defined format. If not applicable, and  
 27271 (*keyword*) is specified, then this conversion shall be equivalent to `%(keyword)u`. If not  
 27272 applicable, and (*keyword*) is omitted, then this conversion shall be equivalent to <space>.

27273 10. An additional conversion character, **F**, shall be used to specify a path name. The **F**  
 27274 conversion character can be preceded by a sequence of comma-separated keywords:

27275 `( keyword[ ,keyword] . . . )`

27276 The values for all the keywords that are non-null shall be concatenated together, each  
 27277 separated by a `'/'`. The default shall be **(path)** if the keyword **path** is defined; otherwise,  
 27278 the default shall be **(prefix,name)**.

27279 11. An additional conversion character, **L**, shall be used to specify a symbolic line expansion. If  
 27280 the current file is a symbolic link, then `%L` shall expand to:

27281            "%s -> %s", <value of keyword>, <contents of link>  
 27282            Otherwise, the %L conversion character shall be the equivalent of %F.

### 27283 OPERANDS

27284            The following operands shall be supported:

27285            *directory*    The destination directory path name for **copy** mode.  
 27286            *file*            A path name of a file to be copied or archived.  
 27287            *pattern*        A pattern matching one or more path names of archive members. A pattern must  
 27288                   be given in the name-generating notation of the pattern matching notation in  
 27289                   Section 2.14 (on page 2274), including the file name expansion rules in Section  
 27290                   2.14.3 (on page 2275). The default, if no *pattern* is specified, is to select all members  
 27291                   in the archive.

### 27292 STDIN

27293            In **write** mode, the standard input shall be used only if no *file* operands are specified. It shall be a  
 27294            text file containing a list of path names, one per line, without leading or trailing <blank>  
 27295            characters.

27296            In **list** and **read** modes, if **-f** is not specified, the standard input shall be an archive file.

27297            Otherwise, the standard input shall not be used.

### 27298 INPUT FILES

27299            The input file named by the *archive* option-argument, or standard input when the archive is read  
 27300            from there, shall be a file formatted according to one of the specifications in the EXTENDED  
 27301            DESCRIPTION section or some other implementation-defined format.

27302            The file **/dev/tty** shall be used to write prompts and read responses.

### 27303 ENVIRONMENT VARIABLES

27304            The following environment variables shall affect the execution of *pax*:

27305            *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 27306                   If *LANG* is unset or null, the corresponding value from the implementation-  
 27307                   defined default locale shall be used. If any of the internationalization variables  
 27308                   contains an invalid setting, the utility shall behave as if none of the variables had  
 27309                   been defined.

27310            *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 27311                   internationalization variables.

#### 27312 *LC\_COLLATE*

27313                   Determine the locale for the behavior of ranges, equivalence classes and multi-  
 27314                   character collating elements used in the pattern matching expressions for the  
 27315                   *pattern* operand, the basic regular expression for the **-s** option, and the extended  
 27316                   regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
 27317                   category.

27318            *LC\_CTYPE*    Determine the locale for the interpretation of sequences of bytes of text data as  
 27319                   characters (for example, single-byte as opposed to multi-byte characters in  
 27320                   arguments and input files), the behavior of character classes used in the extended  
 27321                   regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
 27322                   category, and pattern matching.

#### 27323 *LC\_MESSAGES*

27324                   Determine the locale for the processing of affirmative responses that should be

- 27325 used to affect the format and contents of diagnostic messages written to standard  
27326 error.
- 27327 **LC\_TIME** Determine the format and contents of date and time strings when the **-v** option is  
27328 specified.
- 27329 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 27330 **TMPDIR** Determine the path name that provides part of the default global extended header  
27331 record file, as described for the **-o globexthdr=** keyword as described in the  
27332 OPTIONS section.
- 27333 **ASYNCHRONOUS EVENTS**
- 27334 Default.
- 27335 **STDOUT**
- 27336 In **write** mode, if **-f** is not specified, the standard output shall be the archive formatted  
27337 according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
27338 implementation-defined format (see **-x format**).
- 27339 In **list** mode, when the **-olistopt=format** has been specified, the selected archive members shall  
27340 be written to standard output using the format described under **List Mode Format**  
27341 **Specifications** (on page 2918). In **list** mode without the **-olistopt=format** option, the table of  
27342 contents of the selected archive members shall be written to standard output using the  
27343 following format:
- 27344 "%s\n", <path name>
- 27345 If the **-v** option is specified in **list** mode, the table of contents of the selected archive members  
27346 shall be written to standard output using the following formats.
- 27347 For path names representing hard links to previous members of the archive:
- 27348 "%sΔ=Δ%s\n", <ls -l listing>, <linkname>
- 27349 For all other path names:
- 27350 "%s\n", <ls -l listing>
- 27351 where <ls -l listing> shall be the format specified by the *ls* utility with the **-l** option. When  
27352 writing path names in this format, it is unspecified what is written for fields for which the  
27353 underlying archive format does not have the correct information, although the correct number of  
27354 <blank> character-separated fields shall be written.
- 27355 In **list** mode, standard output shall not be buffered more than a line at a time.
- 27356 **STDERR**
- 27357 If **-v** is specified in **read**, **write**, or **copy** modes, *pax* shall write the path names it processes to the  
27358 standard error output using the following format:
- 27359 "%s\n", <path name>
- 27360 These path names shall be written as soon as processing is begun on the file or archive member,  
27361 and shall be flushed to standard error. The trailing <newline> character, which shall not be  
27362 buffered, is written when the file has been read or written.
- 27363 If the **-s** option is specified, and the replacement string has a trailing 'p', substitutions shall be  
27364 written to standard error in the following format:
- 27365 "%sΔ>>Δ%s\n", <original path name>, <new path name>

27366 In all operating modes of *pax*, optional messages of unspecified format concerning the input  
 27367 archive format and volume number, the number of files, blocks, volumes, and media parts as  
 27368 well as other diagnostic messages may be written to standard error.

27369 In all formats, for both standard output and standard error, it is unspecified how non-printable  
 27370 characters in path names or link names are written.

27371 When *pax* is in **read** mode or **list** mode, using the **-xpax** archive format, and a file name, link  
 27372 name, owner name, or any other field in an extended header record cannot be translated from  
 27373 the *pax* UTF-8 codeset format to the codeset and current locale of the implementation, *pax* shall  
 27374 write a diagnostic message to standard error, shall process the file as described for the **-o**  
 27375 **invalid=option**, and then shall process the next file in the archive.

#### 27376 OUTPUT FILES

27377 In **read** mode, the extracted output files shall be of the archived file type. In **copy** mode, the  
 27378 copied output files shall be the type of the file being copied. In either mode, existing files in the  
 27379 destination hierarchy shall be overwritten only when all permission (**-p**), modification time (**-u**),  
 27380 and invalid-value (**-oinvalid=**) tests allow it.

27381 In **write** mode, the output file named by the **-f** option-argument shall be a file formatted  
 27382 according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
 27383 implementation-defined format.

#### 27384 EXTENDED DESCRIPTION

##### 27385 **pax Interchange Format**

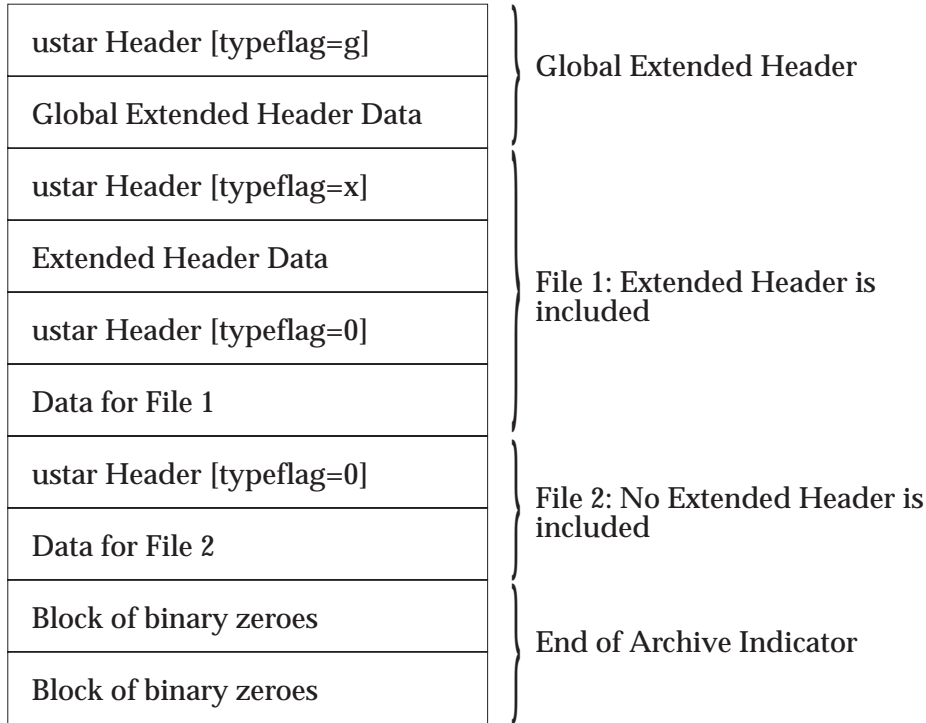
27386 A *pax* archive tape or file produced in the **-xpax** format shall contain a series of blocks. The  
 27387 physical layout of the archive shall be identical to the **ustar** format described in **ustar**  
 27388 **Interchange Format** (on page 2926). Each file archived shall be represented by the following  
 27389 sequence:

- 27390 • An optional header block with extended header records. This header block is of the form  
 27391 described in **pax Header Block** (on page 2922), with a *typflag* value of **x** or **g**. The extended  
 27392 header records, described in **pax Extended Header** (on page 2923), are included as the data  
 27393 for this header block.
- 27394 • A header block that describes the file. Any fields in the preceding optional extended header  
 27395 override the associated fields in this header block for this file.
- 27396 • Zero or more blocks that contain the contents of the file.

27397 At the end of the archive file there shall be two 512-byte blocks filled with binary zeroes,  
 27398 interpreted as an end-of-archive indicator.

27399 A schematic of an example archive with global extended header records and two actual files is  
 27400 shown in Figure 4-1 (on page 2922). In the example, the second file in the archive has no  
 27401 extended header preceding it, presumably because it has no need for extended attributes.

27402



27403

Figure 4-1 pax Format Archive Example

27404

**pax Header Block**

27405  
27406

The *pax* header block shall be identical to the **ustar** header block described in **ustar Interchange Format** (on page 2926), except that two additional *typeflag* values are defined:

27407  
27408  
27409

**x** Represents extended header records for the following file in the archive (which shall have its own **ustar** header block). The format of these extended header records shall be as described in **pax Extended Header** (on page 2923).

27410  
27411  
27412  
27413  
27414  
27415

**g** Represents global extended header records for the following files in the archive. The format of these extended header records shall be as described in **pax Extended Header** (on page 2923). Each value shall affect all subsequent files that do not override that value in their own extended header record and until another global extended header record is reached that provides another value for the same field. The *typeflag g* global headers should not be used with interchange media that could suffer partial data loss in transporting the archive.

27416  
27417  
27418  
27419  
27420  
27421

For both of these types, the *size* field shall be the size of the extended header records in octets. The other fields in the header block are not meaningful to this version of the *pax* utility. However, if this archive is read by a *pax* utility conforming to a previous version of IEEE Std. 1003.1-200x, the header block fields are used to create a regular file that contains the extended header records as data. Therefore, header block field values should be selected to provide reasonable file access to this regular file.

27422  
27423  
27424

A further difference from the **ustar** header block is that data blocks for files of *typeflag 1* (the digit one) (hard link) may be included, which means that the *size* field may be greater than zero. Archives created by **pax -o linkdata** shall include these data blocks with the hard links.



27425 **pax Extended Header**

27426 A *pax* extended header contains values that are inappropriate for the **ustar** header block because  
 27427 of limitations in that format: fields requiring a character encoding other than that described in  
 27428 the ISO/IEC 646:1991 standard, fields representing file attributes not described in the **ustar**  
 27429 header, and fields whose format or length do not fit the requirements of the **ustar** header. The  
 27430 values in an extended header add attributes to the following file (or files; see the description of  
 27431 the *typeflag g* header block) or override values in the following header block(s), as indicated in  
 27432 the following list of keywords.

27433 An extended header shall consist of one or more records, each constructed as follows:

27434 "%d %s=%s\n", <length>, <keyword>, <value>

27435 The extended header records shall be encoded according to the ISO/IEC 10646-1:1993 standard  
 27436 (UTF-8). The <length> field, <blank> character, equals sign, and <newline> character shown  
 27437 shall be limited to the portable character set, as encoded in UTF-8. The <keyword> and <value>  
 27438 fields can be any UTF-8 characters. The <length> field shall be the decimal length of the extended  
 27439 header record in octets, including the trailing <newline> character.

27440 The <keyword> field shall be one of the entries from the following list or a keyword provided as  
 27441 an implementation extension. Keywords consisting entirely of lowercase letters, digits, and  
 27442 periods are reserved for future standardization. A keyword shall not include an equals sign. (In  
 27443 the following list, the notations “file(s)” or “block(s)” is used to acknowledge that a keyword  
 27444 affects the following single file after a *typeflag x* extended header, but possibly multiple files after  
 27445 *typeflag g*. Any requirements in the list for *pax* to include a record when in **write** or **copy** mode  
 27446 shall apply only when such a record has not already been provided through the use of the **-o**  
 27447 option. When used in **copy** mode, *pax* shall behave as if an archive had been created with  
 27448 applicable extended header records and then extracted.)

27449 **atime** The file access time for the following file(s), equivalent to the value of the *st\_atime*  
 27450 member of the **stat** structure for a file, as described by the *stat()* function. The  
 27451 access time shall be restored if the process has the appropriate privilege required  
 27452 to do so. The format of the <value> shall be as described in **pax Extended Header**  
 27453 **File Times** (on page 2926).

27454 **charset** The name of the character set used to encode the data in the following file(s). The  
 27455 entries in the following table are defined to refer to known standards; additional  
 27456 names may be agreed on between the originator and recipient.

| <value>                 | Formal Standard               |
|-------------------------|-------------------------------|
| ISO-IRΔ646Δ1990         | ISO/IEC 646: 1990             |
| ISO-IRΔ8859Δ1Δ1987      | ISO/IEC 8859-1: 1987          |
| ISO-IRΔ8859Δ2Δ1987      | ISO/IEC 8859-2: 1987          |
| ISO-IRΔ10646Δ1993       | ISO/IEC 10646: 1993           |
| ISO-IRΔ10646Δ1993ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY                  | None.                         |

27464 The encoding is included in an extended header for information only; when *pax* is  
 27465 used as described in IEEE Std. 1003.1-200x, it shall not translate the file data into  
 27466 any other encoding. The **BINARY** entry indicates unencoded binary data.

27467 When used in **write** or **copy** mode, it is implementation-defined whether *pax*  
 27468 includes a **charset** extended header record for a file.

27469 **comment** A series of characters used as a comment. All characters in the <value> field shall  
 27470 be ignored by *pax*.

|       |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27471 | <b>ctime</b>        | The file creation time for the following file(s), equivalent to the value of the <i>st_ctime</i> member of the <b>stat</b> structure for a file, as described by the <i>stat()</i> function. The creation time shall be restored if the process has the appropriate privilege required to do so. The format of the <value> shall be as described in <b>pax Extended Header File Times</b> (on page 2926).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 27472 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27473 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27474 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27475 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27476 | <b>gid</b>          | The group ID of the group that owns the file, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the <i>gid</i> field in the following header block(s). When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <i>gid</i> extended header record for each file whose group ID is greater than 2 097 151 (octal 7 777 777).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27477 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27478 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27479 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27480 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27481 | <b>gname</b>        | The group of the file(s), formatted as a group name in the group database. This record shall override the <i>gid</i> and <i>gname</i> fields in the following header block(s), and any <i>gid</i> extended header record. When used in <b>read</b> , <b>copy</b> , or <b>list</b> mode, <i>pax</i> shall translate the name from the UTF-8 encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the UTF-8 characters cannot be translated, and if the <b>-oinvalid=UTF-8</b> option is not specified, the results are implementation-defined. When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <b>gname</b> extended header record for each file whose group name cannot be represented entirely with the letters and digits of the portable character set.                                                                   |
| 27482 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27483 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27484 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27485 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27486 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27487 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27488 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27489 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27490 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27491 | <b>linkpath</b>     | The path name of a link being created to another file, of any type, previously archived. This record shall override the <i>linkname</i> field in the following <b>ustar</b> header block(s). The following <b>ustar</b> header block shall determine the type of link created. If <i>typeflag</i> of the following header block is 1, it shall be a hard link. If <i>typeflag</i> is 2, it shall be a symbolic link and the <b>linkpath</b> value shall be the contents of the symbolic link. The <i>pax</i> utility shall translate the name of the link (contents of the symbolic link) from the UTF-8 encoding to the character set appropriate for the local file system. When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <b>linkpath</b> extended header record for each link whose path name cannot be represented entirely with the members of the portable character set other than NUL. |
| 27492 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27493 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27494 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27495 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27496 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27497 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27498 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27499 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27500 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27501 | <b>mtime</b>        | The file modification time of the following file(s), equivalent to the value of the <i>st_mtime</i> member of the <b>stat</b> structure for a file, as described in the <i>stat()</i> function. This record shall override the <i>mtime</i> field in the following header block(s). The modification time shall be restored if the process has the appropriate privilege required to do so. The format of the <value> shall be as described in <b>pax Extended Header File Times</b> (on page 2926).                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27502 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27503 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27504 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27505 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27506 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27507 | <b>path</b>         | The path name of the following file(s). This record shall override the <i>name</i> and <i>prefix</i> fields in the following header block(s). The <i>pax</i> utility shall translate the path name of the file from the UTF-8 encoding to the character set appropriate for the local file system.<br><br>When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <i>path</i> extended header record for each file whose path name cannot be represented entirely with the members of the portable character set other than NUL.                                                                                                                                                                                                                                                                                                                                                                         |
| 27508 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27509 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27510 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27511 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27512 | <b>realtime.any</b> | The keywords prefixed by “realtime.” are reserved for future standardization.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 27513 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27514 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27515 | <b>security.any</b> | The keywords prefixed by “security.” are reserved for future standardization.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 27516 | <b>size</b>         | The size of the file in octets, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the <i>size</i> field in the following header block(s). When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 27517 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27518 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27518 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

- 27519 *size* extended header record for each file with a size value greater than 8 589 934 591  
27520 (octal 7 777 777 777).
- 27521 **uid** The user ID of the file owner, expressed as a decimal number using digits from the  
27522 ISO/IEC 646:1991 standard. This record shall override the *uid* field in the  
27523 following header block(s). When used in **write** or **copy** mode, *pax* shall include a  
27524 *uid* extended header record for each file whose owner ID is greater than 2 097 151  
27525 (octal 7 777 777).
- 27526 **uname** The owner of the following file(s), formatted as a user name in the user database.  
27527 This record shall override the *uid* and *uname* fields in the following header block(s),  
27528 and any *uid* extended header record. When used in **read**, **copy**, or **list** mode, *pax*  
27529 shall translate the name from the UTF-8 encoding in the header record to the  
27530 character set appropriate for the user database on the receiving system. If any of  
27531 the UTF-8 characters cannot be translated, and if the **-oinvalid=** UTF-8 option is  
27532 not specified, the results are implementation-defined. When used in **write** or **copy**  
27533 mode, *pax* shall include a **uname** extended header record for each file whose user  
27534 name cannot be represented entirely with the letters and digits of the portable  
27535 character set.
- 27536 If the *<value>* field is zero length, it shall delete any header block field, previously entered  
27537 extended header value, or global extended header value of the same name.
- 27538 If a keyword in an extended header record (or in a **-o** option-argument) overrides or deletes a  
27539 corresponding field in the **ustar** header block, *pax* shall ignore the contents of that header block  
27540 field.
- 27541 Unlike the **ustar** header block fields, NULs shall not delimit *<value>*s; all characters within the  
27542 *<value>* field shall be considered data for the field. None of the length limitations of the **ustar**  
27543 header block fields in Table 4-13 (on page 2927) shall apply to the extended header records.
- 27544 **pax Extended Header Keyword Precedence**
- 27545 This section describes the precedence in which the various header records and fields and  
27546 command line options are selected to apply to a file in the archive. When *pax* is used in **read** or  
27547 **list** modes, it shall determine a file attribute in the following sequence:
- 27548 1. If **-odelete=keyword-prefix** is used, the affected attributes shall be determined from step 7.,  
27549 if applicable, or ignored otherwise.
  - 27550 2. If **-okeyword:=** is used, the affected attributes shall be ignored.
  - 27551 3. If **-okeyword:=value** is used, the affected attribute shall be assigned the value.
  - 27552 4. If there is a *typeflag* **x** extended header record, the affected attribute shall be assigned the  
27553 *<value>*. When extended header records conflict, the last one given in the header shall take  
27554 precedence.
  - 27555 5. If **-okeyword=value** is used, the affected attribute shall be assigned the value.
  - 27556 6. If there is a *typeflag* **g** global extended header record, the affected attribute shall be  
27557 assigned the *<value>*. When global extended header records conflict, the last one given in  
27558 the global header shall take precedence.
  - 27559 7. Otherwise, the attribute shall be determined from the **ustar** header block.

27560 **pax Extended Header File Times**27561 **Notes to Reviewers**27562 *This section with side shading will not appear in the final copy. - Ed.*

27563 D3, XCU, ERN 158 proposes new wording for the first half of the following paragraph: "pax shall  
 27564 write an mtime record for each file in write or copy modes if the file's modification time cannot  
 27565 be represented exactly in the ustar header block described in ustar Interchange Format. This can  
 27566 occur if the time is out of ustar range, or if the file system of the underlying implementation  
 27567 supports non-integer time granularities and the time is not an integer."

27568 The *pax* utility shall write **atime** and **ctime** records for each file in **write** or **copy** modes only if  
 27569 the **-otimes** option is specified; *pax* shall write a **mtime** record for each file in **write** or **copy**  
 27570 modes if the file system of the underlying implementation supports time granularities smaller  
 27571 than that required by the **ustar** header block described in **ustar Interchange Format**. All of these  
 27572 time records shall be formatted as a decimal representation of the time in seconds since the  
 27573 Epoch. If a period ( ' . ' ) decimal point character is present, the digits to the right of the point  
 27574 shall represent the units of a subsecond timing granularity, where the first digit is tenths of a  
 27575 second and each subsequent digit is a tenth of the previous digit. Implementations may ignore  
 27576 any portion of the subsecond digits for which they do not support the necessary timing  
 27577 granularity; they shall not perform any rounding operation.

27578 **Notes to Reviewers**27579 *This section with side shading will not appear in the final copy. - Ed.*

27580 D3, XCU, ERN 173, proposes new text for the previous sentence because a pax implementation  
 27581 on a single platform should not be allowed to lose information when it writes an extended  
 27582 header time and then reads it back in again: "In read or copy mode, the pax utility shall truncate  
 27583 the time of a file to the greatest value that is not greater than the input header file time. In write  
 27584 or copy mode, the pax utility shall output a time exactly if it can be represented exactly as a  
 27585 decimal number, and otherwise shall generate only enough digits so that the same time shall be  
 27586 recovered if the file is extracted on a system whose underlying implementation supports the  
 27587 same time granularity."

27588 **ustar Interchange Format**

27589 A **ustar** archive tape or file shall contain a series of blocks. Each block shall be a fixed-size block  
 27590 of 512 octets (see below). Although this format may be thought of as being stored on 9-track  
 27591 industry-standard 12.7mm (0.5in) magnetic tape, other types of transportable media are not  
 27592 excluded. Each file archived shall be represented by a header block that describes the file,  
 27593 followed by zero or more blocks that give the contents of the file. At the end of the archive file  
 27594 there shall be two 512-octet blocks filled with binary zeros, interpreted as an end-of-archive  
 27595 indicator.

27596 The blocks may be grouped for physical I/O operations, as described under the **-blocksize** and  
 27597 **-x ustar** options. Each group of blocks may be written with a single operation equivalent to the  
 27598 *write()* function. On magnetic tape, the result of this write shall be a single tape record. The last  
 27599 group of blocks always shall be at the full size, so blocks after the two zero blocks may contain  
 27600 undefined data.

27601 The header block shall be structured as shown in the following table. All lengths and offsets are  
 27602 in decimal.

27603 **Table 4-13** ustar Header Block

27604  
27605  
27606  
27607  
27608  
27609  
27610  
27611  
27612  
27613  
27614  
27615  
27616  
27617  
27618  
27619  
27620  
27621

| Field Name      | Octet Offset | Length (in Octets) |
|-----------------|--------------|--------------------|
| <i>name</i>     | 0            | 100                |
| <i>mode</i>     | 100          | 8                  |
| <i>uid</i>      | 108          | 8                  |
| <i>gid</i>      | 116          | 8                  |
| <i>size</i>     | 124          | 12                 |
| <i>mtime</i>    | 136          | 12                 |
| <i>chksum</i>   | 148          | 8                  |
| <i>typeflag</i> | 156          | 1                  |
| <i>linkname</i> | 157          | 100                |
| <i>magic</i>    | 257          | 6                  |
| <i>version</i>  | 263          | 2                  |
| <i>uname</i>    | 265          | 32                 |
| <i>gname</i>    | 297          | 32                 |
| <i>devmajor</i> | 329          | 8                  |
| <i>devminor</i> | 337          | 8                  |
| <i>prefix</i>   | 345          | 155                |

27622  
27623  
27624  
27625  
27626  
27627  
27628

All characters in the header block shall be represented in the coded character set of the ISO/IEC 646:1991 standard. For maximum portability between implementations, names should be selected from characters represented by the portable file name character set as octets with the most significant bit zero. If an implementation supports the use of characters outside of slash and the portable file name character set in names for files, users, and groups, one or more implementation-defined encodings of these characters shall be provided for interchange purposes.

27629  
27630  
27631  
27632  
27633

However, the *pax* utility shall never create file names on the local system that cannot be accessed via the procedures described in IEEE Std. 1003.1-200x. If a file name is found on the medium that would create an invalid file name, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

27634  
27635

Each field within the header block is contiguous; that is, there is no padding used. Each character on the archive medium shall be stored contiguously.

27636  
27637  
27638  
27639  
27640  
27641

The fields *magic*, *uname*, and *gname* are character strings each terminated by a NUL character. The fields *name*, *linkname*, and *prefix* are NUL-terminated character strings except when all characters in the array contain non-NUL characters including the last character. The *version* field is two octets containing the characters "00" (zero-zero). The *typeflag* contains a single character. All other fields are leading zero-filled octal numbers using digits from the ISO/IEC 646:1991 standard IRV. Each numeric field is terminated by one or more <space> or NUL characters.

27642  
27643  
27644  
27645  
27646  
27647  
27648

The *name* and the *prefix* fields shall produce the path name of the file. A new path name shall be formed, if *prefix* is not an empty string (its first character is not NUL), by concatenating *prefix* (up to the first NUL character), a slash character, and *name*; otherwise, *name* is used alone. In either case, *name* is terminated at the first NUL character. If *prefix* begins with a NUL character, it shall be ignored. In this manner, path names of at most 256 characters can be supported. If a path name does not fit in the space provided, *pax* shall notify the user of the error, and shall not store any part of the file—header or data—on the medium.

27649  
27650  
27651

The *linkname* field, described below, shall not use the *prefix* to produce a path name. As such, a *linkname* is limited to 100 characters. If the name does not fit in the space provided, *pax* shall notify the user of the error, and shall not attempt to store the link on the medium.

27652 The *mode* field provides 12 bits encoded in the ISO/IEC 646:1991 standard octal digit  
27653 representation. The encoded bits shall represent the following values:

27654 **Table 4-14** ustar *mode* Field

| 27655 | Bit Value | IEEE Std. 1003.1-200x Bit | Description                                     |
|-------|-----------|---------------------------|-------------------------------------------------|
| 27656 | 04 000    | S_ISUID                   | Set UID on execution.                           |
| 27657 | 02 000    | S_ISGID                   | Set GID on execution.                           |
| 27658 | 01 000    | <reserved>                | Reserved for future standardization.            |
| 27659 | 00 400    | S_IRUSR                   | Read permission for file owner class.           |
| 27660 | 00 200    | S_IWUSR                   | Write permission for file owner class.          |
| 27661 | 00 100    | S_IXUSR                   | Execute/search permission for file owner class. |
| 27662 | 00 040    | S_IRGRP                   | Read permission for file group class.           |
| 27663 | 00 020    | S_IWGRP                   | Write permission for file group class.          |
| 27664 | 00 010    | S_IXGRP                   | Execute/search permission for file group class. |
| 27665 | 00 004    | S_IROTH                   | Read permission for file other class.           |
| 27666 | 00 002    | S_IWOTH                   | Write permission for file other class.          |
| 27667 | 00 001    | S_IXOTH                   | Execute/search permission for file other class. |

27668 When appropriate privilege is required to set one of these mode bits, and the user restoring the  
27669 files from the archive does not have the appropriate privilege, the mode bits for which the user  
27670 does not have appropriate privilege shall be ignored. Some of the mode bits in the archive  
27671 format are not mentioned elsewhere in this volume of IEEE Std. 1003.1-200x. If the  
27672 implementation does not support those bits, they may be ignored.

27673 The *uid* and *gid* fields are the user and group ID of the owner and group of the file, respectively.

27674 The *size* field is the size of the file in octets. If the *typeflag* field is set to specify a file to be of type  
27675 1 (a link) or 2 (reserved for symbolic links), the *size* field shall be specified as zero. If the *typeflag*  
27676 field is set to specify a file of type 5 (directory), the *size* field shall be interpreted as described  
27677 under the definition of that record type. No data blocks are stored for types 1, 2, or 5. If the  
27678 *typeflag* field is set to 3 (character special file), 4 (block special file), or 6 (FIFO), the meaning of  
27679 the *size* field is unspecified by this volume of IEEE Std. 1003.1-200x, and no data blocks shall be  
27680 stored on the medium. Additionally, for type 6, the *size* field shall be ignored when reading. If  
27681 the *typeflag* field is set to any other value, the number of blocks written following the header  
27682 shall be  $(size+511)/512$ , ignoring any fraction in the result of the division.

27683 The *mtime* field shall be the modification time of the file at the time it was archived. It is the  
27684 ISO/IEC 646:1991 standard representation of the octal value of the modification time obtained  
27685 from the *stat()* function.

27686 The *chksum* field shall be the ISO/IEC 646:1991 standard IRV representation of the octal value of  
27687 the simple sum of all octets in the header block. Each octet in the header shall be treated as an  
27688 unsigned value. These values shall be added to an unsigned integer, initialized to zero, the  
27689 precision of which is not less than 17 bits. When calculating the checksum, the *chksum* field is  
27690 treated as if it were all spaces.

27691 The *typeflag* field specifies the type of file archived. If a particular implementation does not  
27692 recognize the type, or the user does not have appropriate privilege to create that type, the file  
27693 shall be extracted as if it were a regular file if the file type is defined to have a meaning for the  
27694 *size* field that could cause data blocks to be written on the medium (see the previous description  
27695 for *size*). If conversion to a regular file occurs, the *pax* utility shall produce an error indicating  
27696 that the conversion took place. All of the *typeflag* fields shall be coded in the ISO/IEC 646:1991  
27697 standard IRV:

|       |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27698 | 0    | Represents a regular file. For backward compatibility, a <i>typeflag</i> value of binary zero (' <code>\0</code> ') should be recognized as meaning a regular file when extracting files from the archive. Archives written with this version of the archive file format create regular files with a <i>typeflag</i> value of the ISO/IEC 646:1991 standard IRV ' <code>0</code> '.                                                                                                                                                                                                                                                                                                                                                   |
| 27699 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27700 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27701 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27702 | 1    | Represents a file linked to another file, of any type, previously archived. Such files are identified by each file having the same device and file serial number. The linked-to name is specified in the <i>linkname</i> field with a NUL-character terminator if it is less than 100 octets in length.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 27703 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27704 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27705 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27706 | 2    | Represents a symbolic link. The contents of the symbolic link shall be stored in the <i>linkname</i> field.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 27707 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27708 | 3, 4 | Represent character special files and block special files respectively. In this case the <i>devmajor</i> and <i>devminor</i> fields shall contain information defining the device, the format of which is unspecified by this volume of IEEE Std. 1003.1-200x. Implementations may map the device specifications to their own local specification or may ignore the entry.                                                                                                                                                                                                                                                                                                                                                            |
| 27709 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27710 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27711 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27712 | 5    | Specifies a directory or subdirectory. On systems where disk allocation is performed on a directory basis, the <i>size</i> field shall contain the maximum number of octets (which may be rounded to the nearest disk block allocation unit) that the directory may hold. A <i>size</i> field of zero indicates no such limiting. Systems that do not support limiting in this manner should ignore the <i>size</i> field.                                                                                                                                                                                                                                                                                                            |
| 27713 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27714 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27715 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27716 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27717 | 6    | Specifies a FIFO special file. Note that the archiving of a FIFO file archives the existence of this file and not its contents.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27718 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27719 | 7    | Reserved to represent a file to which an implementation has associated some high-performance attribute. Implementations without such extensions should treat this file as a regular file (type 0).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 27720 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27721 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27722 | A-Z  | The letters ' <code>A</code> ' to ' <code>Z</code> ', inclusive, are reserved for custom implementations. All other values are reserved for future revisions of IEEE Std. 1003.1-200x.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 27723 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27724 |      | The <i>magic</i> field is the specification that this archive was output in this archive format. If this field contains <b>ustar</b> (the five characters from the ISO/IEC 646:1991 standard IRV shown followed by NUL), the <i>uname</i> and <i>gname</i> fields shall contain the ISO/IEC 646:1991 standard IRV representation of the owner and group of the file, respectively (truncated to fit, if necessary). When the file is restored by a privileged, protection-preserving version of the utility, the user and group databases shall be scanned for these names. If found, the user and group IDs contained within these files shall be used rather than the values contained within the <i>uid</i> and <i>gid</i> fields. |
| 27725 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27726 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27727 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27728 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27729 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27730 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27731 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27732 |      | <b>cpio Interchange Format</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 27733 |      | The octet-oriented <i>cpio</i> archive format shall be a series of entries, each comprising a header that describes the file, the name of the file, and then the contents of the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 27734 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27735 |      | An archive may be recorded as a series of fixed-size blocks of octets. This blocking shall be used only to make physical I/O more efficient. The last group of blocks shall be always at the full size.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 27736 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27737 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27738 |      | For the octet-oriented <i>cpio</i> archive format, the individual entry information shall be in the order indicated and described by the following table; see also the <code>&lt;cpio.h&gt;</code> header.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 27739 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

27740

Table 4-15 Octet-Oriented *cpio* Archive Entry

27741

27742

27743

27744

27745

27746

27747

27748

27749

27750

27751

27752

27753

27754

27755

27756

| Header Field Name    | Length (in Octets) | Interpreted as   |
|----------------------|--------------------|------------------|
| <i>c_magic</i>       | 6                  | Octal number     |
| <i>c_dev</i>         | 6                  | Octal number     |
| <i>c_ino</i>         | 6                  | Octal number     |
| <i>c_mode</i>        | 6                  | Octal number     |
| <i>c_uid</i>         | 6                  | Octal number     |
| <i>c_gid</i>         | 6                  | Octal number     |
| <i>c_nlink</i>       | 6                  | Octal number     |
| <i>c_rdev</i>        | 6                  | Octal number     |
| <i>c_mtime</i>       | 11                 | Octal number     |
| <i>c_namesize</i>    | 6                  | Octal number     |
| <i>c_filesize</i>    | 11                 | Octal number     |
| File Name Field Name | Length             | Interpreted as   |
| <i>c_name</i>        | <i>c_namesize</i>  | Path name string |
| File Data Field Name | Length             | Interpreted as   |
| <i>c_filedata</i>    | <i>c_filesize</i>  | Data             |

27757

**cpio Header**

27758

27759

27760

27761

27762

27763

For each file in the archive, a header as defined previously shall be written. The information in the header fields is written as streams of the ISO/IEC 646: 1991 standard characters interpreted as octal numbers. The octal numbers shall be extended to the necessary length by appending the ISO/IEC 646: 1991 standard IRV zeros at the most-significant-digit end of the number; the result is written to the most-significant digit of the stream of octets first. The fields shall be interpreted as follows:

27764

27765

*c\_magic* Identify the archive as being a transportable archive by containing the identifying value "070707".

27766

27767

27768

*c\_dev, c\_ino* Contains values that uniquely identify the file within the archive (that is, no files contain the same pair of *c\_dev* and *c\_ino* values unless they are links to the same file). The values shall be determined in an unspecified manner.

27769

*c\_mode* Contains the file type and access permissions as defined in the following table.



27770

Table 4-16 Values for *cpio c\_mode* Field

27771

27772

27773

27774

27775

27776

27777

27778

27779

27780

27781

27782

27783

27784

27785

27786

27787

27788

27789

27790

27791

27792

| File Permissions Name | Value    | Indicates              |
|-----------------------|----------|------------------------|
| C_IRUSR               | 000 400  | Read by owner          |
| C_IWUSR               | 000 200  | Write by owner         |
| C_IXUSR               | 000 100  | Execute by owner       |
| C_IRGRP               | 000 040  | Read by group          |
| C_IWGRP               | 000 020  | Write by group         |
| C_IXGRP               | 000 010  | Execute by group       |
| C_IROTH               | 000 004  | Read by others         |
| C_IWOTH               | 000 002  | Write by others        |
| C_IXOTH               | 000 001  | Execute by others      |
| C_ISUID               | 004 000  | Set <i>uid</i>         |
| C_ISGID               | 002 000  | Set <i>gid</i>         |
| C_ISVTX               | 001 000  | Reserved               |
| File Type Name        | Value    | Indicates              |
| C_ISDIR               | 040 000  | Directory              |
| C_ISFIFO              | 010 000  | FIFO                   |
| C_ISREG               | 0100 000 | Regular file           |
| C_ISBLK               | 060 000  | Block special file     |
| C_ISCHR               | 020 000  | Character special file |
| C_ISCTG               | 0110 000 | Reserved               |
| C_ISLNK               | 0120 000 | Reserved               |
| C_ISSOCK              | 0140 000 | Reserved               |

27793

27794

27795

27796

27797

Directories, FIFOs, and regular files shall be supported on a system conforming to this volume of IEEE Std. 1003.1-200x; additional values defined previously are reserved for compatibility with existing systems. Additional file types may be supported; however, such files should not be written to archives intended to be transported to other systems.

27798

*c\_uid*

Contains the user ID of the owner.

27799

*c\_gid*

Contains the group ID of the group.

27800

*c\_nlink*

Contains the number of links referencing the file at the time the archive was created.

27801

27802

*c\_rdev*

Contains implementation-defined information for character or block special files.

27803

*c\_mtime*

Contains the latest time of modification of the file at the time the archive was created.

27804

27805

*c\_namesize*

Contains the length of the path name, including the terminating NUL character.

27806

*c\_filesize*

Contains the length of the file in octets. This shall be the length of the data section following the header structure.

27807

|           |                                                                                                                    |
|-----------|--------------------------------------------------------------------------------------------------------------------|
| 27808     | <b>cpio File Name</b>                                                                                              |
| 27809     | The <i>c_name</i> field shall contain the path name of the file. The length of this field in octets is the         |
| 27810     | value of <i>c_namesize</i> .                                                                                       |
| 27811 MAN | If a file name is found on the medium that would create an invalid path name, it is                                |
| 27812     | implementation-defined whether the data from the file is stored on the file hierarchy and under                    |
| 27813     | what name it is stored.                                                                                            |
| 27814     | All characters shall be represented in the ISO/IEC 646:1991 standard IRV. For maximum                              |
| 27815     | portability between implementations, names should be selected from characters represented by                       |
| 27816     | the portable file name character set as octets with the most significant bit zero. If an                           |
| 27817     | implementation supports the use of characters outside the portable file name character set in                      |
| 27818     | names for files, users, and groups, one or more implementation-defined encodings of these                          |
| 27819     | characters shall be provided for interchange purposes. However, the <i>pax</i> utility shall never                 |
| 27820     | create file names on the local system that cannot be accessed via the procedures described                         |
| 27821     | previously in this volume of IEEE Std. 1003.1-200x. If a file name is found on the medium that                     |
| 27822     | would create an invalid file name, it is implementation-defined whether the data from the file is                  |
| 27823     | stored on the local file system and under what name it is stored. The <i>pax</i> utility may choose to             |
| 27824     | ignore these files as long as it produces an error indicating that the file is being ignored.                      |
| 27825     | <b>cpio File Data</b>                                                                                              |
| 27826     | Following <i>c_name</i> , there shall be <i>c_filesize</i> octets of data. Interpretation of such data occurs in a |
| 27827     | manner dependent on the file. If <i>c_filesize</i> is zero, no data shall be contained in <i>c_filedata</i> .      |
| 27828 MAN | When restoring from an archive:                                                                                    |
| 27829     | • If the user does not have the appropriate privilege to create a file of the specified type, <i>pax</i>           |
| 27830     | shall ignore the entry and write an error message to standard error.                                               |
| 27831     | • Only regular files have data to be restored. Presuming a regular file meets any selection                        |
| 27832     | criteria that might be imposed on the format-reading utility by the user, such data shall be                       |
| 27833     | restored.                                                                                                          |
| 27834     | • If a user does not have appropriate privilege to set a particular mode flag, the flag shall be                   |
| 27835     | ignored. Some of the mode flags in the archive format are not mentioned elsewhere in this                          |
| 27836     | volume of IEEE Std. 1003.1-200x. If the implementation does not support those flags, they                          |
| 27837     | may be ignored.                                                                                                    |
| 27838     |                                                                                                                    |
| 27839     | <b>cpio Special Entries</b>                                                                                        |
| 27840     | FIFO special files, directories, and the trailer shall be recorded with <i>c_filesize</i> equal to zero. For       |
| 27841     | other special files, <i>c_filesize</i> is unspecified by this volume of IEEE Std. 1003.1-200x. The header          |
| 27842     | for the next file entry in the archive shall be written directly after the last octet of the file entry            |
| 27843     | preceding it. A header denoting the file name <b>TRAILER!!!</b> indicates the end of the archive; the              |
| 27844     | contents of octets in the last block of the archive following such a header are undefined.                         |
| 27845     | <b>EXIT STATUS</b>                                                                                                 |
| 27846     | The following exit values shall be returned:                                                                       |
| 27847     | 0 All files were processed successfully.                                                                           |
| 27848     | >0 An error occurred.                                                                                              |

27849 **CONSEQUENCES OF ERRORS**

27850 If *pax* cannot create a file or a link when reading an archive or cannot find a file when writing an  
 27851 archive, or cannot preserve the user ID, group ID, or file mode when the **-p** option is specified, a  
 27852 diagnostic message shall be written to standard error and a non-zero exit status shall be  
 27853 returned, but processing shall continue. In the case where *pax* cannot create a link to a file, *pax*  
 27854 shall not, by default, create a second copy of the file.

27855 If the extraction of a file from an archive is prematurely terminated by a signal or error, *pax* may  
 27856 have only partially extracted the file or (if the **-n** option was not specified) may have extracted a  
 27857 file of the same name as that specified by the user, but which is not the file the user wanted.  
 27858 Additionally, the file modes of extracted directories may have additional bits from the S\_IRWXU  
 27859 mask set as well as incorrect modification and access times.

27860 **APPLICATION USAGE**

27861 The **-p** (privileges) option was invented to reconcile differences between historical *tar* and *cpio*  
 27862 implementations. In particular, the two utilities use **-m** in diametrically opposed ways. The **-p**  
 27863 option also provides a consistent means of extending the ways in which future file attributes can  
 27864 be addressed, such as for enhanced security systems or high-performance files. Although it may  
 27865 seem complex, there are really two modes that are most commonly used:

27866 **-p e** “Preserve everything”. This would be used by the historical superuser, someone with  
 27867 all the appropriate privileges, to preserve all aspects of the files as they are recorded in  
 27868 the archive. The **e** flag is the sum of **o** and **p**, and other implementation-defined  
 27869 attributes.

27870 **-p p** “Preserve” the file mode bits. This would be used by the user with regular privileges  
 27871 who wished to preserve aspects of the file other than the ownership. The file times are  
 27872 preserved by default, but two other flags are offered to disable these and use the time  
 27873 of extraction.

27874 The one path name per line format of standard input precludes path names containing  
 27875 <newline> characters. Although such path names violate the portable file name guidelines, they  
 27876 may exist and their presence may inhibit usage of *pax* within shell scripts. This problem is  
 27877 inherited from historical archive programs. The problem can be avoided by listing file name  
 27878 arguments on the command line instead of on standard input.

27879 It is almost certain that appropriate privileges are required for *pax* to accomplish parts of this  
 27880 volume of IEEE Std. 1003.1-200x. Specifically, creating files of type block special or character  
 27881 special, restoring file access times unless the files are owned by the user (the **-t** option), or  
 27882 preserving file owner, group, and mode (the **-p** option) all probably require appropriate  
 27883 privileges.

27884 In **read** mode, implementations are permitted to overwrite files when the archive has multiple  
 27885 members with the same name. This may fail if permissions on the first version of the file do not  
 27886 permit it to be overwritten.

27887 The **cpio** and **ustar** formats can only support files up to 8 gigabytes in size.

27888 **EXAMPLES**

27889 The following command:

```
27890 pax -w -f /dev/rmt/1m .
```

27891 copies the contents of the current directory to tape drive 1, medium density (assuming historical  
 27892 System V device naming procedures. The historical BSD device name would be **/dev/rmt9**).

27893 The following commands:

27894 `mkdir newdir`  
 27895 `pax -rw olddir newdir`

27896 copy the *olddir* directory hierarchy to *newdir*.

27897 `pax -r -s ',^//*usr//*,,' -f a.pax`

27898 reads the archive **a.pax**, with all files rooted in **/usr** in the archive extracted relative to the current  
 27899 directory.

27900 Using the option:

27901 `-o listopt="%M %(atime)T %(size)D %(name)s"`

27902 overrides the default output description in Standard Output and instead writes:

27903 `-rw-rw--- Jan 12 15:53 1492 /usr/foo/bar`

27904 Using the options:

27905 `-o listopt='%L\t%(size)D\n%.7' \`  
 27906 `-o listopt='(name)s\n%(ctime)T\n%T'`

27907 overrides the default output description in Standard Output and instead writes:

27908 `/usr/foo/bar -> /tmp 1492`  
 27909 `/usr/fo`  
 27910 `Jan 12 1991`  
 27911 `Jan 31 15:53`

27912 **RATIONALE**

27913 The *pax* utility was new, commissioned for the ISO POSIX-2:1993 standard. It represents a  
 27914 peaceful compromise between advocates of the historical *tar* and *cpio* utilities.

27915 A fundamental difference between *cpio* and *tar* was in the way directories were treated. The *cpio*  
 27916 utility did not treat directories differently from other files, and to select a directory and its  
 27917 contents required that each file in the hierarchy be explicitly specified. For *tar*, a directory  
 27918 matched every file in the file hierarchy it rooted.

27919 The *pax* utility offers both interfaces; by default, directories map into the file hierarchy they root.  
 27920 The **-d** option causes *pax* to skip any file not explicitly referenced, as *cpio* historically did. The *tar*  
 27921 **-style** behavior was chosen as the default because it was believed that this was the more  
 27922 common usage and because *tar* is the more commonly available interface, as it was historically  
 27923 provided on both System V and BSD implementations.

27924 The data interchange format specification in this volume of IEEE Std. 1003.1-200x requires that  
 27925 processes with “appropriate privileges” shall always restore the ownership and permissions of  
 27926 extracted files exactly as archived. If viewed from the historic equivalence between superuser  
 27927 and “appropriate privileges”, there are two problems with this requirement. First, users running  
 27928 as superusers may unknowingly set dangerous permissions on extracted files. Second, it is  
 27929 needlessly limiting, in that superusers cannot extract files and own them as superuser unless the  
 27930 archive was created by the superuser. (It should be noted that restoration of ownerships and  
 27931 permissions for the superuser, by default, is historical practice in *cpio*, but not in *tar*.) In order to  
 27932 avoid these two problems, the *pax* specification has an additional “privilege” mechanism, the **-p**  
 27933 option. Only a *pax* invocation with the privileges needed, and which has the **-p** option set using  
 27934 the **e** specification character, has the “appropriate privilege” to restore full ownership and  
 27935 permission information.

27936 Note also that this volume of IEEE Std. 1003.1-200x requires that the file ownership and access  
 27937 permissions shall be set, on extraction, in the same fashion as the *creat()* function when provided

- 27938 the mode stored in the archive. This means that the file creation mask of the user is applied to  
27939 the file permissions.
- 27940 Users should note that directories may be created by *pax* while extracting files with permissions  
27941 that are different from those that existed at the time the archive was created. When extracting  
27942 sensitive information into a directory hierarchy that no longer exists, users are encouraged to set  
27943 their file creation mask appropriately to protect these files during extraction.
- 27944 The table of contents output is written to standard output to facilitate pipeline processing.
- 27945 An early proposal had hard links displaying for all path names. This was removed because it  
27946 complicates the output of the case where `-v` is not specified and does not match historical *cpio*  
27947 usage. The hard-link information is available in the `-v` display.
- 27948 The archive formats inherited from the POSIX.1-1990 standard have certain restrictions that  
27949 have been brought along from historical usage. For example, there are restrictions on the length  
27950 of path names stored in the archive. When *pax* is used in `copy(-rw)` mode (copying directory  
27951 hierarchies), the ability to use extensions from the `-xpax` format overcomes these restrictions.
- 27952 The default *blocksize* value of 5 120 bytes for *cpio* was selected because it is one of the standard  
27953 block-size values for *cpio*, set when the `-B` option is specified. (The other default block-size value  
27954 for *cpio* is 512 bytes, and this was considered to be too small.) The default block value of 10 240  
27955 bytes for *tar* was selected because that is the standard block-size value for BSD *tar*. The  
27956 maximum block size of 32 256 bytes ( $2^{15}$ -512 bytes) is the largest multiple of 512 bytes that fits  
27957 into a signed 16-bit tape controller transfer register. There are known limitations in some  
27958 historical systems that would prevent larger blocks from being accepted. Historical values were  
27959 chosen to improve compatibility with historical scripts using *dd* or similar utilities to manipulate  
27960 archives. Also, default block sizes for any file type other than character special file has been  
27961 deleted from this volume of IEEE Std. 1003.1-200x as unimportant and not likely to affect the  
27962 structure of the resulting archive.
- 27963 Implementations are permitted to modify the block-size value based on the archive format or  
27964 the device to which the archive is being written. This is to provide implementations with the  
27965 opportunity to take advantage of special types of devices, and it should not be used without a  
27966 great deal of consideration as it almost certainly decreases archive portability.
- 27967 The intended use of the `-n` option was to permit extraction of one or more files from the archive  
27968 without processing the entire archive. This was viewed by the standard developers as offering  
27969 significant performance advantages over historical implementations. The `-n` option in early  
27970 proposals had three effects; the first was to cause special characters in patterns to not be treated  
27971 specially. The second was to cause only the first file that matched a pattern to be extracted. The  
27972 third was to cause *pax* to write a diagnostic message to standard error when no file was found  
27973 matching a specified pattern. Only the second behavior is retained by this volume of  
27974 IEEE Std. 1003.1-200x, for many reasons. First, it is in general not acceptable for a single option to  
27975 have multiple effects. Second, the ability to make pattern matching characters act as normal  
27976 characters is useful for parts of *pax* other than file extraction. Third, a finer degree of control over  
27977 the special characters is useful because users may wish to normalize only a single special  
27978 character in a single file name. Fourth, given a more general escape mechanism, the previous  
27979 behavior of the `-n` option can be easily obtained using the `-s` option or a *sed* script. Finally,  
27980 writing a diagnostic message when a pattern specified by the user is unmatched by any file is  
27981 useful behavior in all cases.
- 27982 In this version, the `-n` was removed from the `copy` mode synopsis of *pax*; it is inapplicable  
27983 because there are no pattern operands specified in this mode.
- 27984 There is another method than *pax* for copying subtrees in IEEE Std. 1003.1-200x described as part  
27985 of the *cp* utility. Both methods are historical practice: *cp* provides a simpler, more intuitive

27986 interface, while *pax* offers a finer granularity of control. Each provides additional functionality to  
27987 the other; in particular, *pax* maintains the hard-link structure of the hierarchy while *cp* does not.  
27988 It is the intention of the standard developers that the results be similar (using appropriate option  
27989 combinations in both utilities). The results are not required to be identical; there seemed  
27990 insufficient gain to applications to balance the difficulty of implementations having to guarantee  
27991 that the results would be exactly identical.

27992 A single archive may span more than one file. It is suggested that implementations provide  
27993 informative messages to the user on standard error whenever the archive file is changed.

27994 The **-d** option (do not create intermediate directories not listed in the archive) found in early  
27995 proposals was originally provided as a complement to the historic **-d** option of *cpio*. It has been  
27996 deleted.

27997 The **-s** option in early proposals specified a subset of the substitution command from the *ed*  
27998 utility. As there was no reason for only a subset to be supported, the **-s** option is now  
27999 compatible with the current *ed* specification. Since the delimiter can be any non-null character,  
28000 the following usage with single spaces is valid:

```
28001 pax -s " foo bar " ...
```

28002 The **-t** option (specify an implementation-defined identifier naming an input or output device)  
28003 found in early proposals has been deleted because it is not historical practice and is of limited  
28004 utility. In particular, historic versions of neither *cpio* nor *tar* had the concept of devices that were  
28005 not mapped into the file system; if the devices are mapped into the file system, the **-f** option is  
28006 sufficient.

28007 The default behavior of *pax* with regard to file modification times is the same as historical  
28008 implementations of *tar*. It is not the historical behavior of *cpio*.

28009 Because the **-i** option uses **/dev/tty**, utilities without a controlling terminal are not able to use  
28010 this option.

28011 The **-y** option, found in early proposals, has been deleted because a line containing a single  
28012 period for the **-i** option has equivalent functionality. The special lines for the **-i** option (a single  
28013 period and the empty line) are historical practice in *cpio*.

28014 In early drafts, an **-echarmap** option was included to increase portability of files between systems  
28015 using different coded character sets. This option was omitted because it was apparent that  
28016 consensus could not be formed for it. In this version, the use of UTF-8 should be an adequate  
28017 substitute.

28018 The **-k** option was added to address international concerns about the dangers involved in the  
28019 character set transformations of **-e** (if the target character set were different than the source, the  
28020 file names might be transformed into names matching existing files) and also was made more  
28021 general to protect files transferred between file systems with different {NAME\_MAX} values  
28022 (truncating a file name on a smaller system might also inadvertently overwrite existing files). As  
28023 stated, it prevents any overwriting, even if the target file is older than the source. This version  
28024 adds more granularity of options to solve this problem by introducing the **-oinvalid=** option—  
28025 specifically the UTF-8 action. (Note that an existing file that is named with a UTF-8 encoding is  
28026 still subject to overwriting in this case. The **-k** option closes that loophole.)

28027 Some of the file characteristics referenced in this volume of IEEE Std. 1003.1-200x might not be  
28028 supported by some archive formats. For example, neither the *tar* nor *cpio* formats contain the file  
28029 access time. For this reason, the **e** specification character has been provided, intended to cause all  
28030 file characteristics specified in the archive to be retained.

28031 It is required that extracted directories, by default, have their access and modification times and  
 28032 permissions set to the values specified in the archive. This has obvious problems in that the  
 28033 directories are almost certainly modified after being extracted and that directory permissions  
 28034 may not permit file creation. One possible solution is to create directories with the mode  
 28035 specified in the archive, as modified by the *umask* of the user, with sufficient permissions to  
 28036 allow file creation. After all files have been extracted, *pax* would then reset the access and  
 28037 modification times and permissions as necessary.

28038 The list-mode formatting description borrows heavily from the one defined by the *printf* utility.  
 28039 However, since there is no separate operand list to get conversion arguments, the format was  
 28040 extended to allow specifying the name of the conversion argument as part of the conversion  
 28041 specification.

28042 The **T** specifier allows time fields to be displayed in any of the date formats. Unlike the *ls* utility,  
 28043 *pax* does not adjust the format when the date is less than six months in the past. This makes  
 28044 parsing the output more predictable.

28045 The **D** specifier handles the ability to display the major/minor or file size, as with *ls*, by using  
 28046 **%-8(size)D**.

28047 The **L** specifier handles the *ls* display for symbolic links.

28048 Conversion specifiers were added to generate existing known types used for *ls*.

#### 28049 **pax Interchange Format**

28050 The new POSIX data interchange format was developed primarily to satisfy international  
 28051 concerns that the **ustar** and *cpio* formats did not provide for file, user, and group names encoded  
 28052 in characters outside a subset of the ISO/IEC 646:1991 standard. The standard developers  
 28053 realized that this new POSIX data interchange format should be very extensible because there  
 28054 were other requirements they foresaw in the near future:

- 28055 • Support international character encodings and locale information
- 28056 • Support security information (ACLs, and so on)
- 28057 • Support future file types, such as realtime or contiguous files
- 28058 • Include data areas for implementation use
- 28059 • Support systems with words larger than 32 bits and timers with subsecond granularity

28060 The following were not goals for this format because these are better handled by separate  
 28061 utilities or are inappropriate for a portable format:

- 28062 • Encryption
- 28063 • Compression
- 28064 • Data translation between locales and codesets
- 28065 • *inode* storage

28066 The format chosen to support the goals is an extension of the **ustar** format. Of the two formats  
 28067 previously available, only the **ustar** format was selected for extensions because:

- 28068 • It was easier to extend in an upward-compatible way. It offered version flags and header  
 28069 block type fields with room for future standardization. The *cpio* format, while possessing a  
 28070 more flexible file naming methodology, could not be extended without breaking some  
 28071 theoretical implementation or using a dummy file name that could be a legitimate file name.

28072       • Industry experience since the original “tar wars” fought in developing the ISO POSIX-1  
28073       standard has clearly been in favor of the **ustar** format, which is generally the default output  
28074       format selected for *pax* implementations on new systems.

28075       The new format was designed with one additional goal in mind: reasonable behavior when an  
28076       older *tar* or *pax* utility happened to read an archive. Since the POSIX.1-1990 standard mandated  
28077       that a “format-reading utility” had to treat unrecognized *typeflag* values as regular files, this  
28078       allowed the format to include all the extended information in a pseudo-regular file that preceded  
28079       each real file. An option is given that allows the archive creator to set up reasonable names for  
28080       these files on the older systems. Also, the normative text suggests that reasonable file access  
28081       values be used for this **ustar** header block. Making these header files inaccessible for convenient  
28082       reading and deleting would not be reasonable. File permissions of 600 or 700 are suggested.

28083       The **ustar** *typeflag* field was used to accommodate the additional functionality of the new format  
28084       rather than magic or version because the POSIX.1-1990 standard (and, by reference, the previous  
28085       version of *pax*), mandated the behavior of the format-reading utility when it encountered an  
28086       unknown *typeflag*, but was silent about the other two fields.

28087       Early proposals of the first revision to IEEE Std. 1003.1-200x contained a proposed archive  
28088       format that was based on compatibility with the standard for tape files (ISO 1001, similar to the  
28089       format used historically on many mainframes and minicomputers). This format was overly  
28090       complex and required considerable overhead in volume and header records. Furthermore, the  
28091       standard developers felt that it would not be acceptable to the community of POSIX developers,  
28092       so it was later changed to be a format more closely related to historical practice on POSIX  
28093       systems.

28094       The prefix and name split of path names in **ustar** was replaced by the single path extended  
28095       header record for simplicity.

28096       The concept of a global extended header (*typeflag g*) was controversial. If this were applied to an  
28097       archive being recorded on magnetic tape, a few unreadable blocks at the beginning of the tape  
28098       could be a serious problem; a utility attempting to extract as many files as possible from a  
28099       damaged archive could lose a large percentage of file header information in this case. However,  
28100       if the archive were on a reliable medium, such as a CD-ROM, the global extended header offers  
28101       considerable potential size reductions by eliminating redundant information. Thus, the text  
28102       warns against using the global method for unreliable media and provides a method for  
28103       implanting global information in the extended header for each file, rather than in the *typeflag g*  
28104       records.

28105       No facility for data translation or filtering on a per-file basis is included because the standard  
28106       developers could not invent an interface that would allow this in an efficient manner. If a filter,  
28107       such as encryption or compression, is to be applied to all the files, it is more efficient to apply the  
28108       filter to the entire archive as a single file. The standard developers considered interfaces that  
28109       would invoke a shell script for each file going into or out of the archive, but the system overhead  
28110       in this approach was considered to be too high.

28111       One such approach would be to have **filter=** records that give a path name for an executable.  
28112       When the program is invoked, the file and archive would be open for standard input/output  
28113       and all the header fields would be available as environment variables or command-line  
28114       arguments. The standard developers did discuss such schemes, but they were omitted from  
28115       IEEE Std. 1003.1-200x due to concerns about excessive overhead. Also, the program itself would  
28116       need to be in the archive if it were to be used portably.

28117       There is currently no portable means of identifying the character set(s) used for a file in the file  
28118       system. Therefore, *pax* has not been given a mechanism to generate charset records  
28119       automatically. The only portable means of doing this is for the user to write the archive using the



28120        –**charset=string** command line option. This assumes that all of the files in the archive use the  
 28121 same encoding. The “implementation-defined” text is included to allow for a system that can  
 28122 identify the encodings used for each of its files.

28123        The table of standards that accompanies the charset record description is acknowledged to be  
 28124 very limited. Only a limited number of character set standards is reasonable for maximal  
 28125 interchange. Any character set is, of course, possible by prior agreement. It was suggested that  
 28126 EBCDIC be listed, but it was omitted because it is not defined by a formal standard. Formal  
 28127 standards, and then only those with reasonably large followings, can be included here, simply as  
 28128 a matter of practicality. The <value>s represent names of officially registered charactersets in the  
 28129 format required by the ISO 2375:1985 standard.

28130        The normal comma or <blank>-separated list rules are not followed in the case of keyword  
 28131 options to allow ease of argument parsing for *getopts*.

28132        Further information on character encodings is in **pax Archive Character Set Encoding/Decoding**  
 28133 (on page 2941).

28134        The standard developers have reserved keyword name space for vendor extensions. It is  
 28135 suggested that the format to be used is:

28136        *VENDOR.keyword*

28137        where *VENDOR* is the name of the vendor or organization in all uppercase letters. It is further  
 28138 suggested that the keyword following the period be named differently than any of the standard  
 28139 keywords so that it could be used for future standardization, if appropriate, by omitting the  
 28140 *VENDOR* prefix.

28141        The <length> field in the extended header record was included to make it simpler to step  
 28142 through the records, even if a record contains an unknown format (to a particular *pax*) with  
 28143 complex interactions of special characters. It also provides a minor integrity checkpoint within  
 28144 the records to aid a program attempting to recover files from a damaged archive.

28145        There are no extended header versions of the *devmajor* and *devminor* fields because the  
 28146 unspecified format **ustar** header field should be sufficient. If they are not, vendor-specific  
 28147 extended keywords (such as *VENDOR.devmajor*) should be used.

28148        Device and *i*-number labeling of files was not adopted from *cpio*; files are interchanged strictly  
 28149 on a symbolic name basis, as in **ustar**.

28150        Just as with the **ustar** format descriptions, the new format makes no special arrangements for  
 28151 multi-volume archives. Each of the *pax* archive types is assumed to be inside a single POSIX file  
 28152 and splitting that file over multiple volumes (diskettes, tape cartridges, and so on), processing  
 28153 their labels, and mounting each in the proper sequence are considered to be implementation  
 28154 details that cannot be described portably.

28155        The *pax* format is intended for interchange, not only for backup on a single (family of) systems. It  
 28156 is not as densely packed as might be possible for backup:

28157        

- It contains information as coded characters that could be coded in binary.
- It identifies extended records with name fields that could be omitted in favor of a fixed-field layout.
- It translates names into a portable character set and identifies locale-related information, both of which are probably unnecessary for backup.

28162        The requirements on restoring from an archive are slightly different from the historical wording,  
 28163 allowing for non-monolithic privilege to bring forward as much as possible. In particular,  
 28164 attributes such as “high performance file” might be broadly but not universally granted while

28165 set-user-ID or *chown()* might be much more restricted. There is no implication in  
28166 IEEE Std. 1003.1-200x that the security information be honored after it is restored to the file  
28167 hierarchy, in spite of what might be improperly inferred by the silence on that topic. That is a  
28168 topic for another standard.

28169 Links are recorded in the fashion described here because a link can be to any file type. It is  
28170 desirable in general to be able to restore part of an archive selectively and restore all of those  
28171 files completely. If the data is not associated with each link, it is not possible to do this.  
28172 However, the data associated with a file can be large, and when selective restoration is not  
28173 needed, this can be a significant burden. The archive is structured so that files that have no  
28174 associated data can always be restored by the name of any link name of any link, and the user  
28175 may choose whether data is recorded with each instance of a file that contains data. The format  
28176 permits mixing of both types of links in a single archive; this can be done for special needs, and  
28177 *pax* is expected to interpret such archives on input properly, despite the fact that there is no *pax*  
28178 option that would force this mixed case on output. (When **-o linkdata** is used, the output must  
28179 contain the duplicate data, but the implementation is free to include it or omit it when **-o**  
28180 **linkdata** is not used.)

28181 The time values are included as extended header records for those implementations needing  
28182 more than the eleven octal digits allowed by the **ustar** format. Portable file timestamps cannot be  
28183 negative. If *pax* encounters a file with a negative timestamp in **copy** or **write** mode, it can reject  
28184 the file, substitute a non-negative timestamp, or generate a non-portable timestamp with a  
28185 leading ' - '. Even though some implementations can support finer file-time granularities than  
28186 seconds, the normative text requires support only for seconds since the Epoch because the  
28187 ISO POSIX-1 standard states them that way. The **ustar** format includes only *mtime*; the new  
28188 format adds *atime* and *ctime* for symmetry. The *atime* access time restored to the file system will  
28189 be affected by the **-p a** and **-p e** options. The *ctime* creation time (actually *inode* modification  
28190 time) is described with “appropriate privilege” so that it can be ignored when writing to the file  
28191 system. POSIX does not provide a portable means to change file creation time. Nothing is  
28192 intended to prevent a non-portable implementation of *pax* from restoring the value.

28193 The *gid*, *size*, and *uid* extended header records were included to allow expansion beyond the  
28194 sizes specified in the regular *tar* header. New file system architectures are emerging that will  
28195 exhaust the 12-digit size field. There are probably not many systems requiring more than 8 digits  
28196 for user and group IDs, but the extended header values were included for completeness,  
28197 allowing overrides for all of the decimal values in the *tar* header.

28198 The standard developers intended to describe the effective results of *pax* with regard to file  
28199 ownerships and permissions; implementations are not restricted in timing or sequencing the  
28200 restoration of such, provided the results are as specified.

28201 Much of the text describing the extended headers refers to use in “**write** or **copy** modes”. The  
28202 **copy** mode references are due to the normative text: “The effect of the copy shall be as if the  
28203 copied files were written to an archive file and then subsequently extracted ...”. There is  
28204 certainly no way to test whether *pax* is actually generating the extended headers in **copy** mode,  
28205 but the effects must be as if it had.

**28206 pax Archive Character Set Encoding/Decoding**

28207 There is a need to exchange archives of files between systems of different native codesets. File  
28208 names, group names, and user names must be preserved to the fullest extent possible when an  
28209 archive is read on the receiving platform. Translation of the contents of files is not within the  
28210 scope of the *pax* utility.

28211 There will also be the need to represent glyphs that are not available on the receiving platform.  
28212 (A *glyph* is commonly called a character, but without any reference to a specific encoding of that  
28213 character. The term *glyph* refers to the symbol itself.) These unsupported glyphs cannot be  
28214 automatically folded to the local set of glyphs due to the chance of collisions. This could result in  
28215 overwriting previous extracted files from the archive or pre-existing files on the system.

28216 For these reasons, the codeset used to represent glyphs within the extended header records of  
28217 the *pax* archive must be sufficiently rich to handle all commonly used character sets. The fields  
28218 requiring translation include, at a minimum, file names, user names, group names, and link path  
28219 names. Implementations may wish to have localized extended keywords that use non-portable  
28220 characters.

28221 The standard developers considered the following options:

- 28222 • The archive creator specifies the well-defined name of the source codeset. The receiver must  
28223 then recognize the codeset name and perform the appropriate translations to the destination  
28224 codeset.
- 28225 • The archive creator includes within the archive the character mapping table for the source  
28226 codeset used to encode extended header records. The receiver must then read the character  
28227 mapping table and perform the appropriate translations to the destination codeset.
- 28228 • The archive creator translates the extended header records in the source codeset into a  
28229 canonical form. The receiver must then perform the appropriate translations to the  
28230 destination codeset.

28231 The approach that incorporates the name of the source codeset poses the problem of codeset  
28232 name registration, and makes the archive useless to *pax* archive decoders that do not recognize  
28233 that codeset.

28234 Because parts of an archive may be corrupted, the standard developers felt that including the  
28235 character map of the source codeset was too fragile. The loss of this one key component could  
28236 result in making the entire archive useless. (The difference between this and the global extended  
28237 header decision was that the latter has a workaround—duplicating extended header records on  
28238 unreliable media—but this would be too burdensome for large character set maps.)

28239 Both of the above approaches also put an undue burden on the *pax* archive receiver to handle the  
28240 cross-product of all source and destination codesets.

28241 To simplify the translation from the source codeset to the canonical form and from the canonical  
28242 form to the destination codeset, the standard developers decided that the internal representation  
28243 should be a stateless encoding. A stateless encoding is one where each codepoint has the same  
28244 meaning, without regard to the decoder being in a specific state. An example of a stateful  
28245 encoding would be the Japanese Shift-JIS; an example of a stateless encoding would be the  
28246 ISO/IEC 646:1991 standard (equivalent to 7-bit ASCII).

28247 For these reasons, the standard developers decided to adopt a canonical format for the  
28248 representation of file information strings. The obvious, well-endorsed candidate is the  
28249 ISO/IEC 10646-1:1993 standard (based in part on Unicode), which can be used to represent the  
28250 glyphs of virtually all standardized character sets. The standard developers initially agreed upon  
28251 using UCS2 (16-bit Unicode) as the internal representation. This repertoire of glyphs provides a

28252 sufficiently rich set to represent all commonly-used codesets.

28253 However, the standard developers found that the 16-bit Unicode representation had some  
 28254 problems. It forced the issue of standardizing byte ordering. The 2-byte length of each character  
 28255 made the extended header records twice as long for the case of strings coded entirely from  
 28256 historical 7-bit ASCII. For these reasons, the standard developers chose the UTF-8 defined in the  
 28257 ISO/IEC 10646-1:1993 standard. This multi-byte representation encodes UCS2 or UCS4  
 28258 characters reliably and deterministically, eliminating the need for a canonical byte ordering. In  
 28259 addition, NUL octets and other characters possibly confusing to POSIX file systems do not  
 28260 appear, except to represent themselves. It was realized that certain national codesets take up  
 28261 more space after the encoding, due to their placement within the UCS range; it was felt that the  
 28262 usefulness of the encoding of the names outweighs the disadvantage of size increase for file,  
 28263 user, and group names.

28264 The encoding of UTF-8 is as follows:

| 28265 | UCS4 Hex Encoding | UTF-8 Binary Encoding                                 |
|-------|-------------------|-------------------------------------------------------|
| 28266 | 00000000-0000007F | 0xxxxxxx                                              |
| 28267 | 00000080-000007FF | 110xxxxx 10xxxxxx                                     |
| 28268 | 00000800-0000FFFF | 1110xxxx 10xxxxxx 10xxxxxx                            |
| 28269 | 00010000-001FFFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx                   |
| 28270 | 00200000-03FFFFFF | 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx          |
| 28271 | 04000000-7FFFFFFF | 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx |

28272 where each 'x' represents a bit value from the character being translated.

### 28273 **ustar Interchange Format**

28274 The description of the **ustar** format reflects numerous enhancements over pre-1988 versions of  
 28275 the historical *tar* utility. The goal of these changes was not only to provide the functional  
 28276 enhancements desired, but also to retain compatibility between new and old versions. This  
 28277 compatibility has been retained. Archives written using the old archive format are compatible  
 28278 with the new format.

28279 Implementors should be aware that the previous file format did not include a mechanism to  
 28280 archive directory type files. For this reason, the convention of using a file name ending with  
 28281 slash was adopted to specify a directory on the archive.

28282 The total size of the *name* and *prefix* fields have been set to meet the minimum requirements for  
 28283 {PATH\_MAX}. If a path name will fit within the *name* field, it is recommended that the path  
 28284 name be stored there without the use of the *prefix* field. Although the name field is known to be  
 28285 too small to contain {PATH\_MAX} characters, the value was not changed in this version of the  
 28286 archive file format to retain backward compatibility, and instead the prefix was introduced.  
 28287 Also, because of the earlier version of the format, there is no way to remove the restriction on the  
 28288 *linkname* field being limited in size to just that of the *name* field.

28289 The *size* field is required to be meaningful in all implementation extensions, although it could be  
 28290 zero. This is required so that the data blocks can always be properly counted.

28291 It is suggested that if device special files need to be represented that cannot be represented in the  
 28292 standard format that one of the extension types (A-Z) be used, and that the additional  
 28293 information for the special file be represented as data and be reflected in the *size* field.

28294 Attempting to restore a special file type, where it is converted to ordinary data and conflicts  
 28295 with an existing file name, need not be specially detected by the utility. If run as an ordinary  
 28296 user, *pax* should not be able to overwrite the entries in, for example, */dev* in any case (whether  
 28297 the file is converted to another type or not). If run as a privileged user, it should be able to do so,

28298 and it would be considered a bug if it did not. The same is true of ordinary data files and  
 28299 similarly named special files; it is impossible to anticipate the needs of the user (who could  
 28300 really intend to overwrite the file), so the behavior should be predictable (and thus regular) and  
 28301 rely on the protection system as required.

28302 The value 7 in the *typeflag* field is intended to define how contiguous files can be stored in a  
 28303 **ustar** archive. IEEE Std. 1003.1-200x does not require the contiguous file extension, but does  
 28304 define a standard way of archiving such files so that all conforming systems can interpret these  
 28305 file types in a meaningful and consistent manner. On a system that does not support extended  
 28306 file types, the *pax* utility should do the best it can with the file and go on to the next.

28307 The file protection modes are those conventionally used by the *ls* utility. This is extended  
 28308 beyond the usage in the ISO POSIX-2 standard to support the “shared text” or “sticky” bit. It is  
 28309 intended that the conformance document should not document anything beyond the existence  
 28310 of and support of such a mode. Further extensions are expected to these bits, particularly with  
 28311 overloading the set-user-ID and set-group-ID flags.

### 28312 **cpio Interchange Format**

28313 The reference to appropriate privilege in the *cpio* format refers to an error on standard output;  
 28314 the **ustar** format does not make comparable statements.

28315 The model for this format was the historical System V *cpio-c* data interchange format. This  
 28316 model documents the portable version of the *cpio* format and not the binary version. It has the  
 28317 flexibility to transfer data of any type described within IEEE Std. 1003.1-200x, yet is extensible to  
 28318 transfer data types specific to extensions beyond IEEE Std. 1003.1-200x (for example, contiguous  
 28319 files). Because it describes existing practice, there is no question of maintaining upward  
 28320 compatibility.

### 28321 **cpio Header**

28322 There has been some concern that the size of the *c\_ino* field of the header is too small to handle  
 28323 those systems that have very large *inode* numbers. However, the *c\_ino* field in the header is used  
 28324 strictly as a hard-link resolution mechanism for archives. It is not necessarily the same value as  
 28325 the *inode* number of the file in the location from which that file is extracted.

28326 The name *c\_magic* is based on historical usage.

### 28327 **cpio File Name**

28328 For most historical implementations of the *cpio* utility, {PATH\_MAX} octets can be used to  
 28329 describe the path name without the addition of any other header fields (the NUL character  
 28330 would be included in this count). {PATH\_MAX} is the minimum value for path name size,  
 28331 documented as 256 bytes. However, an implementation may use *c\_namesize* to determine the  
 28332 exact length of the path name. With the current description of the **<cpio.h>** header, this path  
 28333 name size can be as large as a number that is described in six octal digits.

28334 Two values are documented under the *c\_mode* field values to provide for extensibility for known  
 28335 file types:

28336 **Notes to Reviewers**28337 *This section with side shading will not appear in the final copy. - Ed.*28338 Note that the sockets extension below needs to be integrated, now that sockets have been  
28339 merged28340 **0110 000** Reserved for contiguous files. The implementation may treat the rest of the  
28341 information for this archive like a regular file. If this file type is undefined, the  
28342 implementation may create the file as a regular file.28343 **0140 000** Reserved for sockets. If this type is undefined on the target system, the  
28344 implementation may decide to ignore this file type and output a warning message.28345 This provides for extensibility of the *cpio* format while allowing for the ability to read old  
28346 archives. Files of an unknown type may be read as “regular files” on some implementations. On  
28347 a system that does not support extended file types, the *pax* utility should do the best it can with  
28348 the file and go on to the next.28349 **FUTURE DIRECTIONS**

28350 None.

28351 **SEE ALSO**28352 *cp*, *ed*, *getopts*, *printf*, the Base Definitions volume of IEEE Std. 1003.1-200x, **<cpio.h>**, the System  
28353 Interfaces volume of IEEE Std. 1003.1-200x, *chown()*, *creat()*, *mkdir()*, *stat()*, *write()*28354 **CHANGE HISTORY**

28355 First released in Issue 4.

28356 **Issue 5**28357 A note is added to the APPLICATION USAGE indicating that the *cpio* and *tar* formats can only  
28358 support files up to 8 gigabytes in size.28359 **Issue 6**28360 The *pax* utility is aligned with the IEEE P1003.2b draft standard:

- 28361
- Support has been added for symbolic links in the options and interchange formats.
  - A new format has been devised, based on extensions to *ustar*.
  - References to the “extended” *tar* and *cpio* formats derived from the POSIX.1-1990 standard have been changed to remove the “extended” adjective because this could cause confusion with the extended *tar* header added in this revision. (All references to *tar* are actually to **ustar**).

28367 IEEE PASC Interpretation 1003.2 #168 is applied clarifying that *mkdir()* and *mkfifo()* calls can  
28368 ignore an [EEXIST] error when extracting an archive.

## 28369 NAME

28370 pr — print files

## 28371 SYNOPSIS

```
28372 pr [+page][-column][-adFmrt][-e[char][gap]][-h header][-i[char][gap]]
28373 xSI [-l lines][-n[char][width]][-o offset][-s[char]][-w width][-fp]
28374 [file...]
```

## 28375 DESCRIPTION

28376 The *pr* utility is a printing and pagination filter. If multiple input files are specified, each shall be  
 28377 read, formatted, and written to standard output. By default, the input shall be separated into 66-  
 28378 line pages, each with:

- 28379 • A 5-line header that includes the page number, date, time, and the path name of the file
- 28380 • A 5-line trailer consisting of blank lines

28381 If standard output is associated with a terminal, diagnostic messages shall be deferred until the  
 28382 *pr* utility has completed processing.

28383 When options specifying multi-column output are specified, output text columns shall be of  
 28384 equal width; input lines that do not fit into a text column shall be truncated. By default, text  
 28385 columns shall be separated with at least one <blank> character.

## 28386 OPTIONS

28387 The *pr* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 28388 Utility Syntax Guidelines, except that: the *page* option has a '+' delimiter; *page* and *column* can  
 28389 be multi-digit numbers; some of the option-arguments are optional; and some of the option-  
 28390 arguments cannot be specified as separate arguments from the preceding option letter. In  
 28391 particular, the *-s* option does not allow the option letter to be separated from its argument, and  
 28392 the options *-e*, *-i*, and *-n* require that both arguments, if present, not be separated from the  
 28393 option letter.

28394 The following options shall be supported. In the following option descriptions, *column*, *lines*,  
 28395 *offset*, *page*, and *width* are positive decimal integers; *gap* is a non-negative decimal integer.

- 28396 *+page*      Begin output at page number *page* of the formatted input.
- 28397 *-column*     Produce multi-column output that is arranged in *column* columns (the default shall  
 28398 be 1) and is written down each column in the order in which the text is received  
 28399 from the input file. This option should not be used with *-m*. The options *-e* and *-i*  
 28400 shall be assumed for multiple text-column output. Whether or not text columns  
 28401 are produced with identical vertical lengths is unspecified, but a text column shall  
 28402 never exceed the length of the page (see the *-l* option). When used with *-t*, use the  
 28403 minimum number of lines to write the output.
- 28404 *-a*          Modify the effect of the *-column* option so that the columns are filled across the  
 28405 page in a round-robin order (for example, when *column* is 2, the first input line  
 28406 heads column 1, the second heads column 2, the third is the second line in column  
 28407 1, and so on).
- 28408 *-d*          Produce output that is double-spaced; append an extra <newline> character  
 28409 following every <newline> character found in the input.
- 28410 *-e[*char*][*gap*]*  
 28411              Expand each input <tab> character to the next greater column position specified  
 28412 by the formula  $n * \textit{gap} + 1$ , where *n* is an integer > 0. If *gap* is zero or is omitted, it  
 28413 shall default to 8. All <tab> characters in the input shall be expanded into the  
 28414 appropriate number of <space> characters. If any non-digit character, *char*, is

- 28415 specified, it shall be used as the input <tab> character.
- 28416 XSI **-f** Use a <form-feed> character for new pages, instead of the default behavior that  
28417 uses a sequence of <newline> characters. Pause before beginning the first page if  
28418 the standard output is associated with a terminal.
- 28419 **-F** Use a <form-feed> character for new pages, instead of the default behavior that  
28420 uses a sequence of <newline> characters.
- 28421 **-h header** Use the string *header* to replace the contents of the *file* operand in the page header.
- 28422 **-i[char][gap]**  
28423 In output, replace multiple <space> characters with <tab> characters wherever  
28424 two or more adjacent <space> characters reach column positions *gap*+1, 2\* *gap*+1,  
28425 3\* *gap*+1, and so on. If *gap* is zero or is omitted, default tab settings at every eighth  
28426 column position shall be assumed. If any non-digit character, *char*, is specified, it  
28427 shall be used as the output <tab> character.
- 28428 **-l lines** Override the 66-line default and reset the page length to *lines*. If *lines* is not greater  
28429 than the sum of both the header and trailer depths (in lines), the *pr* utility shall  
28430 suppress both the header and trailer, as if the **-t** option were in effect.
- 28431 **-m** Merge files. Standard output shall be formatted so the *pr* utility writes one line  
28432 from each file specified by a *file* operand, side by side into text columns of equal  
28433 fixed widths, in terms of the number of column positions. Implementations shall  
28434 support merging of at least nine *file* operands.
- 28435 **-n[char][width]**  
28436 Provide *width*-digit line numbering (default for *width* shall be 5). The number shall  
28437 occupy the first *width* column positions of each text column of default output or  
28438 each line of **-m** output. If *char* (any non-digit character) is given, it shall be  
28439 appended to the line number to separate it from whatever follows (default for *char*  
28440 is a <tab> character).
- 28441 **-o offset** Each line of output shall be preceded by offset <space>*s*. If the **-o** option is not  
28442 specified, the default offset shall be zero. The space taken is in addition to the  
28443 output line width (see the **-w** option below).
- 28444 **-p** Pause before beginning each page if the standard output is directed to a terminal  
28445 (*pr* shall write an <alert> character to standard error and wait for a <carriage-  
28446 return> character to be read on */dev/tty*).
- 28447 **-r** Write no diagnostic reports on failure to open files.
- 28448 **-s[char]** Separate text columns by the single character *char* instead of by the appropriate  
28449 number of <space> characters (default for *char* shall be the <tab> character).
- 28450 **-t** Write neither the five-line identifying header nor the five-line trailer usually  
28451 supplied for each page. Quit writing after the last line of each file without spacing  
28452 to the end of the page.
- 28453 **-w width** Set the width of the line to *width* column positions for multiple text-column output  
28454 only. If the **-w** option is not specified and the **-s** option is not specified, the default  
28455 width shall be 72. If the **-w** option is not specified and the **-s** option is specified,  
28456 the default width shall be 512.
- 28457 For single column output, input lines shall not be truncated.



28458 **OPERANDS**

28459 The following operand shall be supported:

28460 *file* A path name of a file to be written. If no *file* operands are specified, or if a *file*  
 28461 operand is '-', the standard input shall be used.

28462 **STDIN**

28463 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
 28464 See the INPUT FILES section.

28465 **INPUT FILES**

28466 The input files shall be text files.

28467 The file `/dev/tty` is used to read responses required by the `-p` option.28468 **ENVIRONMENT VARIABLES**28469 The following environment variables shall affect the execution of *pr*:

28470 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 28471 If *LANG* is unset or null, the corresponding value from the implementation-  
 28472 defined default locale shall be used. If any of the internationalization variables  
 28473 contains an invalid setting, the utility shall behave as if none of the variables had  
 28474 been defined.

28475 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 28476 internationalization variables.

28477 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 28478 characters (for example, single-byte as opposed to multi-byte characters in  
 28479 arguments and input files) and which characters are defined as printable (character  
 28480 class **print**). Non-printable characters are still written to standard output, but are  
 28481 not counted for the purpose for column-width and line-length calculations.

28482 *LC\_MESSAGES*

28483 Determine the locale that should be used to affect the format and contents of  
 28484 diagnostic messages written to standard error.

28485 *LC\_TIME* Determine the format of the date and time for use in writing header lines.28486 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.28487 *TZ* Determine the timezone for use in writing header lines.28488 **ASYNCHRONOUS EVENTS**

28489 If *pr* receives an interrupt while writing to a terminal, it shall flush all accumulated error  
 28490 messages to the screen before terminating.

28491 **STDOUT**

28492 The *pr* utility output shall be a paginated version of the original file (or files). This pagination  
 28493 shall be accomplished using either `<form-feed>` characters or a sequence of `<newline>`  
 28494 XSI characters, as controlled by the `-F` or `-f` option. Page headers shall be generated unless the `-t`  
 28495 option is specified. The page headers shall be of the form:

28496 `"\n\n%s %s Page %d\n\n\n", <output of date>, <file>, <page number>`

28497 In the POSIX locale, the `<output of date>` field, representing the date and time of last modification  
 28498 of the input file (or the current date and time if the input file is standard input), shall be  
 28499 equivalent to the output of the following command as it would appear if executed at the given  
 28500 time:

- 28501           date "+%b %e %H:%M %Y"
- 28502           without the trailing <newline> character, if the page being written is from standard input. If the  
28503           page being written is not from standard input, in the POSIX locale, the same format shall be  
28504           used, but the time used shall be the modification time of the file corresponding to *file* instead of  
28505           the current time. When the *LC\_TIME* locale category is not set to the POSIX locale, a different  
28506           format and order of presentation of this field may be used.
- 28507           If the standard input is used instead of a *file* operand, the <*file*> field shall be replaced by a null  
28508           string.
- 28509           If the **-h** option is specified, the <*file*> field shall be replaced by the *header* argument.
- 28510 **STDERR**
- 28511           Used for diagnostic messages and for alerting the terminal when **-p** is specified.
- 28512 **OUTPUT FILES**
- 28513           None.
- 28514 **EXTENDED DESCRIPTION**
- 28515           None.
- 28516 **EXIT STATUS**
- 28517           The following exit values shall be returned:
- 28518           0   Successful completion.
- 28519           >0  An error occurred.
- 28520 **CONSEQUENCES OF ERRORS**
- 28521           Default.
- 28522 **APPLICATION USAGE**
- 28523           None.
- 28524 **EXAMPLES**
- 28525           1.  Print a numbered list of all files in the current directory:
- 28526                 ls -a | pr -n -h "Files in \$(pwd)."
- 28527           2.  Print **file1** and **file2** as a double-spaced, three-column listing headed by "file list":
- 28528                 pr -3d -h "file list" file1 file2
- 28529           3.  Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, ...:
- 28530                 pr -e9 -t <file1 >file2
- 28531 **RATIONALE**
- 28532           This utility is one of those that does not follow the Utility Syntax Guidelines because of its  
28533           historical origins. The standard developers could have added new options that obeyed the  
28534           guidelines (and marked the old options *obsolescent*) or devised an entirely new utility; there are  
28535           examples of both actions in this volume of IEEE Std. 1003.1-200x. Because of its widespread use  
28536           by historical applications, the standard developers decided to exempt this version of *pr* from  
28537           many of the guidelines.
- 28538           Implementations are required to accept option-arguments to the **-h**, **-l**, **-o**, and **-w** options  
28539           whether presented as part of the same argument or as a separate argument to *pr*, as suggested by  
28540           the Utility Syntax Guidelines. The **-n** and **-s** options, however, are specified as in historical  
28541           practice because they are frequently specified without their optional arguments. If a <blank>  
28542           were allowed before the option-argument in these cases, a *file* operand could mistakenly be

- 28543 interpreted as an option-argument in historical applications.
- 28544 The text about the minimum number of lines in multi-column output was included to ensure  
28545 that a best effort is made in balancing the length of the columns. There are known historical  
28546 implementations in which, for example, 60-line files are listed by *pr -2* as one column of 56 lines  
28547 and a second of 4. Although this is not a problem when a full page with headers and trailers is  
28548 produced, it would be relatively useless when used with *-t*.
- 28549 Historical implementations of the *pr* utility have differed in the action taken for the *-f* option.  
28550 BSD uses it as described here for the *-F* option; System V uses it to change trailing *<newline>s*  
28551 on each page to a *<form-feed>* and, if standard output is a TTY device, sends an *<alert>*  
28552 to standard error and reads a line from */dev/tty* before the first page. There were strong arguments  
28553 from both sides of this issue concerning historical practice and additional arguments against the  
28554 System V *-f* behavior, on the grounds that having the behavior of an option change depending  
28555 on where output is directed was not a modular design. Therefore, the *-f* option is not specified  
28556 and the *-F* option has been added.
- 28557 The *<output of date>* field in the *-I* format is specified only for the POSIX locale. As noted, the  
28558 format can be different in other locales. No mechanism for defining this is present in this volume  
28559 of IEEE Std. 1003.1-200x, as the appropriate vehicle is a message catalog; that is, the format  
28560 should be specified as a “message”.
- 28561 **FUTURE DIRECTIONS**
- 28562 It is possible that a new interface that conforms to the Utility Syntax Guidelines will be  
28563 introduced.
- 28564 **SEE ALSO**
- 28565 *expand, lp*
- 28566 **CHANGE HISTORY**
- 28567 First released in Issue 2.
- 28568 **Issue 4**
- 28569 Aligned with the ISO/IEC 9945-2:1993 standard.
- 28570 **Issue 6**
- 28571 The following new requirements on POSIX implementations derive from alignment with the  
28572 Single UNIX Specification:
- 28573
  - The *-p* option is added.
- 28574 The normative text is reworded to avoid use of the term “must” for application requirements.

28575 **NAME**

28576 printf — write formatted output

28577 **SYNOPSIS**28578 printf *format*[*argument...*]28579 **DESCRIPTION**28580 The *printf* utility shall write formatted operands to the standard output. The *argument* operands  
28581 shall be formatted under control of the *format* operand.28582 **OPTIONS**

28583 None.

28584 **OPERANDS**

28585 The following operands shall be supported:

28586 *format* A string describing the format to use to write the remaining operands. See the  
28587 EXTENDED DESCRIPTION section.28588 *argument* The strings to be written to standard output, under the control of *format*. See the  
28589 EXTENDED DESCRIPTION section.28590 **STDIN**

28591 Not used.

28592 **INPUT FILES**

28593 None.

28594 **ENVIRONMENT VARIABLES**28595 The following environment variables shall affect the execution of *printf*:28596 *LANG* Provide a default value for the internationalization variables that are unset or null.  
28597 If *LANG* is unset or null, the corresponding value from the implementation-  
28598 defined default locale shall be used. If any of the internationalization variables  
28599 contains an invalid setting, the utility shall behave as if none of the variables had  
28600 been defined.28601 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
28602 internationalization variables.28603 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
28604 characters (for example, single-byte as opposed to multi-byte characters in  
28605 arguments).28606 *LC\_MESSAGES*28607 Determine the locale that should be used to affect the format and contents of  
28608 diagnostic messages written to standard error.28609 *LC\_NUMERIC*28610 Determine the locale for numeric formatting. It shall affect the format of numbers  
28611 written using the *e*, *E*, *f*, *g*, and *G* conversion characters (if supported).28612 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.28613 **ASYNCHRONOUS EVENTS**

28614 Default.

28615 **STDOUT**

28616 See the EXTENDED DESCRIPTION section.

28617 **STDERR**

28618 Used only for diagnostic messages.

28619 **OUTPUT FILES**

28620 None.

28621 **EXTENDED DESCRIPTION**28622 The *format* operand shall be used as the *format* string described in the Base Definitions volume of  
28623 IEEE Std. 1003.1-200x, Chapter 5, File Format Notation with the following exceptions:

- 28624 1. A <space> character in the format string, in any context other than a flag of a conversion  
28625 specification, shall be treated as an ordinary character that is copied to the output.
- 28626 2. A ' $\Delta$ ' character in the format string shall be treated as a ' $\Delta$ ' character, not as a <space>  
28627 character.
- 28628 3. In addition to the escape sequences shown in the Base Definitions volume of  
28629 IEEE Std. 1003.1-200x, Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n',  
28630 '\r', '\t', '\v'), "\ddd", where *ddd* is a one, two, or three-digit octal number, shall be  
28631 written as a byte with the numeric value specified by the octal number.
- 28632 4. The implementation shall not precede or follow output from the *d* or *u* conversion  
28633 specifications with <blank> characters not specified by the *format* operand.
- 28634 5. The implementation shall not precede output from the *o* conversion specification with  
28635 zeros not specified by the *format* operand.
- 28636 6. The *e*, *E*, *f*, *g*, and *G* conversion specifications need not be supported.
- 28637 7. An additional conversion character, *b*, shall be supported as follows. The argument shall  
28638 be taken to be a string that may contain backslash-escape sequences. The following  
28639 backslash-escape sequences shall be supported:
- 28640 — The escape sequences listed in the Base Definitions volume of IEEE Std. 1003.1-200x,  
28641 Chapter 5, File Format Notation ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'),  
28642 which shall be converted to the characters they represent
  - 28643 — "\0ddd", where *ddd* is a zero, one, two, or three-digit octal number that shall be  
28644 converted to a byte with the numeric value specified by the octal number
  - 28645 — '\c', which shall not be written and shall cause *printf* to ignore any remaining  
28646 characters in the string operand containing it, any remaining string operands, and any  
28647 additional characters in the *format* operand
- 28648 The interpretation of a backslash followed by any other sequence of characters is  
28649 unspecified.
- 28650 Bytes from the converted string shall be written until the end of the string or the number of  
28651 bytes indicated by the precision specification is reached. If the precision is omitted, it shall  
28652 be taken to be infinite, so all bytes up to the end of the converted string shall be written.
- 28653 8. For each specification that consumes an argument, the next argument operand shall be  
28654 evaluated and converted to the appropriate type for the conversion as specified below.
- 28655 9. The *format* operand shall be reused as often as necessary to satisfy the argument operands.  
28656 Any extra *c* or *s* conversion specifications shall be evaluated as if a null string argument  
28657 were supplied; other extra conversion specifications shall be evaluated as if a zero  
28658 argument were supplied. If the *format* operand contains no conversion specifications and

- 28659 *argument* operands are present, the results are unspecified.
- 28660 10. If a character sequence in the *format* operand begins with a '%' character, but does not  
28661 form a valid conversion specification, the behavior is unspecified.
- 28662 The *argument* operands shall be treated as strings if the corresponding conversion character is *b*,  
28663 *c*, or *s*; otherwise, it shall be evaluated as a C constant, as described by the ISO C standard, with  
28664 the following extensions:
- 28665 • A leading plus or minus sign shall be allowed.
  - 28666 • If the leading character is a single-quote or double-quote, the value shall be the numeric  
28667 value in the underlying codeset of the character following the single-quote or double-quote.
- 28668 If an argument operand cannot be completely converted into an internal value appropriate to  
28669 the corresponding conversion specification, a diagnostic message shall be written to standard  
28670 error and the utility shall not exit with a zero exit status, but shall continue processing any  
28671 remaining operands and shall write the value accumulated at the time the error was detected to  
28672 standard output.
- 28673 It is not considered an error if an argument operand is not completely used for a *c* or *s*  
28674 conversion or if a string operand's first or second character is used to get the numeric value of a  
28675 character.
- 28676 **EXIT STATUS**
- 28677 The following exit values shall be returned:
- 28678 0 Successful completion.
  - 28679 >0 An error occurred.
- 28680 **CONSEQUENCES OF ERRORS**
- 28681 Default.
- 28682 **APPLICATION USAGE**
- 28683 The floating-point formatting conversion specifications of *printf()* are not required because all  
28684 arithmetic in the shell is integer arithmetic. The *awk* utility performs floating-point calculations  
28685 and provides its own **printf** function. The *bc* utility can perform arbitrary-precision floating-  
28686 point arithmetic, but does not provide extensive formatting capabilities. (This *printf* utility  
28687 cannot really be used to format *bc* output; it does not support arbitrary precision.)  
28688 Implementations are encouraged to support the floating-point conversions as an extension.
- 28689 Note that this *printf* utility, like the *printf()* function defined in the System Interfaces volume of  
28690 IEEE Std. 1003.1-200x on which it is based, makes no special provision for dealing with multi-  
28691 byte characters when using the *%c* conversion specification or when a precision is specified in a  
28692 *%b* or *%s* conversion specification. Applications should be extremely cautious using either of  
28693 these features when there are multi-byte characters in the character set.
- 28694 No provision is made in this volume of IEEE Std. 1003.1-200x which allows field widths and  
28695 precisions to be specified as '\*' since the '\*' can be replaced directly in the *format* operand  
28696 using shell variable substitution. Implementations can also provide this feature as an extension  
28697 if they so choose.
- 28698 Hexadecimal character constants as defined in the ISO C standard are not recognized in the  
28699 *format* operand because there is no consistent way to detect the end of the constant. Octal  
28700 character constants are limited to, at most, three octal digits, but hexadecimal character  
28701 constants are only terminated by a non-hex-digit character. In the ISO C standard, the "##"  
28702 concatenation operator can be used to terminate a constant and follow it with a hexadecimal  
28703 character to be written. In the shell, concatenation occurs before the *printf* utility has a chance to

28704 parse the end of the hexadecimal constant.

28705 The `%b` conversion specification is not part of the ISO C standard; it has been added here as a  
 28706 portable way to process backslash escapes expanded in string operands as provided by the `echo`  
 28707 utility. See also the APPLICATION USAGE section of `echo` (on page 2543) for ways to use `printf`  
 28708 as a replacement for all of the traditional versions of the `echo` utility.

28709 If an argument cannot be parsed correctly for the corresponding conversion specification, the  
 28710 `printf` utility is required to report an error. Thus, overflow and extraneous characters at the end  
 28711 of an argument being used for a numeric conversion shall be reported as errors.

#### 28712 EXAMPLES

28713 To alert the user and then print and read a series of prompts:

```
28714 printf "\aPlease fill in the following: \nName: "
28715 read name
28716 printf "Phone number: "
28717 read phone
```

28718 To read out a list of right and wrong answers from a file, calculate the percentage correctly, and  
 28719 print them out. The numbers are right-justified and separated by a single <tab> character. The  
 28720 percentage is written to one decimal place of accuracy:

```
28721 while read right wrong ; do
28722 percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
28723 printf "%2d right\t%2d wrong\t(%%s%%)\n" \
28724 $right $wrong $percent
28725 done < database_file
```

28726 The command:

```
28727 printf "%5d%4d\n" 1 21 321 4321 54321
```

28728 produces:

```
28729 1 21
28730 3214321
28731 54321 0
```

28732 Note that the `format` operand is used three times to print all of the given strings and that a '0'  
 28733 was supplied by `printf` to satisfy the last `%4d` conversion specification.

28734 The `printf` utility is required to notify the user when conversion errors are detected while  
 28735 producing numeric output; thus, the following results would be expected on an implementation  
 28736 with 32-bit twos-complement integers when `%d` is specified as the `format` operand:

| Argument    | Standard Output | Diagnostic Output                                      |
|-------------|-----------------|--------------------------------------------------------|
| 5a          | 5               | <code>printf: "5a" not completely converted</code>     |
| 9999999999  | 2147483647      | <code>printf: "9999999999" arithmetic overflow</code>  |
| -9999999999 | -2147483648     | <code>printf: "-9999999999" arithmetic overflow</code> |
| ABC         | 0               | <code>printf: "ABC" expected numeric value</code>      |

28743 The diagnostic message format is not specified, but these examples convey the type of  
 28744 information that should be reported. Note that the value shown on standard output is what  
 28745 would be expected as the return value from the `strtol()` function as defined in the System  
 28746 Interfaces volume of IEEE Std. 1003.1-200x. A similar correspondence exists between `%u` and  
 28747 `strtoul()` and `%e`, `%f`, and `%g` (if the implementation supports floating-point conversions) and  
 28748 `strtod()`.

- 28749 In a locale using the ISO/IEC 646: 1991 standard as the underlying codeset, the command:
- 28750 `printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"`
- 28751 produces:
- 28752 3 Numeric value of constant 3
- 28753 3 Numeric value of constant 3
- 28754 -3 Numeric value of constant -3
- 28755 51 Numeric value of the character '3' in the ISO/IEC 646: 1991 standard codeset
- 28756 43 Numeric value of the character '+' in the ISO/IEC 646: 1991 standard codeset
- 28757 45 Numeric value of the character '-' in the ISO/IEC 646: 1991 standard codeset
- 28758 Note that in a locale with multi-byte characters, the value of a character is intended to be the
- 28759 value of the equivalent of the `wchar_t` representation of the character as described in the System
- 28760 Interfaces volume of IEEE Std. 1003.1-200x.
- 28761 **RATIONALE**
- 28762 The *printf* utility was added to provide functionality that has historically been provided by *echo*.
- 28763 However, due to irreconcilable differences in the various versions of *echo* extant, the version has
- 28764 few special features, leaving those to this new *printf* utility, which is based on one in the Ninth
- 28765 Edition system.
- 28766 The EXTENDED DESCRIPTION section almost exactly matches the *printf()* function in the
- 28767 ISO C standard, although it is described in terms of the file format notation in the Base
- 28768 Definitions volume of IEEE Std. 1003.1-200x, Chapter 5, File Format Notation.
- 28769 **FUTURE DIRECTIONS**
- 28770 None.
- 28771 **SEE ALSO**
- 28772 *awk*, *bc*, *echo*, the System Interfaces volume of IEEE Std. 1003.1-200x, *printf()*
- 28773 **CHANGE HISTORY**
- 28774 First released in Issue 4.



## 28775 NAME

28776 prs — print an SCCS file (**DEVELOPMENT**)

## 28777 SYNOPSIS

28778 xSI prs [-a][-d *dataspec*][-r[*SID*]] *file...*28779 xSI prs [-e|-l] -c *cutoff* [-d *dataspec*] *file...*28780 xSI prs [-e|-l] -r[*SID*][-d *dataspec*]*file...*

28781

## 28782 DESCRIPTION

28783 The *prs* utility shall write to standard output parts or all of an SCCS file in a user-supplied  
28784 format.

## 28785 OPTIONS

28786 The *prs* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
28787 12.2, Utility Syntax Guidelines, except that the *-r* option has an optional option-argument. This  
28788 optional option-argument cannot be presented as a separate argument. The following options  
28789 shall be supported:28790 *-d dataspec* Specify the output data specification. The *dataspec* shall be a string consisting of  
28791 SCCS file *data keywords* (see **Data Keywords** (on page 2956)) interspersed with  
28792 optional user-supplied text.28793 *-r[SID]* Specify the SCCS identification string (SID) of a delta for which information is  
28794 desired. If no *SID* option-argument is specified, the SID of the most recently  
28795 created delta is assumed.28796 *-e* Request information for all deltas created earlier than and including the delta  
28797 designated via the *-r* option or the date-time given by the *-c* option.28798 *-l* Request information for all deltas created later than and including the delta  
28799 designated via the *-r* option or the date-time given by the *-c* option.28800 *-c cutoff* Indicate the *cutoff* date-time, in the form:

28801 YY[MM[DD[HH[MM[SS]]]]]

28802 For the YY component, values in the range [69-99] shall refer to years in the  
28803 twentieth century (1969 to 1999 inclusive); values in the range [00-68] shall refer to  
28804 years in the twenty-first century (2000 to 2068 inclusive).28805 No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
28806 date-time shall be included in the output. Units omitted from the date-time default  
28807 to their maximum possible values; for example, *-c 7502* is equivalent to  
28808 *-c 750228235959*.28809 *-a* Request writing of information for both removed, that is, *delta type=R* (see *rmddel*  
28810 (on page 3037)) and existing, that is, *delta type=D*, deltas. If the *-a* option is not  
28811 specified, information for existing deltas only shall be provided.

## 28812 OPERANDS

28813 The following operand shall be supported:

28814 *file* A path name of an existing SCCS file or a directory. If *file* is a directory, the *prs*  
28815 utility shall behave as though each file in the directory were specified as a named  
28816 file, except that non-SCCS files (last component of the path name does not begin  
28817 with *s.*) and unreadable files shall be silently ignored.

28818 If a single instance *file* is specified as *'-'*, the standard input shall be read; each  
28819 line of the standard input shall be taken to be the name of an SCCS file to be  
28820 processed. Non-SCCS files and unreadable files shall be silently ignored.

#### 28821 **STDIN**

28822 The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
28823 line of the text file shall be interpreted as an SCCS path name.

#### 28824 **INPUT FILES**

28825 Any SCCS files displayed are files of an unspecified format.

#### 28826 **ENVIRONMENT VARIABLES**

28827 The following environment variables shall affect the execution of *prs*:

28828 *LANG* Provide a default value for the internationalization variables that are unset or null.  
28829 If *LANG* is unset or null, the corresponding value from the implementation-  
28830 defined default locale shall be used. If any of the internationalization variables  
28831 contains an invalid setting, the utility shall behave as if none of the variables had  
28832 been defined.

28833 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
28834 internationalization variables.

28835 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
28836 characters (for example, single-byte as opposed to multi-byte characters in  
28837 arguments and input files).

#### 28838 *LC\_MESSAGES*

28839 Determine the locale that should be used to affect the format and contents of  
28840 diagnostic messages written to standard error.

28841 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 28842 **ASYNCHRONOUS EVENTS**

28843 Default.

#### 28844 **STDOUT**

28845 The standard output shall be a text file whose format is dependent on the data keywords  
28846 specified with the *-d* option.

#### 28847 **Data Keywords**

28848 Data keywords specify which parts of an SCCS file shall be retrieved and output. All parts of an  
28849 SCCS file have an associated data keyword. A data keyword may appear in a *dataspec* multiple  
28850 times.

28851 The information written by *prs* consists of:

- 28852 1. The user-supplied text
- 28853 2. Appropriate values (extracted from the SCCS file) substituted for the recognized data  
28854 keywords in the order of appearance in the *dataspec*

28855 The format of a data keyword value shall either be simple (*'S'*), in which keyword substitution  
28856 is direct, or multi-line (*'M'*).

28857 User-supplied text shall be any text other than recognized data keywords. A *<tab>* character  
28858 shall be specified by *'\t'* and *<newline>* by *'\n'*. When the *-r* option is not specified, the  
28859 default *dataspec* shall be:

28860 :PN: :\n\n

28861 and the following *dataspec* shall be used for each selected delta:

28862 :Dt: \t:DL: \nMRs: \n:MR: COMMENTS: \n:C: \n

28863

28864

28865

28866

28867

28868

28869

28870

28871

28872

28873

28874

28875

28876

28877

28878

28879

28880

28881

28882

28883

28884

28885

28886

28887

28888

28889

28890

28891

28892

28893

28894

28895

28896

28897

28898

28899

28900

28901

28902

28903

28904

28905

28906

| SCCS File Data Keywords |                                                         |              |               |        |
|-------------------------|---------------------------------------------------------|--------------|---------------|--------|
| Keyword                 | Data Item                                               | File Section | Value         | Format |
| :Dt:                    | Delta information                                       | Delta Table  | See below*    | S      |
| :DL:                    | Delta line statistics                                   | "            | :Li:/Ld:/Lu:  | S      |
| :Li:                    | Lines inserted by Delta                                 | "            | nnnnn         | S      |
| :Ld:                    | Lines deleted by Delta                                  | "            | nnnnn         | S      |
| :Lu:                    | Lines unchanged by Delta                                | "            | nnnnn         | S      |
| :DT:                    | Delta type                                              | "            | D or R        | S      |
| :I:                     | SCCS ID string (SID)                                    | "            | See below**   | S      |
| :R:                     | Release number                                          | "            | nnnn          | S      |
| :L:                     | Level number                                            | "            | nnnn          | S      |
| :B:                     | Branch number                                           | "            | nnnn          | S      |
| :S:                     | Sequence number                                         | "            | nnnn          | S      |
| :D:                     | Date delta created                                      | "            | :Dy:/Dm:/Dd:  | S      |
| :Dy:                    | Year delta created                                      | "            | nn            | S      |
| :Dm:                    | Month delta created                                     | "            | nn            | S      |
| :Dd:                    | Day delta created                                       | "            | nn            | S      |
| :T:                     | Time delta created                                      | "            | :Th::Tm::Ts:  | S      |
| :Th:                    | Hour delta created                                      | "            | nn            | S      |
| :Tm:                    | Minutes delta created                                   | "            | nn            | S      |
| :Ts:                    | Seconds delta created                                   | "            | nn            | S      |
| :P:                     | Programmer who created Delta                            | "            | logname       | S      |
| :DS:                    | Delta sequence number                                   | "            | nnnn          | S      |
| :DP:                    | Predecessor Delta sequence number                       | "            | nnnn          | S      |
| :DI:                    | Sequence number of deltas included, excluded or ignored | "            | :Dn:/Dx:/Dg:  | S      |
| :Dn:                    | Deltas included (sequence #)                            | "            | :DS: :DS: ... | S      |
| :Dx:                    | Deltas excluded (sequence #)                            | "            | :DS: :DS: ... | S      |
| :Dg:                    | Deltas ignored (sequence #)                             | "            | :DS: :DS: ... | S      |
| :MR:                    | MR numbers for delta                                    | "            | text          | M      |
| :C:                     | Comments for delta                                      | "            | text          | M      |
| :UN:                    | User names                                              | User Names   | text          | M      |
| :FL:                    | Flag list                                               | Flags        | text          | M      |
| :Y:                     | Module type flag                                        | "            | text          | S      |
| :MF:                    | MR validation flag                                      | "            | yes or no     | S      |
| :MP:                    | MR validation program name                              | "            | text          | S      |
| :KF:                    | Keyword error, warning flag                             | "            | yes or no     | S      |
| :KV:                    | Keyword validation string                               | "            | text          | S      |
| :BF:                    | Branch flag                                             | "            | yes or no     | S      |
| :J:                     | Joint edit flag                                         | "            | yes or no     | S      |
| :LK:                    | Locked releases                                         | "            | :R: ...       | S      |
| :Q:                     | User-defined keyword                                    | "            | text          | S      |

28907  
28908  
28909  
28910  
28911  
28912  
28913  
28914  
28915  
28916  
28917  
28918  
28919  
28920  
28921  
28922

| SCCS File Data Keywords |                              |              |                   |        |
|-------------------------|------------------------------|--------------|-------------------|--------|
| Keyword                 | Data Item                    | File Section | Value             | Format |
| :M:                     | Module name                  | "            | <i>text</i>       | S      |
| :FB:                    | Floor boundary               | "            | :R:               | S      |
| :CB:                    | Ceiling boundary             | "            | :R:               | S      |
| :Ds:                    | Default SID                  | "            | :I:               | S      |
| :ND:                    | Null delta flag              | "            | yes or no         | S      |
| :FD:                    | File descriptive text        | Comments     | <i>text</i>       | M      |
| :BD:                    | Body                         | Body         | <i>text</i>       | M      |
| :GB:                    | Gotten body                  | "            | <i>text</i>       | M      |
| :W:                     | A form of <i>what</i> string | N/A          | :Z::M:\t:I:       | S      |
| :A:                     | A form of <i>what</i> string | N/A          | :Z::Y: :M: :I::Z: | S      |
| :Z:                     | <i>what</i> string delimiter | N/A          | @( # )            | S      |
| :F:                     | SCCS file name               | N/A          | <i>text</i>       | S      |
| :PN:                    | SCCS file path name          | N/A          | <i>text</i>       | S      |

28923 \* :Dt::DT: :I: :D: :T: :P: :DS: :DP:  
28924 \*\* :R::L::B::S: if the delta is a branch delta (:BF:= =yes)  
28925 :R::L: if the delta is not a branch delta (:BF:= =no)

28926 **STDERR**

28927 Used only for diagnostic messages.

28928 **OUTPUT FILES**

28929 None.

28930 **EXTENDED DESCRIPTION**

28931 None.

28932 **EXIT STATUS**

28933 The following exit values shall be returned:

28934 0 Successful completion.

28935 >0 An error occurred.

28936 **CONSEQUENCES OF ERRORS**

28937 Default.

28938 **APPLICATION USAGE**

28939 None.

28940 **EXAMPLES**

28941 1. The following example:

28942 prs -d "User Names for :F: are:\n:UN:" s.file

28943 may write to standard output:

28944 User Names for s.file are:

28945 xyz

28946 131

28947 abc

28948 2. The following example:

28949 prs -d "Delta for pgm :M:: :I: - :D: By :P:" -r s.file

28950 may write to standard output:

28951 Delta for pgm main.c: 3.7 - 77/12/01 By cas

28952 3. As a special case:

28953 prs s.file

28954 may write to standard output:

28955 s.file:

28956 <blank line>

28957 D 1.1 77/12/01 00:00:00 cas 1 000000/00000/00000

28958 MRs:

28959 b178-12345

28960 b179-54321

28961 COMMENTS:

28962 this is the comment line for s.file initial delta

28963 <blank line>

28964 for each delta table entry of the **D** type. The only option allowed to be used with this

28965 special case is the **-a** option.

28966 **RATIONALE**

28967 None.

28968 **FUTURE DIRECTIONS**

28969 None.

28970 **SEE ALSO**

28971 *admin, delta, get, what*

28972 **CHANGE HISTORY**

28973 First released in Issue 2.

28974 **Issue 4**

28975 Format reorganized.

28976 Exceptions to Utility Syntax Guidelines conformance noted.

28977 Internationalized environment variable support mandated.

28978 **Issue 5**

28979 The phrase “in which keyword substitution is followed by a <newline>” is deleted from the end

28980 of the second paragraph of **Data Keywords** (on page 2956).

28981 The interpretation of the **YY** component of the **-c cutoff** argument is noted.

28982 **Issue 6**

28983 The normative text is reworded to emphasise the term “shall” for implementation requirements.

## 28984 NAME

28985 ps — report process status

## 28986 SYNOPSIS

28987 UP XSI ps [-aA][--defl][--G *grouplist*][--o *format*]...[-p *proclist*][-t *termlist*]28988 [-U *userlist*][--g *grouplist*][--n *namelist*][--u *userlist*]

28989

## 28990 DESCRIPTION

28991 The *ps* utility shall write information about processes, subject to having the appropriate  
28992 privileges to obtain information about those processes.28993 By default, *ps* selects all processes with the same effective user ID as the current user and the  
28994 same controlling terminal as the invoker.

## 28995 OPTIONS

28996 The *ps* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
28997 Utility Syntax Guidelines.

28998 The following options shall be supported:

28999 **-a** Write information for all processes associated with terminals. Implementations  
29000 may omit session leaders from this list.29001 **-A** Write information for all processes.29002 XSI **-d** Write information for all processes, except session leaders.29003 XSI **-e** Write information for all processes. (Equivalent to **-A**.)29004 XSI **-f** Generate a **full** listing. (See the STDOUT section for the contents of a **full** listing.)29005 XSI **-g *grouplist*** Write information for processes whose session leaders are given in *grouplist*. The  
29006 application shall ensure that the *grouplist* is a single argument in the form of a  
29007 <blank> or comma-separated list.29008 **-G *grouplist*** Write information for processes whose real group ID numbers are given in  
29009 *grouplist*. The application shall ensure that the *grouplist* is a single argument in the  
29010 form of a <blank> or comma-separated list.29011 XSI **-l** Generate a **long** listing. (See the STDOUT section for the contents of a **long** listing.)

29012

29013 XSI **-n *namelist*** Specify the name of an alternative system *namelist* file in place of the default. The  
29014 name of the default file and the format of a *namelist* file are unspecified.29015 **-o *format*** Write information according to the format specification given in *format*. This is  
29016 fully described in the STDOUT section. Multiple **-o** options can be specified; the  
29017 format specification shall be interpreted as the <space> character-separated  
29018 concatenation of all the *format* option-arguments.29019 **-p *proclist*** Write information for processes whose process ID numbers are given in *proclist*.  
29020 The application shall ensure that the *proclist* is a single argument in the form of a  
29021 <blank> or comma-separated list.29022 **-t *termlist*** Write information for processes associated with terminals given in *termlist*. The  
29023 application shall ensure that the *termlist* is a single argument in the form of a  
29024 <blank> or comma-separated list. Terminal identifiers shall be given in an  
29025 XSI implementation-defined format. On XSI-conformant systems, they shall be given  
29026 in one of two forms: the device's file name (for example, **tty04**) or, if the device's

- 29027 file name starts with **tty**, just the identifier following the characters **tty** (for  
29028 example, "04").
- 29029 XSI **-u *userlist*** Write information for processes whose user ID numbers or login names are given  
29030 in *userlist*. The application shall ensure that the *userlist* is a single argument in the  
29031 form of a <blank> or comma-separated list. In the listing, the numerical user ID is  
29032 written unless the **-f** option is used, in which case the login name is written.
- 29033 **-U *userlist*** Write information for processes whose real user ID numbers or login names are  
29034 given in *userlist*. The application shall ensure that the *userlist* is a single argument  
29035 in the form of a <blank> or comma-separated list.
- 29036 With the exception of **-o *format***, all of the options shown are used to select processes. If any are  
29037 specified, the default list shall be ignored and *ps* shall select the processes represented by the  
29038 bitwise-inclusive OR of all the selection-criteria options.
- 29039 **OPERANDS**
- 29040 None.
- 29041 **STDIN**
- 29042 Not used.
- 29043 **INPUT FILES**
- 29044 None.
- 29045 **ENVIRONMENT VARIABLES**
- 29046 The following environment variables shall affect the execution of *ps*:
- 29047 **COLUMNS** Override the system-selected horizontal screen size, used to determine the number  
29048 of text columns to display. See the Base Definitions volume of  
29049 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid values and  
29050 results when it is unset or null.
- 29051 **LANG** Provide a default value for the internationalization variables that are unset or null.  
29052 If *LANG* is unset or null, the corresponding value from the implementation-  
29053 defined default locale shall be used. If any of the internationalization variables  
29054 contains an invalid setting, the utility shall behave as if none of the variables had  
29055 been defined.
- 29056 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
29057 internationalization variables.
- 29058 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
29059 characters (for example, single-byte as opposed to multi-byte characters in  
29060 arguments).
- 29061 **LC\_MESSAGES**
- 29062 Determine the locale that should be used to affect the format and contents of  
29063 diagnostic messages written to standard error and informative messages written to  
29064 standard output.
- 29065 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 29066 **LC\_TIME** Determine the format and contents of the date and time strings displayed.
- 29067 **ASYNCHRONOUS EVENTS**
- 29068 Default.

29069 **STDOUT**

29070 When the **-o** option is not specified, the standard output format is unspecified.

29071 XSI On XSI-conformant systems, the output format is as follows. The column headings and  
 29072 descriptions of the columns in a *ps* listing are given below. The precise meanings of these fields  
 29073 are implementation-defined. The letters 'f' and 'l' (below) indicate the option (**full** or **long**)  
 29074 that shall cause the corresponding heading to appear; **all** means that the heading always  
 29075 appears. Note that these two options determine only what information is provided for a process;  
 29076 they do not determine which processes are listed.

|       |              |       |                                                                                                      |
|-------|--------------|-------|------------------------------------------------------------------------------------------------------|
| 29077 | <b>F</b>     | (l)   | Flags (octal and additive) associated with the process.                                              |
| 29078 | <b>S</b>     | (l)   | The state of the process.                                                                            |
| 29079 | <b>UID</b>   | (f,l) | The user ID number of the process owner; the login name is printed<br>under the <b>-f</b> option.    |
| 29081 | <b>PID</b>   | (all) | The process ID of the process; it is possible to kill a process if this<br>datum is known.           |
| 29082 |              |       |                                                                                                      |
| 29083 | <b>PPID</b>  | (f,l) | The process ID of the parent process.                                                                |
| 29084 | <b>C</b>     | (f,l) | Processor utilization for scheduling.                                                                |
| 29085 | <b>PRI</b>   | (l)   | The priority of the process; higher numbers mean lower priority.                                     |
| 29086 | <b>NI</b>    | (l)   | Nice value; used in priority computation.                                                            |
| 29087 | <b>ADDR</b>  | (l)   | The address of the process.                                                                          |
| 29088 | <b>SZ</b>    | (l)   | The size in blocks of the core image of the process.                                                 |
| 29089 | <b>WCHAN</b> | (l)   | The event for which the process is waiting or sleeping; if blank, the<br>process is running.         |
| 29090 |              |       |                                                                                                      |
| 29091 | <b>STIME</b> | (f)   | Starting time of the process.                                                                        |
| 29092 | <b>TTY</b>   | (all) | The controlling terminal for the process.                                                            |
| 29093 | <b>TIME</b>  | (all) | The cumulative execution time for the process.                                                       |
| 29094 | <b>CMD</b>   | (all) | The command name; the full command name and its arguments are<br>written under the <b>-f</b> option. |
| 29095 |              |       |                                                                                                      |

29096 A process that has exited and has a parent, but has not yet been waited for by the parent, is  
 29097 marked **defunct**.

29098 Under the option **-f**, *ps* tries to determine the command name and arguments given when the  
 29099 process was created by examining memory or the swap area. Failing this, the command name, as  
 29100 it would appear without the option **-f**, is written in square brackets.

29101 The **-o** option allows the output format to be specified under user control.

29102 The application shall ensure that the format specification is a list of names presented as a single  
 29103 argument, <blank> or comma-separated. Each variable has a default header. The default header  
 29104 can be overridden by appending an equals sign and the new text of the header. The rest of the  
 29105 characters in the argument shall be used as the header text. The fields specified shall be written  
 29106 in the order specified on the command line, and should be arranged in columns in the output.  
 29107 The field widths shall be selected by the system to be at least as wide as the header text (default  
 29108 or overridden value). If the header text is null, such as **-o user=**, the field width shall be at least  
 29109 as wide as the default header text. If all header text fields are null, no header line shall be  
 29110 written.

29111 The following names are recognized in the POSIX locale:

29112 **ruser** The real user ID of the process. This shall be the textual user ID, if it can be obtained  
 29113 and the field width permits, or a decimal representation otherwise.



|       |               |                                                                                                                   |
|-------|---------------|-------------------------------------------------------------------------------------------------------------------|
| 29114 | <b>user</b>   | The effective user ID of the process. This shall be the textual user ID, if it can be                             |
| 29115 |               | obtained and the field width permits, or a decimal representation otherwise.                                      |
| 29116 | <b>rgroup</b> | The real group ID of the process. This shall be the textual group ID, if it can be obtained                       |
| 29117 |               | and the field width permits, or a decimal representation otherwise.                                               |
| 29118 | <b>group</b>  | The effective group ID of the process. This shall be the textual group ID, if it can be                           |
| 29119 |               | obtained and the field width permits, or a decimal representation otherwise.                                      |
| 29120 | <b>pid</b>    | The decimal value of the process ID.                                                                              |
| 29121 | <b>ppid</b>   | The decimal value of the parent process ID.                                                                       |
| 29122 | <b>pgid</b>   | The decimal value of the process group ID.                                                                        |
| 29123 | <b>pcpu</b>   | The ratio of CPU time used recently to CPU time available in the same period,                                     |
| 29124 |               | expressed as a percentage. The meaning of “recently” in this context is unspecified. The                          |
| 29125 |               | CPU time available is determined in an unspecified manner.                                                        |
| 29126 | <b>vsz</b>    | The size of the process in (virtual) memory in kilobytes as a decimal integer.                                    |
| 29127 | <b>nice</b>   | The decimal value of the nice value of the process; see <i>nice</i> (on page 2872).                               |
| 29128 | <b>etime</b>  | In the POSIX locale, the elapsed time since the process was started, in the form:                                 |
| 29129 |               | [ [ <i>dd</i> -] <i>hh</i> : ] <i>mm</i> : <i>ss</i>                                                              |
| 29130 |               | where <i>dd</i> shall represent the number of days, <i>hh</i> the number of hours, <i>mm</i> the number           |
| 29131 |               | of minutes, and <i>ss</i> the number of seconds. The <i>dd</i> field shall be a decimal integer. The              |
| 29132 |               | <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be two-digit decimal integers padded on the left with zeros.   |
| 29133 | <b>time</b>   | In the POSIX locale, the cumulative CPU time of the process in the form:                                          |
| 29134 |               | [ [ <i>dd</i> -] <i>hh</i> : <i>mm</i> : <i>ss</i>                                                                |
| 29135 |               | The <i>dd</i> , <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be as described in the <b>etime</b> specifier. |
| 29136 | <b>tty</b>    | The name of the controlling terminal of the process (if any) in the same format used by                           |
| 29137 |               | the <i>who</i> utility.                                                                                           |
| 29138 | <b>comm</b>   | The name of the command being executed ( <i>argv</i> [0] value) as a string.                                      |
| 29139 | <b>args</b>   | The command with all its arguments as a string. The implementation may truncate this                              |
| 29140 |               | value to the field width; it is implementation-defined whether any further truncation                             |
| 29141 |               | occurs. It is unspecified whether the string represented is a version of the argument list                        |
| 29142 |               | as it was passed to the command when it started, or is a version of the arguments as                              |
| 29143 |               | they may have been modified by the application. Applications cannot depend on being                               |
| 29144 |               | able to modify their argument list and having that modification be reflected in the                               |
| 29145 |               | output of <i>ps</i> .                                                                                             |
| 29146 |               | Any field need not be meaningful in all implementations. In such a case a hyphen (‘-’) should                     |
| 29147 |               | be output in place of the field value.                                                                            |
| 29148 |               | Only <b>comm</b> and <b>args</b> shall be allowed to contain <blank> characters; all others shall not. Any        |
| 29149 |               | implementation-defined variables shall be specified in the system documentation along with the                    |
| 29150 |               | default header and indicating if the field may contain <blank> characters.                                        |
| 29151 |               | The following table specifies the default header to be used in the POSIX locale corresponding to                  |
| 29152 |               | each format specifier.                                                                                            |

29153

Table 4-17 Variable Names and Default Headers in *ps*

29154

| Format Specifier | Default Header | Format Specifier | Default Header |
|------------------|----------------|------------------|----------------|
| <b>args</b>      | <b>COMMAND</b> | <b>ppid</b>      | <b>PPID</b>    |
| <b>comm</b>      | <b>COMMAND</b> | <b>rgroup</b>    | <b>RGROUP</b>  |
| <b>etime</b>     | <b>ELAPSED</b> | <b>ruser</b>     | <b>RUSER</b>   |
| <b>group</b>     | <b>GROUP</b>   | <b>time</b>      | <b>TIME</b>    |
| <b>nice</b>      | <b>NI</b>      | <b>tty</b>       | <b>TT</b>      |
| <b>pcpu</b>      | <b>%CPU</b>    | <b>user</b>      | <b>USER</b>    |
| <b>pgid</b>      | <b>PGID</b>    | <b>vsz</b>       | <b>VSZ</b>     |
| <b>pid</b>       | <b>PID</b>     |                  |                |

29155

29156

29157

29158

29159

29160

29161

29162

**29163 STDERR**

29164 Used only for diagnostic messages.

**29165 OUTPUT FILES**

29166 None.

**29167 EXTENDED DESCRIPTION**

29168 None.

**29169 EXIT STATUS**

29170 The following exit values shall be returned:

29171 0 Successful completion.

29172 &gt;0 An error occurred.

**29173 CONSEQUENCES OF ERRORS**

29174 Default.

**29175 APPLICATION USAGE**29176 Things can change while *ps* is running; the snapshot it gives is only true for an instant, and might  
29177 not be accurate by the time it is displayed.29178 The **args** format specifier is allowed to produce a truncated version of the command arguments.  
29179 In some implementations, this information is no longer available when the *ps* utility is executed.29180 If the field width is too narrow to display a textual ID, the system may use a numeric version.  
29181 Normally, the system would be expected to choose large enough field widths, but if a large  
29182 number of fields were selected to write, it might squeeze fields to their minimum sizes to fit on  
29183 one line. One way to ensure adequate width for the textual IDs is to override the default header  
29184 for a field to make it larger than most or all user or group names.29185 There is no special quoting mechanism for header text. The header text is the rest of the  
29186 argument. If multiple header changes are needed, multiple **-o** options can be used, such as:29187 `ps -o "user=User Name" -o pid=Process\ ID`29188 On some systems, especially multi-level secure systems, *ps* may be severely restricted and  
29189 produce information only about child processes owned by the user.**29190 EXAMPLES**

29191 The command:

29192 `ps -o user,pid,ppid=MOM -o args`

29193 writes at least the following in the POSIX locale:

29194 USER PID MOM COMMAND

29195 helene 34 12 ps -o uid,pid,ppid=MOM -o args

29196 The contents of the **COMMAND** field need not be the same in all implementations, due to  
29197 possible truncation.

29198 **RATIONALE**

29199 There is very little commonality between BSD and System V implementations of *ps*. Many  
29200 options conflict or have subtly different usages. The standard developers attempted to select a  
29201 set of options that were useful on a wide range of systems and selected options that either can be  
29202 implemented on both BSD and System V-based systems without breaking the current  
29203 implementations or where the options are sufficiently similar that any changes would not be  
29204 unduly problematic for users or implementors.

29205 It is recognized that on some systems, especially multi-level secure systems, *ps* may be nearly  
29206 useless. The default output has therefore been chosen such that it does not break historical  
29207 implementations and also is likely to provide at least some useful information on most systems.

29208 The major change is the addition of the format specification capability. The motivation for this  
29209 invention is to provide a mechanism for users to access a wider range of system information, if  
29210 the system permits it, in a portable manner. The fields chosen to appear in this volume of  
29211 IEEE Std. 1003.1-200x were arrived at after considering what concepts were likely to be both  
29212 reasonably useful to the “average” user and had a reasonable chance of being implemented on a  
29213 wide range of systems. Again it is recognized that not all systems are able to provide all the  
29214 information and, conversely, some may wish to provide more. It is hoped that the approach  
29215 adopted will be sufficiently flexible and extensible to accommodate most systems.  
29216 Implementations may be expected to introduce new format specifiers.

29217 The default output should consist of a short listing containing the process ID, terminal name,  
29218 cumulative execution time, and command name of each process.

29219 The preference of the standard developers would have been to make the format specification an  
29220 operand of the *ps* command. Unfortunately, BSD usage precluded this.

29221 At one time a format was included to display the environment array of the process. This was  
29222 deleted because there is no portable way to display it.

29223 The **-A** option is equivalent to the BSD **-g** and the SVID **-e**. Because the two systems differed, a  
29224 mnemonic compromise was selected.

29225 The **-a** option is described with some optional behavior because the SVID omits session leaders,  
29226 but BSD does not.

29227 In an early proposal, format specifiers appeared for priority and start time. The former was not  
29228 defined adequately in this volume of IEEE Std. 1003.1-200x and was removed in deference to the  
29229 defined nice value; the latter because elapsed time was considered to be more useful.

29230 In a new BSD version of *ps*, a **-O** option can be used to write all of the default information,  
29231 followed by additional format specifiers. This was not adopted because the default output is  
29232 implementation-defined. Nevertheless, this is a useful option that should be reserved for that  
29233 purpose. In the **-o** option for the POSIX Shell and Utilities *ps*, the format is the concatenation of  
29234 each **-o**. Therefore, the user can have an alias or function that defines the beginning of their  
29235 desired format and add more fields to the end of the output in certain cases where that would be  
29236 useful.

29237 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
29238 require that they all use the same format.

29239 The **pcpu** field indicates that the CPU time available is determined in an unspecified manner.  
29240 This is because it is difficult to express an algorithm that is useful across all possible machine  
29241 architectures. Historical counterparts to this value have attempted to show percentage of use in

29242 the recent past, such as the preceding minute. Frequently, these values for all processes did not  
29243 add up to 100%. Implementations are encouraged to provide data in this field to users that will  
29244 help them identify processes currently affecting the performance of the system.

29245 **FUTURE DIRECTIONS**

29246 None.

29247 **SEE ALSO**

29248 *kill, nice, renice*

29249 **CHANGE HISTORY**

29250 First released in Issue 2.

29251 **Issue 4**

29252 Aligned with the ISO/IEC 9945-2:1993 standard.

29253 **Issue 6**

29254 This utility is now marked as part of the User Portability Utilities option.

29255 The normative text is reworded to avoid use of the term “must” for application requirements.

29256 **NAME**

29257           pwd — return working directory name

29258 **SYNOPSIS**

29259           pwd [-L | -P ]

29260 **DESCRIPTION**29261           The *pwd* utility shall write to standard output an absolute path name of the current working  
29262           directory, which does not contain the file names dot or dot-dot.29263 **OPTIONS**29264           The *pwd* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
29265           12.2, Utility Syntax Guidelines.

29266           The following options shall be supported by the implementation:

29267           **-L**           If the *PWD* environment variable contains an absolute path name of the current  
29268           directory that does not contain the file names dot or dot-dot, *pwd* shall write this  
29269           path name to standard output. Otherwise, the **-L** option shall behave as the **-P**  
29270           option.29271           **-P**           The absolute path name written shall not contain file names that, in the context of  
29272           the path name, refer to files of type symbolic link.29273           If both **-L** and **-P** are specified, the last one shall apply. If neither **-L** nor **-P** is specified, the *pwd*  
29274           utility shall behave as if **-L** had been specified.29275 **OPERANDS**

29276           None.

29277 **STDIN**

29278           Not used.

29279 **INPUT FILES**

29280           None.

29281 **ENVIRONMENT VARIABLES**29282           The following environment variables shall affect the execution of *pwd*:29283           **LANG**           Provide a default value for the internationalization variables that are unset or null.  
29284           If *LANG* is unset or null, the corresponding value from the implementation-  
29285           defined default locale shall be used. If any of the internationalization variables  
29286           contains an invalid setting, the utility shall behave as if none of the variables had  
29287           been defined.29288           **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
29289           internationalization variables.29290           **LC\_MESSAGES**29291           Determine the locale that should be used to affect the format and contents of  
29292           diagnostic messages written to standard error.29293 **XSI**           **NLSPATH**       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.29294           **PWD**           If the **-P** option is in effect, this variable shall be set to an absolute path name of  
29295           the current working directory that does not contain any components that specify  
29296           symbolic links, does not contain any components that are dot, and does not  
29297           contain any components that are dot-dot. If an application sets or unsets the value  
29298           of *PWD*, the behavior of *pwd* is unspecified.

29299 **ASYNCHRONOUS EVENTS**

29300 Default.

29301 **STDOUT**29302 The *pwd* utility output is an absolute path name of the current working directory:

29303 "%s\n", &lt;directory pathname&gt;

29304 **STDERR**

29305 Used only for diagnostic messages.

29306 **OUTPUT FILES**

29307 None.

29308 **EXTENDED DESCRIPTION**

29309 None.

29310 **EXIT STATUS**

29311 The following exit values shall be returned:

29312 0 Successful completion.

29313 &gt;0 An error occurred.

29314 **CONSEQUENCES OF ERRORS**

29315 If an error is detected, output shall not be written to standard output, a diagnostic message shall

29316 be written to standard error, and the exit status is not zero.

29317 **APPLICATION USAGE**

29318 None.

29319 **EXAMPLES**

29320 None.

29321 **RATIONALE**29322 Some implementations have historically provided *pwd* as a shell special built-in command.

29323 In most utilities, if an error occurs, partial output may be written to standard output. This does  
29324 not happen in historical implementations of *pwd*. Because *pwd* is frequently used in historical  
29325 shell scripts without checking the exit status, it is important that the historical behavior is  
29326 required here; therefore, the CONSEQUENCES OF ERRORS section specifically disallows any  
29327 partial output being written to standard output.

29328 **FUTURE DIRECTIONS**

29329 None.

29330 **SEE ALSO**29331 *cd*, the System Interfaces volume of IEEE Std. 1003.1-200x, *getcwd()*29332 **CHANGE HISTORY**

29333 First released in Issue 2.

29334 **Issue 4**

29335 Aligned with the ISO/IEC 9945-2: 1993 standard.

29336 **Issue 6**

29337 The **-P** and **-L** options are added to describe actions relating to symbolic links as specified in the  
29338 IEEE P1003.2b draft standard.

## 29339 NAME

29340 qalter — alter batch job

## 29341 SYNOPSIS

```
29342 BE qalter [-a date_time][-A account_string][-c interval][-e path_name]
29343 [-h hold_list][-j join_list][-k keep_list][-l resource_list]
29344 [-m mail_options][-M mail_list][-N name][-o path_name]
29345 [-p priority][-r y|n][-S path_name_list][-u user_list]
29346 job_identifier ...
29347
```

## 29348 DESCRIPTION

29349 The attributes of a batch job are altered by a request to the batch server that manages the batch  
 29350 job. The *qalter* utility is a user-accessible batch client that requests the alteration of the attributes  
 29351 of one or more batch jobs.

29352 The *qalter* utility shall alter the attributes of those batch jobs, and only those batch jobs, for which  
 29353 a batch *job\_identifier* is presented to the utility.

29354 The *qalter* utility shall alter the attributes of batch jobs in the order in which the batch  
 29355 *job\_identifiers* are presented to the utility.

29356 If the *qalter* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 29357 process the remaining batch *job\_identifiers*, if any.

29358 For each batch *job\_identifier* for which the *qalter* utility succeeds, each attribute of the identified  
 29359 batch job shall be altered as indicated by all the options presented to the utility.

29360 For each identified batch job for which the *qalter* utility fails, the utility shall not alter any  
 29361 attribute of the batch job.

29362 For each batch job that the *qalter* utility processes, the utility shall not modify any attribute other  
 29363 than those required by the options and option-arguments presented to the utility.

29364 The *qalter* utility shall alter batch jobs by sending a *Modify Job Request* to the batch server that  
 29365 manages each batch job. At the time the *qalter* utility exits, it shall have modified the batch job  
 29366 corresponding to each successfully processed batch *job\_identifier*. An attempt to alter the  
 29367 attributes of a batch job in the RUNNING state is implementation-defined.

## 29368 OPTIONS

29369 The *qalter* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 29370 12.2, Utility Syntax Guidelines.

29371 The following options shall be supported by the implementation:

29372 **-a *date\_time*** Redefine the time at which the batch job becomes eligible for execution.

29373 The *qalter* utility shall accept an option-argument that conforms to the syntax of  
 29374 the *date\_time* operand of the *touch* utility.

29375 The *qalter* utility shall set the *Execution\_Time* attribute of the batch job to the  
 29376 number of seconds since the Epoch that is equivalent to the local time expressed  
 29377 by the value of the *date\_time* option-argument. Specifying a *date\_time* option-  
 29378 argument that represents a time (number of seconds since the Epoch) earlier than  
 29379 the time at which the utility exits shall have the same effect on batch job execution  
 29380 as if the **-a** option had not been presented to the utility.

29381 **-A *account\_string***

29382 Redefine the account to which the resource consumption of the batch job should be  
 29383 charged.

- 29384 The syntax of the *account\_string* option-argument is unspecified.
- 29385 The *qalter* utility shall set the *Account\_Name* attribute of the batch job to the value  
29386 of the *account\_string* option-argument.
- 29387 **-c interval** Redefine whether the batch job should be checkpointed, and if so, how often.
- 29388 The *qalter* utility shall accept a value for the interval option-argument that is one of  
29389 the following:
- 29390 *n* No checkpointing is to be performed on the batch batch job  
29391 (NO\_CHECKPOINT).
- 29392 *s* Checkpointing is to be performed only when the batch server is shut  
29393 down (CHECKPOINT\_AT\_SHUTDOWN).
- 29394 *c* Automatic periodic checkpointing is to be performed at the  
29395 *Minimum\_Cpu\_Interval* attribute of the batch queue, in units of CPU  
29396 minutes (CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL).
- 29397 *c=minutes* Automatic periodic checkpointing is to be performed every *minutes*  
29398 of CPU time, or every *Minimum\_Cpu\_Interval* minutes, whichever is  
29399 greater. The *minutes* argument shall conform to the syntax for  
29400 unsigned integers and shall be greater than zero.
- 29401 An implementation may define other checkpoint intervals. The conformance  
29402 document for an implementation shall describe any alternative checkpoint  
29403 intervals, how they are specified, their internal behavior, and how they affect the  
29404 behavior of the utility.
- 29405 The *qalter* utility shall set the *Checkpoint* attribute of the batch job to the value of the  
29406 *interval* option-argument.
- 29407 **-e path\_name** Redefine the path to be used for the standard error stream of the batch job.
- 29408 The *qalter* utility shall accept a *path\_name* option-argument that conforms to the  
29409 syntax of the *path\_name* element defined in the POSIX.1-1990 standard, which can  
29410 be preceded by a host name element of the form *hostname:*.
- 29411 If the *path\_name* option-argument constitutes an absolute path name, the *qalter*  
29412 utility shall set the *Error\_Path* attribute of the batch job to the value of the  
29413 *path\_name* option-argument, including the host name element, if present.
- 29414 If the *path\_name* option-argument constitutes a relative path name and no host  
29415 name element is specified, the *qalter* utility shall set the *Error\_Path* attribute of the  
29416 batch job to the value of the absolute path name derived by expanding the  
29417 *path\_name* option-argument relative to the current directory of the process that  
29418 executes the *qalter* utility.
- 29419 If the *path\_name* option-argument constitutes a relative path name and a host name  
29420 element is specified, the *qalter* utility shall set the *Error\_Path* attribute of the batch  
29421 job to the value of the option-argument without expansion.
- 29422 If the *path\_name* option-argument does not include a host name element, the *qalter*  
29423 utility shall prefix the path name in the *Error\_Path* attribute with *hostname:*, where  
29424 *hostname* is the name of the host upon which the *qalter* utility is being executed.
- 29425 **-h hold\_list** Redefine the types of holds, if any, on the batch job. The *qalter* **-h** option shall  
29426 accept a value for the *hold\_list* option-argument that is a string of alphanumeric  
29427 characters in the portable character set.



29428 The *qalter* utility shall accept a value for the *hold\_list* option-argument that is a  
 29429 string of one or more of the characters 'u', 's', or 'o', or the single character  
 29430 'n'. For each unique character in the *hold\_list* option-argument, the *qalter* utility  
 29431 shall add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 29432 representing a different hold type:

29433 *u* USER

29434 *s* SYSTEM

29435 *o* OPERATOR

29436 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 29437 duplicates shall be ignored. An existing *Hold\_Types* attribute can be cleared by the  
 29438 hold type:

29439 *n* NO\_HOLD

29440 The *qalter* utility shall consider it an error if any hold type other than **n** is combined  
 29441 with hold type **n**. Strictly conforming applications shall not repeat any of the  
 29442 characters 'u', 's', 'o', or 'n' within the *hold\_list* option-argument. The *qalter*  
 29443 utility shall permit the repetition of characters, but shall not assign additional  
 29444 meaning to the repeated characters. An implementation may define other hold  
 29445 types. The conformance document for an implementation shall describe any  
 29446 additional hold types, how they are specified, their internal behavior, and how  
 29447 they affect the behavior of the utility.

29448 **-j** *join\_list* Redefine which streams of the batch job are to be merged. The *qalter* **-j** option shall  
 29449 accept a value for the *join\_list* option-argument that is a string of alphanumeric  
 29450 characters in the portable character set.

29451 The *qalter* utility shall accept a *join\_list* option-argument that consists of one or  
 29452 more of the characters 'e' and 'o', or the single character 'n'.

29453 All of the other batch job output streams specified shall be merged into the output  
 29454 stream represented by the character listed first in the *join\_list* option-argument.

29455 For each unique character in the *join\_list* option-argument, the *qalter* utility shall  
 29456 add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 29457 a different batch job stream to join:

29458 *e* The standard error of the batch batch job (JOIN\_STD\_ERROR).

29459 *o* The standard output of the batch batch job (JOIN\_STD\_OUTPUT).

29460 An existing *Join\_Path* attribute can be cleared by the join type:

29461 **n** NO\_JOIN

29462 If **n** is specified, then no files are joined. The *qalter* utility shall consider it an error if  
 29463 any join type other than **n** is combined with join type **n**.

29464 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 29465 'n' within the *join\_list* option-argument. The *qalter* utility shall permit the  
 29466 repetition of characters, but shall not assign additional meaning to the repeated  
 29467 characters.

29468 An implementation may define other join types. The conformance document for an  
 29469 implementation shall describe any additional batch job streams, how they are  
 29470 specified, their internal behavior, and how they affect the behavior of the utility.

29471        **-k *keep\_list***   Redefine which output of the batch job to retain on the execution host.

29472                    The *qalter -k* option shall accept a value for the *keep\_list* option-argument that is a  
29473                    string of alphanumeric characters in the portable character set.

29474                    The *qalter* utility shall accept a *keep\_list* option-argument that consists of one or  
29475                    more of the characters 'e' and 'o' or the single character 'n'.

29476                    For each unique character in the *keep\_list* option-argument, the *qalter* utility shall  
29477                    add a value to the *Keep\_Files* attribute of the batch job as follows, each representing  
29478                    a different batch job stream to keep:

29479                    **e**    The standard error of the batch batch job (KEEP\_STD\_ERROR).

29480                    **o**    The standard output of the batch batch job (KEEP\_STD\_OUTPUT).

29481                    If both 'e' and 'o' are specified, then both files are retained. An existing  
29482                    *Keep\_Files* attribute can be cleared by the keep type:

29483                    **n**    NO\_KEEP

29484                    If **n** is specified, then no files are retained. The *qalter* utility shall consider it an error  
29485                    if any keep type other than **n** is combined with keep type **n**.

29486                    Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
29487                    'n' within the *keep\_list* option-argument. The *qalter* utility shall permit the  
29488                    repetition of characters, but shall not assign additional meaning to the repeated  
29489                    characters. An implementation may define other keep types. The conformance  
29490                    document for an implementation shall describe any additional keep types, how  
29491                    they are specified, their internal behavior, and how they affect the behavior of the  
29492                    utility.

29493        **-l *resource\_list***

29494                    Redefine the resources that are allowed or required by the batch job.

29495                    The *qalter* utility shall accept a *resource\_list* option-argument that conforms to the  
29496                    following syntax:

29497                    resource=value[ , , resource=value , , . . . ]

29498                    The *qalter* utility shall set one entry in the value of the *Resource\_List* attribute of the  
29499                    batch job for each resource listed in the *resource\_list* option-argument.

29500                    Because the list of supported resource names might vary by batch server, the *qalter*  
29501                    utility shall rely on the batch server to validate the resource names and associated  
29502                    values. See Section 3.3.3 (on page 2337) for a means of removing *keyword=value*  
29503                    (and *value@keyword*) pairs and other general rules for list-oriented batch job  
29504                    attributes.

29505        **-m *mail\_options***

29506                    Redefine the points in the execution of the batch job at which the batch server is to  
29507                    send mail about a change in the state of the batch job.

29508                    The *qalter -m* option shall accept a value for the *mail\_options* option-argument that  
29509                    is a string of alphanumeric characters in the portable character set.

29510                    The *qalter* utility shall accept a value for the *mail\_options* option-argument that is a  
29511                    string of one or more of the characters 'e', 'b', and 'a', or the single character  
29512                    'n'. For each unique character in the *mail\_options* option-argument, the *qalter*  
29513                    utility shall add a value to the *Mail\_Users* attribute of the batch job as follows, each  
29514                    representing a different time during the life of a batch job at which to send mail:

29515            **e**   MAIL\_AT\_EXIT

29516            **b**   MAIL\_AT\_BEGINNING

29517            **a**   MAIL\_AT\_ABORT

29518            If any of these characters are duplicated in the *mail\_options* option-argument, the  
29519            duplicates shall be ignored.

29520            An existing *Mail\_Points* attribute can be cleared by the mail type:

29521            **n**   NO\_MAIL

29522            If **n** is specified, then mail is not sent. The *qalter* utility shall consider it an error if  
29523            any mail type other than **n** is combined with mail type **n**. Strictly conforming  
29524            applications shall not repeat any of the characters 'e', 'b', 'a', or 'n' within  
29525            the *mail\_options* option-argument. The *qalter* utility shall permit the repetition of  
29526            characters but shall not assign additional meaning to the repeated characters.

29527            An implementation may define other mail types. The conformance document for  
29528            an implementation shall describe any additional mail types, how they are  
29529            specified, their internal behavior, and how they affect the behavior of the utility.

29530            **-M mail\_list** Redefine the list of users to which the batch server that executes the batch job is to  
29531            send mail, if the batch server sends mail about the batch job.

29532            The syntax of the *mail\_list* option-argument is unspecified. If the implementation  
29533            of the *qalter* utility uses a name service to locate users, the utility shall accept the  
29534            syntax used by the name service.

29535            If the implementation of the *qalter* utility does not use a name service to locate  
29536            users, the implementation shall accept the following syntax for user names:

29537            mail\_address[ , mail\_address , ... ]

29538            The interpretation of *mail\_address* is implementation-defined.

29539            The *qalter* utility shall set the *Mail\_Users* attribute of the batch job to the value of  
29540            the *mail\_list* option-argument.

29541            **-N name**   Redefine the name of the batch job.

29542            The *qalter* **-N** option shall accept a value for the *name* option argument that is a  
29543            string of up to 15 alphanumeric characters in the portable character set where the  
29544            first character is alphabetic.

29545            The syntax of the *name* option-argument is unspecified.

29546            The *qalter* utility shall set the *Job\_Name* attribute of the batch job to the value of the  
29547            *name* option-argument.

29548            **-o path\_name** Redefine the path for the standard output of the batch job.

29549            The *qalter* utility shall accept a *path\_name* option-argument that conforms to the  
29550            syntax of the *path\_name* element defined in the POSIX.1-1990 standard, which can  
29551            be preceded by a host name element of the form *hostname*:

29552            If the *path\_name* option-argument constitutes an absolute path name, the *qalter*  
29553            utility shall set the *Output\_Path* attribute of the batch job to the value of the  
29554            *path\_name* option-argument.

29555            If the *path\_name* option-argument constitutes a relative path name and no host  
29556            name element is specified, the *qalter* utility shall set the *Output\_Path* attribute of the

29557 batch job to the absolute path name derived by expanding the *path\_name* option-  
 29558 argument relative to the current directory of the process that executes the *qalter*  
 29559 utility.

29560 If the *path\_name* option-argument constitutes a relative path name and a host name  
 29561 element is specified, the *qalter* utility shall set the *Output\_Path* attribute of the batch  
 29562 job to option-argument without any expansion of the path name.

29563 If the *path\_name* option-argument does not include a host name element, the *qalter*  
 29564 utility shall prefix the path name in the *Output\_Path* attribute with *hostname:*,  
 29565 where *hostname* is the name of the host upon which the *qalter* utility is being  
 29566 executed.

29567 **-p *priority*** Redefine the priority of the batch job.

29568 The *qalter* utility shall accept a value for the priority option-argument that  
 29569 conforms to the syntax for signed decimal integers, and which is not less than  
 29570 -1 024 and not greater than 1 023.

29571 The *qalter* utility shall set the *Priority* attribute of the batch job to the value of the  
 29572 *priority* option-argument.

29573 **-r *y* | *n*** Redefine whether the batch job is rerunable.

29574 If the value of the option-argument is *y*, the *qalter* utility shall set the *Rerunable*  
 29575 attribute of the batch job to TRUE.

29576 If the value of the option-argument is *n*, the *qalter* utility shall set the *Rerunable*  
 29577 attribute of the batch job to FALSE.

29578 The *qalter* utility shall consider it an error if any character other than 'y' or 'n' is  
 29579 specified in the option-argument.

29580 **-S *path\_name\_list***

29581 Redefine the shell that interprets the script at the destination system.

29582 The *qalter* utility shall accept a *path\_name\_list* option-argument that conforms to  
 29583 the following syntax:

29584 `pathname[@host][,pathname[@host],...]`

29585 The *qalter* utility shall accept only one path name that is missing a corresponding  
 29586 host name. The *qalter* utility shall allow only one path name per named host.

29587 The *qalter* utility shall add a value to the *Shell\_Path\_List* attribute of the batch job  
 29588 for each entry in the *path\_name\_list* option-argument. See Section 3.3.3 (on page  
 29589 2337) for a means of removing *keyword=value* (and *value@keyword*) pairs and other  
 29590 general rules for list-oriented batch job attributes.

29591 **-u *user\_list*** Redefine the user name under which the batch job is to run at the destination  
 29592 system.

29593 The *qalter* utility shall accept a *user\_list* option-argument that conforms to the  
 29594 following syntax:

29595 `username[@host][,username[@host],...]`

29596 The *qalter* utility shall accept only one user name that is missing a corresponding  
 29597 host name. The *qalter* utility shall accept only one user name per named host.

29598 The *qalter* utility shall add a value to the *User\_List* attribute of the batch job for each  
 29599 entry in the *user\_list* option-argument. See Section 3.3.3 (on page 2337) for a means

29600 of removing *keyword=value* (and *value@keyword*) pairs and other general rules for  
29601 list-oriented batch job attributes.

#### 29602 OPERANDS

29603 The *qalter* utility shall accept one or more operands that conform to the syntax for a batch  
29604 *job\_identifier* (see Section 3.3.1 (on page 2336)).

#### 29605 STDIN

29606 Not used.

#### 29607 INPUT FILES

29608 None.

#### 29609 ENVIRONMENT VARIABLES

29610 The following environment variables shall affect the execution of *qalter*:

29611 *LANG* Provide a default value for the internationalization variables that are unset or null.  
29612 If *LANG* is unset or null, the corresponding value from the implementation-  
29613 defined default locale shall be used. If any of the internationalization variables  
29614 contains an invalid setting, the utility shall behave as if none of the variables had  
29615 been defined.

29616 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
29617 internationalization variables.

29618 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
29619 characters (for example, single-byte as opposed to multi-byte characters in  
29620 arguments).

#### 29621 LC\_MESSAGES

29622 Determine the locale that should be used to affect the format and contents of  
29623 diagnostic messages written to standard error.

29624 *LC\_TIME* Determine the format and contents of date and time strings written by *qalter*.

29625 *LOGNAME* Determine the login name of the user.

29626 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
29627 is not set, an unspecified system default timezone is used.

#### 29628 ASYNCHRONOUS EVENTS

29629 Default.

#### 29630 STDOUT

29631 None.

#### 29632 STDERR

29633 Used only for diagnostic messages.

#### 29634 OUTPUT FILES

29635 None.

#### 29636 EXTENDED DESCRIPTION

29637 None.

#### 29638 EXIT STATUS

29639 The following exit values shall be returned:

29640 0 Successful completion.

29641 >0 An error occurred.

## 29642 CONSEQUENCES OF ERRORS

29643 In addition to the default behavior, the *qalter* utility shall not be required to write a diagnostic  
29644 message to standard error when the error reply received from a batch server indicates that the  
29645 batch *job\_identifier* does not exist on the server. Whether or not the *qalter* utility attempts to  
29646 locate the batch job on other batch servers is implementation-defined.

## 29647 APPLICATION USAGE

29648 None.

## 29649 EXAMPLES

29650 None.

## 29651 RATIONALE

29652 The *qalter* utility allows users to change the attributes of a batch job.

29653 As a means of altering a queued job, the *qalter* utility is superior to deleting and requeuing the  
29654 batch job insofar as an altered job retains its place in the queue with some traditional selection  
29655 algorithms. In addition, the *qalter* utility is both shorter and simpler than a sequence of *qdel* and  
29656 *qsub* utilities.

29657 The result of an attempt on the part of a user to alter a batch job in a RUNNING state is  
29658 implementation-defined because a batch job in the RUNNING state will already have opened its  
29659 output files and otherwise performed any actions indicated by the options in effect at the time  
29660 the batch job began execution.

29661 The options processed by the *qalter* utility are identical to those of the *qsub* utility, with a few  
29662 exceptions: **-V**, **-v**, and **-q**. The **-V** and **-v** are inappropriate for the *qalter* utility, since they  
29663 capture potentially transient environment information from the submitting process. The **-q**  
29664 option would specify a new queue, which would largely negate the previously stated advantage  
29665 of using *qalter*; furthermore, the *qmove* utility provides a superior means of moving jobs.

29666 Each of the following paragraphs provides the rationale for a *qalter* option.

29667 Additional rationale concerning these options can be found in the rationale for the *qsub* utility.

29668 The **-a** option allows users to alter the date and time at which a batch job becomes eligible to  
29669 run.

29670 The **-A** option allows users to change the account that will be charged for the resources  
29671 consumed by the batch job. Support for the **-A** option is mandatory for conforming  
29672 implementations of *qalter*, even though support of accounting is optional for servers. Whether or  
29673 not to support accounting is left to the implementor of the server, but mandatory support of the  
29674 **-A** option assures users of a consistent interface and allows them to control accounting on  
29675 servers that support accounting.

29676 The **-c** option allows users to alter the checkpointing interval of a batch job. A checkpointing  
29677 system, which is not defined by IEEE Std. 1003.1-200x, allows recovery of a batch job at the most  
29678 recent checkpoint in the event of a crash. Checkpointing is typically used for jobs that consume  
29679 expensive computing time or must meet a critical schedule. Users should be allowed to make  
29680 the tradeoff between the overhead of checkpointing and the risk to the timely completion of the  
29681 batch job; therefore, this volume of IEEE Std. 1003.1-200x provides the checkpointing interval  
29682 option. Support for checkpointing is optional for servers.

29683 The **-e** option allows users to alter the name and location of the standard error stream written by  
29684 a batch job. However, the path of the standard error stream is meaningless if the value of the  
29685 *Join\_Path* attribute of the batch job is TRUE.

29686 The **-h** option allows users to set the hold type in the *Hold\_Types* attribute of a batch job. The  
29687 *qhold* and *qrls* utilities add or remove hold types to the *Hold\_Types* attribute, respectively. The **-h**

- 29688 option has been modified to allow for implementation-defined hold types.
- 29689 The `-j` option allows users to alter the decision to join (merge) the standard error stream of the  
29690 batch job with the standard output stream of the batch job.
- 29691 The `-l` option allows users to change the resource limits imposed on a batch job.
- 29692 The `-m` option allows users to modify the list of points in the life of a batch job at which the  
29693 designated users will receive mail notification.
- 29694 The `-M` option allows users to alter the list of users who will receive notification about events in  
29695 the life of a batch job.
- 29696 The `-N` option allows users to change the name of a batch job.
- 29697 The `-o` option allows users to alter the name and path to which the standard output stream of  
29698 the batch job will be written.
- 29699 The `-P` option allows users to modify the priority of a batch job. Support for priority is optional  
29700 for batch servers.
- 29701 The `-r` option allows users to alter the rerunability status of a batch job.
- 29702 The `-S` option allows users to change the name and location of the shell image that will be  
29703 invoked to interpret the script of the batch job. This option has been modified to allow a list of  
29704 shell name and locations associated with different host.
- 29705 The `-u` option allows users to change the user identifier under which the batch job will execute.
- 29706 The `job_identifier` operand syntax is provided so that the user can differentiate between the  
29707 originating and destination (or executing) batch server. These may or may not be the same. The  
29708 `.server_name` portion identifies the originating batch server, while the `@server` portion identifies  
29709 the destination batch server.
- 29710 Historically, the `qalter` utility has been a component of the Network Queuing System (NQS), the  
29711 existing practice from which this utility has been derived.
- 29712 **FUTURE DIRECTIONS**
- 29713 None.
- 29714 **SEE ALSO**
- 29715 `qdel`, `qhold`, `qmove`, `qrls`, `qsub`, `touch`, Chapter 3 (on page 2313)
- 29716 **CHANGE HISTORY**
- 29717 Derived from IEEE Std. 1003.2d-1994.

29718 **NAME**

29719 qdel — delete batch jobs

29720 **SYNOPSIS**29721 BE qdel *job\_identifier* ...

29722

29723 **DESCRIPTION**29724 A batch job is deleted by sending a request to the batch server that manages the batch job. A  
29725 batch job that has been deleted is no longer subject to management by batch services.29726 The *qdel* utility is a user-accessible client of batch services that requests the deletion of one or  
29727 more batch jobs.29728 The *qdel* utility shall request a batch server to delete those batch jobs for which a batch  
29729 *job\_identifier* is presented to the utility.29730 The *qdel* utility shall delete batch jobs in the order in which their batch *job\_identifiers* are  
29731 presented to the utility.29732 If the *qdel* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
29733 process the remaining batch *job\_identifiers*, if any.29734 The *qdel* utility shall delete each batch job by sending a *Delete Job Request* to the batch server that  
29735 manages the batch job.29736 The *qdel* utility shall not exit until the batch job corresponding to each successfully processed  
29737 batch *job\_identifier* has been deleted.29738 **OPTIONS**

29739 None.

29740 **OPERANDS**29741 The *qdel* utility shall accept one or more operands that conform to the syntax for a batch  
29742 *job\_identifier* (see Section 3.3.1 (on page 2336)).29743 **STDIN**

29744 Not used.

29745 **INPUT FILES**

29746 None.

29747 **ENVIRONMENT VARIABLES**29748 The following environment variables shall affect the execution of *qdel*:29749 *LANG* Provide a default value for the internationalization variables that are unset or null.  
29750 If *LANG* is unset or null, the corresponding value from the implementation-  
29751 defined default locale shall be used. If any of the internationalization variables  
29752 contains an invalid setting, the utility shall behave as if none of the variables had  
29753 been defined.29754 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
29755 internationalization variables.29756 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
29757 characters (for example, single-byte as opposed to multi-byte characters in  
29758 arguments).29759 *LC\_MESSAGES*29760 Determine the locale that should be used to affect the format and contents of  
29761 diagnostic messages written to standard error.



- 29762        *LC\_TIME*    Determine the format and contents of date and time strings written by *qdel*.
- 29763        *LOGNAME*    Determine the login name of the user.
- 29764        *TZ*            Determine the timezone in which the time and date are written. If the *TZ* variable  
29765                    is not set, an unspecified system default timezone is used.
- 29766 **ASYNCHRONOUS EVENTS**
- 29767        Default.
- 29768 **STDOUT**
- 29769        An implementation of the *qdel* utility may write informative messages to standard output.
- 29770 **STDERR**
- 29771        Used only for diagnostic messages.
- 29772 **OUTPUT FILES**
- 29773        None.
- 29774 **EXTENDED DESCRIPTION**
- 29775        None.
- 29776 **EXIT STATUS**
- 29777        The following exit values shall be returned:
- 29778        0    Successful completion.
- 29779        >0    An error occurred.
- 29780 **CONSEQUENCES OF ERRORS**
- 29781        In addition to the default behavior, the *qdel* utility shall not be required to write a diagnostic  
29782        message to standard error when the error reply received from a batch server indicates that the  
29783        batch *job\_identifier* does not exist on the server. Whether or not the *qdel* utility waits to output the  
29784        diagnostic message while attempting to locate the job on other servers is implementation-  
29785        defined.
- 29786 **APPLICATION USAGE**
- 29787        None.
- 29788 **EXAMPLES**
- 29789        None.
- 29790 **RATIONALE**
- 29791        The *qdel* utility allows users and administrators to delete jobs.
- 29792        The *qdel* utility provides functionality that is not otherwise available. For example, the *kill* utility  
29793        of the operating system does not suffice. First, to use the *kill* utility, the user might have to log in  
29794        on a remote node, because the *kill* utility does not operate across the network. Second, unlike  
29795        *qdel*, *kill* cannot remove jobs from queues. Lastly, the arguments of the *qdel* utility are job  
29796        identifiers rather than process identifiers, and so this utility can be passed the output of the  
29797        *qselect* utility, thus providing users with a means of deleting a list of jobs.
- 29798        Because a set of jobs can be selected using the *qselect* utility, the *qdel* utility has not been  
29799        complicated with options that provide for selection of jobs. Instead, the batch jobs to be deleted  
29800        are identified individually by their job identifiers.
- 29801        Historically, the *qdel* utility has been a component of NQS, the existing practice on which it is  
29802        based. However, the *qdel* utility defined in this volume of IEEE Std. 1003.1-200x does not provide  
29803        an option for specifying a signal number to send to the batch job prior to the killing of the  
29804        process; that capability has been subsumed by the *qsig* utility.

29805 A discussion was held about the delays of networking and the possibility that the batch server  
29806 may never respond, due to a down router, down batch server, or other network mishap. The  
29807 DESCRIPTION records this under the words “fails to process any job identifier”. In the broad  
29808 sense, the network problem is also an error, which causes the failure to process the batch job  
29809 identifier.

29810 **FUTURE DIRECTIONS**

29811 None.

29812 **SEE ALSO**

29813 *kill, qselect, qsig*, Chapter 3 (on page 2313)

29814 **CHANGE HISTORY**

29815 Derived from IEEE Std. 1003.2d-1994.

29816 **NAME**

29817 qhold — hold batch jobs

29818 **SYNOPSIS**29819 BE qhold [-h *hold\_list*] *job\_identifier* ...

29820

29821 **DESCRIPTION**

29822 A hold is placed on a batch job by a request to the batch server that manages the batch job. A  
 29823 batch job that has one or more holds is not eligible for execution. The *qhold* utility is a user-  
 29824 accessible client of batch services that requests one or more types of hold to be placed on one or  
 29825 more batch jobs.

29826 The *qhold* utility shall place holds on those batch jobs for which a batch *job\_identifier* is presented  
 29827 to the utility.

29828 The *qhold* utility shall place holds on batch jobs in the order in which their batch *job\_identifiers*  
 29829 are presented to the utility. If the *qhold* utility fails to process any batch *job\_identifier* successfully,  
 29830 the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

29831 The *qhold* utility shall place holds on each batch job by sending a *Hold Job Request* to the batch  
 29832 server that manages the batch job.

29833 The *qhold* utility shall not exit until holds have been placed on the batch job corresponding to  
 29834 each successfully processed batch *job\_identifier*.

29835 **OPTIONS**

29836 The *qhold* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 29837 12.2, Utility Syntax Guidelines.

29838 The following option shall be supported by the implementation:

29839 **-h *hold\_list*** Define the types of holds to be placed on the batch job.

29840 The *qhold* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 29841 string of alphanumeric characters in the portable character set (see the Base  
 29842 Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

29843 The *qhold* utility shall accept a value for the *hold\_list* option-argument that is a  
 29844 string of one or more of the characters 'u', 's', or 'o', or the single character  
 29845 'n'.

29846 For each unique character in the *hold\_list* option-argument, the *qhold* utility shall  
 29847 add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 29848 representing a different hold type:

29849 **u** USER

29850 **s** SYSTEM

29851 **o** OPERATOR

29852 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 29853 duplicates shall be ignored.

29854 An existing *Hold\_Types* attribute can be cleared by the following hold type:

29855 **n** NO\_HOLD

29856 The *qhold* utility shall consider it an error if any hold type other than **n** is combined  
 29857 with hold type **n**.

29858 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
29859 'o', or 'n' within the *hold\_list* option-argument. The *qhold* utility shall permit the  
29860 repetition of characters, but shall not assign additional meaning to the repeated  
29861 characters.

29862 An implementation may define other hold types. The conformance document for  
29863 an implementation shall describe any additional hold types, how they are  
29864 specified, their internal behavior, and how they affect the behavior of the utility.

29865 If the *-h* option is not presented to the *qhold* utility, the implementation shall set  
29866 the *Hold\_Types* attribute to *USER*.

#### 29867 OPERANDS

29868 The *qhold* utility shall accept one or more operands that conform to the syntax for a batch  
29869 *job\_identifier* (see Section 3.3.1 (on page 2336)).

#### 29870 STDIN

29871 Not used.

#### 29872 INPUT FILES

29873 None.

#### 29874 ENVIRONMENT VARIABLES

29875 The following environment variables shall affect the execution of *qhold*:

29876 *LANG* Provide a default value for the internationalization variables that are unset or null.  
29877 If *LANG* is unset or null, the corresponding value from the implementation-  
29878 defined default locale shall be used. If any of the internationalization variables  
29879 contains an invalid setting, the utility shall behave as if none of the variables had  
29880 been defined.

29881 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
29882 internationalization variables.

29883 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
29884 characters (for example, single-byte as opposed to multi-byte characters in  
29885 arguments).

#### 29886 *LC\_MESSAGES*

29887 Determine the locale that should be used to affect the format and contents of  
29888 diagnostic messages written to standard error.

29889 *LC\_TIME* Determine the format and contents of date and time strings written by *qhold*.

29890 *LOGNAME* Determine the login name of the user.

29891 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
29892 is not set, an unspecified system default timezone is used.

#### 29893 ASYNCHRONOUS EVENTS

29894 Default.

#### 29895 STDOUT

29896 None.

#### 29897 STDERR

29898 Used only for diagnostic messages.

29899 **OUTPUT FILES**

29900 None.

29901 **EXTENDED DESCRIPTION**

29902 None.

29903 **EXIT STATUS**

29904 The following exit values shall be returned:

29905 0 Successful completion.

29906 &gt;0 An error occurred.

29907 **CONSEQUENCES OF ERRORS**

29908 In addition to the default behavior, the *qhold* utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch *job\_identifier* does not exist on the server. Whether or not the *qhold* utility waits to output the diagnostic message while attempting to locate the job on other servers is implementation-defined.

29913 **APPLICATION USAGE**

29914 None.

29915 **EXAMPLES**

29916 None.

29917 **RATIONALE**

29918 The *qhold* utility allows users to place a hold on one or more jobs. A hold makes a batch job ineligible for execution.

29920 The *qhold* utility has options that allow the user to specify the type of hold. Should the user wish to place a hold on a set of jobs that meet a selection criteria, such a list of jobs can be acquired using the *qselect* utility.

29923 The *-h* option allows the user to specify the type of hold that is to be placed on the job. This option allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The USER and OPERATOR holds are distinct. The batch server that manages the batch job will verify that the user is authorized to set the specified hold for the batch job.

29927 Mail is not required on hold because the administrator has the tools and libraries to build this option if he or she wishes.

29929 Historically, the *qhold* utility has been a part of some existing batch systems, although it has not traditionally been a part of the NQS.

29931 **FUTURE DIRECTIONS**

29932 None.

29933 **SEE ALSO**29934 *qselect*, Chapter 3 (on page 2313)29935 **CHANGE HISTORY**

29936 Derived from IEEE Std. 1003.2d-1994.

29937 **NAME**

29938 qmove — move batch jobs

29939 **SYNOPSIS**29940 BE qmove *destination job\_identifier* ...

29941

29942 **DESCRIPTION**

29943 To move a batch job is to remove the batch job from the batch queue in which it resides and  
 29944 instantiate the batch job in another batch queue. A batch job is moved by a request to the batch  
 29945 server that manages the batch job. The *qmove* utility is a user-accessible batch client that requests  
 29946 the movement of one or more batch jobs.

29947 The *qmove* utility shall move those batch jobs, and only those batch jobs, for which a batch  
 29948 *job\_identifier* is presented to the utility.

29949 The *qmove* utility shall move batch jobs in the order in which the corresponding batch  
 29950 *job\_identifiers* are presented to the utility.

29951 If the *qmove* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 29952 process the remaining batch *job\_identifiers*, if any.

29953 The *qmove* utility shall move batch jobs by sending a *Move Job Request* to the batch server that  
 29954 manages each batch job. The *qmove* utility shall not exit before the batch jobs corresponding to all  
 29955 successfully processed batch *job\_identifiers* have been moved.

29956 **OPTIONS**

29957 None.

29958 **OPERANDS**

29959 The *qmove* utility shall accept one operand that conforms to the syntax for a *destination* (see  
 29960 Section 3.3.2 (on page 2337)).

29961 The *qmove* utility shall accept one or more operands that conform to the syntax for a batch  
 29962 *job\_identifier* (see Section 3.3.1 (on page 2336)).

29963 **STDIN**

29964 Not used.

29965 **INPUT FILES**

29966 None.

29967 **ENVIRONMENT VARIABLES**29968 The following environment variables shall affect the execution of *qmove*:

29969 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 29970 If *LANG* is unset or null, the corresponding value from the implementation-  
 29971 defined default locale shall be used. If any of the internationalization variables  
 29972 contains an invalid setting, the utility shall behave as if none of the variables had  
 29973 been defined.

29974 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 29975 internationalization variables.

29976 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 29977 characters (for example, single-byte as opposed to multi-byte characters in  
 29978 arguments).

29979 *LC\_MESSAGES*

29980 Determine the locale that should be used to affect the format and contents of

- 29981 diagnostic messages written to standard error.
- 29982 *LC\_TIME* Determine the format and contents of date and time strings written by *qmove*.
- 29983 *LOGNAME* Determine the login name of the user.
- 29984 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
29985 is not set, an unspecified system default timezone is used.
- 29986 **ASYNCHRONOUS EVENTS**
- 29987 Default.
- 29988 **STDOUT**
- 29989 None.
- 29990 **STDERR**
- 29991 Used only for diagnostic messages.
- 29992 **OUTPUT FILES**
- 29993 None.
- 29994 **EXTENDED DESCRIPTION**
- 29995 None.
- 29996 **EXIT STATUS**
- 29997 The following exit values shall be returned:
- 29998 0 Successful completion.
- 29999 >0 An error occurred.
- 30000 **CONSEQUENCES OF ERRORS**
- 30001 In addition to the default behavior, the *qmove* utility shall not be required to write a diagnostic  
30002 message to standard error when the error reply received from a batch server indicates that the  
30003 batch *job\_identifier* does not exist on the server. Whether or not the *qmove* utility waits to output  
30004 the diagnostic message while attempting to locate the job on other servers is implementation-  
30005 defined.
- 30006 **APPLICATION USAGE**
- 30007 None.
- 30008 **EXAMPLES**
- 30009 None.
- 30010 **RATIONALE**
- 30011 The *qmove* utility allows users to move jobs between queues.
- 30012 The alternative to using the *qmove* utility—deleting the batch job and requeuing it—entails  
30013 considerably more typing.
- 30014 Since the means of selecting jobs based on attributes has been encapsulated in the *qselect* utility,  
30015 the only option of the *qmove* utility concerns authorization. The *-u* option provides the user with  
30016 the convenience of changing the user identifier under which the batch job will execute.
- 30017 Minimalism and consistency has taken precedence over convenience; the *-u* option has been  
30018 deleted because the equivalent capability exists with the *-u* option of the *qalter* utility.
- 30019 **FUTURE DIRECTIONS**
- 30020 None.

30021 **SEE ALSO**

30022           *qalter, qselect*, Chapter 3 (on page 2313)

30023 **CHANGE HISTORY**

30024           Derived from IEEE Std. 1003.2d-1994.



30025 **NAME**

30026 qmsg — send message to batch jobs

30027 **SYNOPSIS**30028 BE qmsg [-E][-O] *message\_string* *job\_identifier* ...

30029

30030 **DESCRIPTION**

30031 To send a message to a batch job is to request that a server write a message string into one or  
 30032 more output files of the batch job. A message is sent to a batch job by a request to the batch  
 30033 server that manages the batch job. The *qmsg* utility is a user-accessible batch client that requests  
 30034 the sending of messages to one or more batch jobs.

30035 The *qmsg* utility shall write messages into the files of batch jobs by sending a *Job Message Request*  
 30036 to the batch server that manages the batch job. The *qmsg* utility shall not directly write the  
 30037 message into the files of the batch job.

30038 The *qmsg* utility shall send a *Job Message Request* for those batch jobs, and only those batch jobs,  
 30039 for which a batch *job\_identifier* is presented to the utility.

30040 The *qmsg* utility shall send *Job Message Requests* for batch jobs in the order in which their batch  
 30041 *job\_identifiers* are presented to the utility.

30042 If the *qmsg* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 30043 process the remaining batch *job\_identifiers*, if any.

30044 The *qmsg* utility shall not exit before a *Job Message Request* has been sent to the server that  
 30045 manages the batch job that corresponds to each successfully processed batch *job\_identifier*.

30046 **OPTIONS**

30047 The *qmsg* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30048 12.2, Utility Syntax Guidelines.

30049 The following options shall be supported by the implementation:

30050 **-E** Specify that the message is written to the standard error of each batch job.

30051 The *qmsg* utility shall write the message into the standard error of the batch job.

30052 **-O** Specify that the message is written to the standard output of each batch job.

30053 The *qmsg* utility shall write the message into the standard output of the batch job.

30054 If neither the **-O** nor the **-E** option is presented to the *qmsg* utility, the utility shall write the  
 30055 message into an implementation-defined file. The conformance document for the  
 30056 implementation shall describe the name and location of the implementation-defined file. If both  
 30057 the **-O** and the **-E** options are presented to the *qmsg* utility, then the utility shall write the  
 30058 messages to both standard output and standard error.

30059 **OPERANDS**

30060 The *qmsg* utility shall accept a minimum of two operands, *message\_string* and one or more batch  
 30061 *job\_identifiers*.

30062 The *message\_string* operand shall be the string to be written to one or more output files of the  
 30063 batch job followed by a <newline>. If the string contains <blank>s, then the application shall  
 30064 ensure that the string is quoted. The *message\_string* shall be encoded in the portable character set  
 30065 (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

30066 All remaining operands are batch *job\_identifiers* that conform to the syntax for a batch  
 30067 *job\_identifier* (see Section 3.3.1 (on page 2336)).

30068 **STDIN**

30069 Not used.

30070 **INPUT FILES**

30071 None.

30072 **ENVIRONMENT VARIABLES**30073 The following environment variables shall affect the execution of *qmsg*:

30074 *LANG* Provide a default value for the internationalization variables that are unset or null.  
30075 If *LANG* is unset or null, the corresponding value from the implementation-  
30076 defined default locale shall be used. If any of the internationalization variables  
30077 contains an invalid setting, the utility shall behave as if none of the variables had  
30078 been defined.

30079 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
30080 internationalization variables.

30081 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
30082 characters (for example, single-byte as opposed to multi-byte characters in  
30083 arguments).

30084 *LC\_MESSAGES*  
30085 Determine the locale that should be used to affect the format and contents of  
30086 diagnostic messages written to standard error.

30087 *LC\_TIME* Determine the format and contents of date and time strings written by *qmsg*.

30088 *LOGNAME* Determine the login name of the user.

30089 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
30090 is not set, an unspecified system default timezone is used.

30091 **ASYNCHRONOUS EVENTS**

30092 Default.

30093 **STDOUT**

30094 None.

30095 **STDERR**

30096 Used only for diagnostic messages.

30097 **OUTPUT FILES**

30098 None.

30099 **EXTENDED DESCRIPTION**

30100 None.

30101 **EXIT STATUS**

30102 The following exit values shall be returned:

30103 0 Successful completion.

30104 &gt;0 An error occurred.

30105 **CONSEQUENCES OF ERRORS**

30106 In addition to the default behavior, the *qmsg* utility shall not be required to write a diagnostic  
30107 message to standard error when the error reply received from a batch server indicates that the  
30108 batch *job\_identifier* does not exist on the server. Whether or not the *qmsg* utility waits to output  
30109 the diagnostic message while attempting to locate the job on other servers is implementation-  
30110 defined.

**30111 APPLICATION USAGE**

30112 None.

**30113 EXAMPLES**

30114 None.

**30115 RATIONALE**

30116 The *qmsg* utility allows users to write messages into the output files of running jobs. Users,  
30117 including operators and administrators, have a number of occasions when they want to place  
30118 messages in the output files of a batch job. For example, if a disk that is being used by a batch job  
30119 is showing errors, the operator might note this in the standard error stream of the batch job.

30120 The options of the *qmsg* utility provide users with the means of placing the message in the  
30121 output stream of their choice. The default output stream for the message—if the user does not  
30122 designate an output stream—is implementation-defined, since many implementations will  
30123 provide, as an extension to this volume of IEEE Std. 1003.1-200x, a log file that shows the history  
30124 of utility execution.

30125 If users wish to send a message to a set of jobs that meet a selection criteria, the *qselect* utility can  
30126 be used to acquire the appropriate list of job identifiers.

30127 The **-E** option allows users to place the message in the standard error stream of the batch job.

30128 The **-O** option allows users to place the message in the standard output stream of the batch job.

30129 Historically, the *qmsg* utility is an existing practice in the offerings of one or more implementors  
30130 of an NQS-derived batch system. The utility has been found to be useful enough that it deserves  
30131 to be included in this volume of IEEE Std. 1003.1-200x.

**30132 FUTURE DIRECTIONS**

30133 None.

**30134 SEE ALSO**

30135 *qselect*, Chapter 3 (on page 2313)

**30136 CHANGE HISTORY**

30137 Derived from IEEE Std. 1003.2d-1994.

30138 **NAME**

30139           qrerun — rerun batch jobs

30140 **SYNOPSIS**30141 BE        `qrerun job_identifier ...`

30142

30143 **DESCRIPTION**

30144       To rerun a batch job is to terminate the session leader of the batch job, delete any associated  
 30145       checkpoint files, and return the batch job to the batch queued state. A batch job is rerun by a  
 30146       request to the batch server that manages the batch job. The *qrerun* utility is a user-accessible  
 30147       batch client that requests the rerunning of one or more batch jobs.

30148       The *qrerun* utility shall rerun those batch jobs for which a batch *job\_identifier* is presented to the  
 30149       utility.

30150       The *qrerun* utility shall rerun batch jobs in the order in which their batch *job\_identifiers* are  
 30151       presented to the utility.

30152       If the *qrerun* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed  
 30153       to process the remaining batch *job\_identifiers*, if any.

30154       The *qrerun* utility shall rerun batch jobs by sending a *Rerun Job Request* to the batch server that  
 30155       manages each batch job.

30156       For each successfully processed batch *job\_identifier*, the *qrerun* utility shall have rerun the  
 30157       corresponding batch batch job at the time the utility exits.

30158 **OPTIONS**

30159           None.

30160 **OPERANDS**

30161       The *qrerun* utility shall accept one or more operands that conform to the syntax for a batch  
 30162       *job\_identifier* (see Section 3.3.1 (on page 2336)).

30163 **STDIN**

30164           Not used.

30165 **INPUT FILES**

30166           None.

30167 **ENVIRONMENT VARIABLES**30168       The following environment variables shall affect the execution of *qrerun*:

30169       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 30170       If *LANG* is unset or null, the corresponding value from the implementation-  
 30171       defined default locale shall be used. If any of the internationalization variables  
 30172       contains an invalid setting, the utility shall behave as if none of the variables had  
 30173       been defined.

30174       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 30175       internationalization variables.

30176       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 30177       characters (for example, single-byte as opposed to multi-byte characters in  
 30178       arguments).

30179       *LC\_MESSAGES*

30180       Determine the locale that should be used to affect the format and contents of  
 30181       diagnostic messages written to standard error.

- 30182        *LC\_TIME*    Determine the format and contents of date and time strings written by *qrerun*.
- 30183        *LOGNAME*    Determine the login name of the user.
- 30184        *TZ*            Determine the timezone in which the time and date are written. If the *TZ* variable  
30185                    is not set, an unspecified system default timezone is used.
- 30186 **ASYNCHRONOUS EVENTS**
- 30187        Default.
- 30188 **STDOUT**
- 30189        None.
- 30190 **STDERR**
- 30191        Used only for diagnostic messages.
- 30192 **OUTPUT FILES**
- 30193        None.
- 30194 **EXTENDED DESCRIPTION**
- 30195        None.
- 30196 **EXIT STATUS**
- 30197        The following exit values shall be returned:
- 30198        0    Successful completion.
- 30199        >0    An error occurred.
- 30200 **CONSEQUENCES OF ERRORS**
- 30201        In addition to the default behavior, the *qrerun* utility shall not be required to write a diagnostic  
30202        message to standard error when the error reply received from a batch server indicates that the  
30203        batch *job\_identifier* does not exist on the server. Whether or not the *qrerun* utility waits to output  
30204        the diagnostic message while attempting to locate the job on other servers is implementation-  
30205        defined.
- 30206 **APPLICATION USAGE**
- 30207        None.
- 30208 **EXAMPLES**
- 30209        None.
- 30210 **RATIONALE**
- 30211        The *qrerun* utility allows users to cause jobs in the running state to exit and rerun.
- 30212        The *qrerun* utility is a new utility, *vis-a-vis* existing practice, that has been defined in this volume  
30213        of IEEE Std. 1003.1-200x to correct user-perceived deficiencies in the existing practice.
- 30214 **FUTURE DIRECTIONS**
- 30215        None.
- 30216 **SEE ALSO**
- 30217        Chapter 3 (on page 2313)
- 30218 **CHANGE HISTORY**
- 30219        Derived from IEEE Std. 1003.2d-1994.

## 30220 NAME

30221 qrls — release batch jobs

## 30222 SYNOPSIS

30223 BE qrls [-h *hold\_list*] *job\_identifier* ...

30224

## 30225 DESCRIPTION

30226 A batch job might have one or more holds, which prevent the batch job from executing. A batch  
 30227 job from which all the holds have been removed becomes eligible for execution and is said to  
 30228 have been released. A batch job hold is removed by sending a request to the batch server that  
 30229 manages the batch job. The *qrls* utility is a user-accessible client of batch services that requests  
 30230 holds be removed from one or more batch jobs.

30231 The *qrls* utility shall remove one or more holds from those batch jobs for which a batch  
 30232 *job\_identifier* is presented to the utility.

30233 The *qrls* utility shall remove holds from batch jobs in the order in which their batch *job\_identifiers*  
 30234 are presented to the utility.

30235 If the *qrls* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 30236 process the remaining batch *job\_identifiers*, if any.

30237 The *qrls* utility shall remove holds on each batch job by sending a *Release Job Request* to the batch  
 30238 server that manages the batch job.

30239 The *qrls* utility shall not exit until the holds have been removed from the batch job  
 30240 corresponding to each successfully processed batch *job\_identifier*.

## 30241 OPTIONS

30242 The *qrls* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30243 12.2, Utility Syntax Guidelines.

30244 The following option shall be supported by the implementation:

30245 **-h *hold\_list*** Define the types of holds to be removed from the batch job.

30246 The *qrls* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 30247 string of alphanumeric characters in the portable character set (see the Base  
 30248 Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

30249 The *qrls* utility shall accept a value for the *hold\_list* option-argument that is a string  
 30250 of one or more of the characters 'u', 's', or 'o', or the single character 'n'.

30251 For each unique character in the *hold\_list* option-argument, the *qrls* utility shall add  
 30252 a value to the *Hold\_Types* attribute of the batch job as follows, each representing a  
 30253 different hold type:

30254 **u** USER

30255 **s** SYSTEM

30256 **o** OPERATOR

30257 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 30258 duplicates shall be ignored.

30259 An existing *Hold\_Types* attribute can be cleared by the following hold type:

30260 **n** NO\_HOLD

- 30261 The *qrls* utility shall consider it an error if any hold type other than **n** is combined  
30262 with hold type **n**.
- 30263 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
30264 'o', or 'n' within the *hold\_list* option-argument. The *qrls* utility shall permit the  
30265 repetition of characters, but shall not assign additional meaning to the repeated  
30266 characters.
- 30267 An implementation may define other hold types. The conformance document for  
30268 an implementation shall describe any additional hold types, how they are  
30269 specified, their internal behavior, and how they affect the behavior of the utility.
- 30270 If the **-h** option is not presented to the *qrls* utility, the implementation shall remove  
30271 the **USER** hold in the *Hold\_Types* attribute.
- 30272 **OPERANDS**
- 30273 The *qrls* utility shall accept one or more operands that conform to the syntax for a batch  
30274 *job\_identifier* (see Section 3.3.1 (on page 2336)).
- 30275 **STDIN**
- 30276 Not used.
- 30277 **INPUT FILES**
- 30278 None.
- 30279 **ENVIRONMENT VARIABLES**
- 30280 The following environment variables shall affect the execution of *qrls*:
- 30281 *LANG* Provide a default value for the internationalization variables that are unset or null.  
30282 If *LANG* is unset or null, the corresponding value from the implementation-  
30283 defined default locale shall be used. If any of the internationalization variables  
30284 contains an invalid setting, the utility shall behave as if none of the variables had  
30285 been defined.
- 30286 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
30287 internationalization variables.
- 30288 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
30289 characters (for example, single-byte as opposed to multi-byte characters in  
30290 arguments).
- 30291 *LC\_MESSAGES*
- 30292 Determine the locale that should be used to affect the format and contents of  
30293 diagnostic messages written to standard error.
- 30294 *LC\_TIME* Determine the format and contents of date and time strings written by *qrls*.
- 30295 *LOGNAME* Determine the login name of the user.
- 30296 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
30297 is not set, an unspecified system default timezone is used.
- 30298 **ASYNCHRONOUS EVENTS**
- 30299 Default.
- 30300 **STDOUT**
- 30301 None.

30302 **STDERR**

30303           Used only for diagnostic messages.

30304 **OUTPUT FILES**

30305           None.

30306 **EXTENDED DESCRIPTION**

30307           None.

30308 **EXIT STATUS**

30309           The following exit values shall be returned:

30310           0   Successful completion.

30311           >0  An error occurred.

30312 **CONSEQUENCES OF ERRORS**

30313           In addition to the default behavior, the *qrls* utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch *job\_identifier* does not exist on the server. Whether or not the *qrls* utility waits to output the diagnostic message while attempting to locate the job on other servers is implementation-defined.

30318 **APPLICATION USAGE**

30319           None.

30320 **EXAMPLES**

30321           None.

30322 **RATIONALE**

30323           The *qrls* utility allows users, operators, and administrators to remove holds from jobs.

30324           The *qrls* utility does not support any job selection options or wildcard arguments. Users may acquire a list of jobs selected by attributes using the *qselect* utility. For example, a user could select all of their held jobs.

30327           The *-h* option allows the user to specify the type of hold that is to be removed. This option allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The batch server that manages the batch job will verify whether the user is authorized to remove the specified hold for the batch job. If more than one type of hold has been placed on the batch job, a user may wish to remove only some of them.

30332           Mail is not required on release because the administrator has the tools and libraries to build this option if required.

30334           The *qrls* utility is a new utility *vis-a-vis* existing practice; it has been defined in this volume of IEEE Std. 1003.1-200x as the natural complement to the *qhold* utility.

30336 **FUTURE DIRECTIONS**

30337           None.

30338 **SEE ALSO**

30339           *qhold*, *qselect*, Chapter 3 (on page 2313)

30340 **CHANGE HISTORY**

30341           Derived from IEEE Std. 1003.2d-1994.



## 30342 NAME

30343 qselect — select batch jobs

## 30344 SYNOPSIS

```
30345 BE qselect [-a [op]date_time][-A account_string][-c [op]interval]
30346 [-h hold_list][-l resource_list][-N name][-p [op]priority]
30347 [-q destination][-r y|n][-s states][-u user_list]
30348
```

## 30349 DESCRIPTION

30350 To select a set of batch jobs is to return the batch *job\_identifiers* for each batch job that meets a list  
 30351 of selection criteria. A set of batch jobs is selected by a request to a batch server. The *qselect*  
 30352 utility is a user-accessible batch client that requests the selection of batch jobs.

30353 Upon successful completion, the *qselect* utility shall have returned a list of zero or more batch  
 30354 *job\_identifiers* that meet the criteria specified by the options and option-arguments presented to  
 30355 the utility.

30356 The *qselect* utility shall select batch jobs by sending a *Select Jobs Request* to a batch server. The  
 30357 *qselect* utility shall not exit until the server replies to each request generated.

30358 For each option presented to the *qselect* utility, the utility shall restrict the set of selected batch  
 30359 jobs as described in the OPTIONS section.

30360 The *qselect* utility shall not restrict selection of batch jobs except by authorization and as required  
 30361 by the options presented to the utility.

30362 When an option is specified with a mandatory or optional *op* component to the option-  
 30363 argument, then *op* shall specify a relation between the value of a certain batch job attribute and  
 30364 the *value* component of the option-argument. If an *op* is allowable on an option, then the  
 30365 description of the option letter indicates the *op* as either mandatory or optional. Acceptable  
 30366 strings for the *op* component, and the relation the string indicates, are shown in the following  
 30367 list:

30368 .eq. The value represented by the attribute of the batch job is equal to the value represented  
 30369 by the option-argument.

30370 .ge. The value represented by the attribute of the batch job is greater than or equal to the  
 30371 value represented by the option-argument.

30372 .gt. The value represented by the attribute of the batch job is greater than the value  
 30373 represented by the option-argument.

30374 .lt. The value represented by the attribute of the batch job is less than the value  
 30375 represented by the option-argument.

30376 .le. The value represented by the attribute of the batch job is less than or equal to the value  
 30377 represented by the option-argument.

30378 .ne. The value represented by the attribute of the batch job is not equal to the value  
 30379 represented by the option-argument.

## 30380 OPTIONS

30381 The *qselect* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30382 12.2, Utility Syntax Guidelines.

30383 The following options shall be supported by the implementation:

30384 **-a [op]date\_time**  
 30385 Restrict selection to a specific time, or a range of times.

- 30386 The *qselect* utility shall select only batch jobs for which the value of the  
30387 *Execution\_Time* attribute is related to the Epoch equivalent of the local time  
30388 expressed by the value of the *date\_time* component of the option-argument in the  
30389 manner indicated by the value of the *op* component of the option-argument.
- 30390 The *qselect* utility shall accept a *date\_time* component of the option-argument that  
30391 conforms to the syntax of the *date\_time* operand of the *touch* utility.
- 30392 If the *op* component of the option-argument is not presented to the *qselect* utility,  
30393 the utility shall select batch jobs for which the *Execution\_Time* attribute is equal to  
30394 the *date\_time* component of the option-argument.
- 30395 When comparing times, the *qselect* utility shall use the following definitions for the  
30396 *op* component of the option-argument:
- 30397 .eq. The time represented by value of the *Execution\_Time* attribute of the batch  
30398 job is equal the time represented by the *date\_time* component of the  
30399 option-argument.
  - 30400 .ge. The time represented by value of the *Execution\_Time* attribute of the batch  
30401 job is after or equal to the time represented by the *date\_time* component of  
30402 the option-argument.
  - 30403 .gt. The time represented by value of the *Execution\_Time* attribute of the batch  
30404 job is after the time represented by the *date\_time* component of the  
30405 option-argument.
  - 30406 .lt. The time represented by value of the *Execution\_Time* attribute of the batch  
30407 job is before the time represented by the *date\_time* component of the  
30408 option-argument.
  - 30409 .le. The time represented by value of the *Execution\_Time* attribute of the batch  
30410 job is before or equal to the time represented by the *date\_time* component  
30411 of the option-argument.
  - 30412 .ne. The time represented by value of the *Execution\_Time* attribute of the batch  
30413 job is not equal to the time represented by the *date\_time* component of the  
30414 option-argument.
- 30415 The *qselect* utility shall accept the defined character strings for the *op* component of  
30416 the option-argument.
- 30417 **-A** *account\_string*  
30418 Restrict selection to the batch jobs charging a specified account.
- 30419 The *qselect* utility shall select only batch jobs for which the value of the  
30420 *Account\_Name* attribute of the batch job matches the value of the *account\_string*  
30421 option-argument.
- 30422 The syntax of the *account\_string* option-argument is unspecified.
- 30423 **-c** [*op*]*interval*  
30424 Restrict selection to batch jobs within a range of checkpoint intervals.
- 30425 The *qselect* utility shall select only batch jobs for which the value of the *Checkpoint*  
30426 attribute relates to the value of the *interval* component of the option-argument in  
30427 the manner indicated by the value of the *op* component of the option-argument.
- 30428 If the *op* component of the option-argument is omitted, the *qselect* utility shall  
30429 select batch jobs for which the value of the *Checkpoint* attribute is equal to the value

30430 of the *interval* component of the option-argument.

30431 When comparing checkpoint intervals, the *qselect* utility shall use the following  
30432 definitions for the *op* component of the option-argument:

30433 .eq. The value of the *Checkpoint* attribute of the batch job equals the value of  
30434 the *interval* component of the option-argument.

30435 .ge. The value of the *Checkpoint* attribute of the batch job is greater than or  
30436 equal to the value of the *interval* component option-argument.

30437 .gt. The value of the *Checkpoint* attribute of the batch job is greater than the  
30438 value of the *interval* component option-argument.

30439 .lt. The value of the *Checkpoint* attribute of the batch job is less than the value  
30440 of the *interval* component option-argument.

30441 .le. The value of the *Checkpoint* attribute of the batch job is less than or equal  
30442 to the value of the *interval* component option-argument.

30443 .ne. The value of the *Checkpoint* attribute of the batch job does not equal the  
30444 value of the *interval* component option-argument.

30445 The *qselect* utility shall accept the defined character strings for the *op* component of  
30446 the option-argument.

30447 The ordering relationship for the values of the interval option-argument is defined  
30448 to be:

30449 'n' .gt. 's' .gt. 'c=minutes' .ge. 'c'

30450 When comparing *Checkpoint* attributes with an interval having the value of the  
30451 single character 'u', only equality or inequality are valid comparisons.

30452 **-h hold\_list** Restrict selection to batch jobs that have a specific type of hold.

30453 The *qselect* utility shall select only batch jobs for which the value of the *Hold\_Types*  
30454 attribute matches the value of the *hold\_list* option-argument.

30455 The *qselect* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
30456 string of alphanumeric characters in the portable character set (see the Base  
30457 Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

30458 The *qselect* utility shall accept a value for the *hold\_list* option-argument that is a  
30459 string of one or more of the characters 'u', 's', or 'o', or the single character  
30460 'n'.

30461 Each unique character in the *hold\_list* option-argument of the *qselect* utility is  
30462 defined as follows, each representing a different hold type:

30463 **u** USER

30464 **s** SYSTEM

30465 **o** OPERATOR

30466 If any of these characters are duplicated in the *hold\_list* option-argument, the  
30467 duplicates shall be ignored.

30468 The *qselect* utility shall consider it an error if any hold type other than **n** is  
30469 combined with hold type **n**.

30470 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
 30471 'o', or 'n' within the *hold\_list* option-argument. The *qselect* utility shall permit  
 30472 the repetition of characters, but shall not assign additional meaning to the repeated  
 30473 characters.

30474 An implementation may define other hold types. The conformance document for  
 30475 an implementation shall describe any additional hold types, how they are  
 30476 specified, their internal behavior, and how they affect the behavior of the utility.

30477 **-I *resource\_list***  
 30478 Restrict selection to batch jobs with specified resource limits and attributes.

30479 The *qselect* utility shall accept a *resource\_list* option-argument with the following  
 30480 syntax:

30481 *resource\_name op value [ , , resource\_name op value , , ... ]*

30482 When comparing resource values, the *qselect* utility shall use the following  
 30483 definitions for the *op* component of the option-argument:

30484 *.eq.* The value of the resource of the same name in the *Resource\_List* attribute  
 30485 of the batch job equals the value of the value component of the option-  
 30486 argument.

30487 *.ge.* The value of the resource of the same name in the *Resource\_List* attribute  
 30488 of the batch job is greater than or equal to the value of the *value*  
 30489 component of the option-argument.

30490 *.gt.* The value of the resource of the same name in the *Resource\_List* attribute  
 30491 of the batch job is greater than the value of the value component of the  
 30492 option-argument.

30493 *.lt.* The value of the resource of the same name in the *Resource\_List* attribute  
 30494 of the batch job is less than the value of the value component of the  
 30495 option-argument.

30496 *.ne.* The value of the resource of the same name in the *Resource\_List* attribute  
 30497 of the batch job does not equal the value of the value component of the  
 30498 option-argument.

30499 *.le.* The value of the resource of the same name in the *Resource\_List* attribute  
 30500 of the batch job is less than or equal to the value of the *value*  
 30501 component of the option-argument.

30502 When comparing the limit of a *Resource\_List* attribute with the *value* component of  
 30503 the option-argument, if the limit, the value, or both are non-numeric, only equality  
 30504 or inequality are valid comparisons.

30505 The *qselect* utility shall select only batch jobs for which the values of the  
 30506 *resource\_names* listed in the *resource\_list* option-argument match the corresponding  
 30507 limits of the *Resource\_List* attribute of the batch job.

30508 Limits of *resource\_names* present in the *Resource\_List* attribute of the batch job that  
 30509 have no corresponding values in the *resource\_list* option-argument shall not be  
 30510 considered when selecting batch jobs.

30511 **-N *name*** Restrict selection to batch jobs with a specified name.

30512 The *qselect* utility shall select only batch jobs for which the value of the *Job\_Name*  
 30513 attribute matches the value of the *name* option-argument. The string specified in

- 30514 the *name* option-argument shall be passed, uninterpreted, to the server. This allows  
30515 an implementation to match “wildcard” patterns against batch job names.
- 30516 An implementation shall describe in the conformance document the format it  
30517 supports for matching against the *Job\_Name* attribute.
- 30518 **-p [op]priority**
- 30519 Restrict selection to batch jobs of the specified priority or range of priorities.
- 30520 The *qselect* utility shall select only batch jobs for which the value of the *Priority*  
30521 attribute of the batch job relates to the value of the *priority* component of the  
30522 option-argument in the manner indicated by the value of the *op* component of the  
30523 option-argument.
- 30524 If the *op* component of the option-argument is omitted, the *qselect* utility shall  
30525 select batch jobs for which the value of the *Priority* attribute of the batch job is  
30526 equal to the value of the *priority* component of the option-argument.
- 30527 When comparing priority values, the *qselect* utility shall use the following  
30528 definitions for the *op* component of the option-argument:
- 30529 .eq. The value of the *Priority* attribute of the batch job equals the value of the  
30530 *priority* component of the option-argument.
- 30531 .ge. The value of the *Priority* attribute of the batch job is greater than or equal  
30532 to the value of the *priority* component option-argument.
- 30533 .gt. The value of the *Priority* attribute of the batch job is greater than the value  
30534 of the *priority* component option-argument.
- 30535 .lt. The value of the *Priority* attribute of the batch job is less than the value of  
30536 the *priority* component option-argument.
- 30537 .lte. The value of the *Priority* attribute of the batch job is less than or equal to  
30538 the value of the *priority* component option-argument.
- 30539 .ne. The value of the *Priority* attribute of the batch job does not equal the value  
30540 of the *priority* component option-argument.
- 30541 **-q destination**
- 30542 Restrict selection to the specified batch queue or server, or both.
- 30543 The *qselect* utility shall select only batch jobs that are located at the destination  
30544 indicated by the value of the *destination* option-argument.
- 30545 The destination defines a batch queue, a server, or a batch queue at a server.
- 30546 The *qselect* utility shall accept an option-argument for the **-q** option that conforms  
30547 to the syntax for a destination. If the **-q** option is not presented to the *qselect* utility,  
30548 the utility shall select batch jobs from all batch queues at the default batch server.
- 30549 If the option-argument describes only a batch queue, the *qselect* utility shall select  
30550 only batch jobs from the batch queue of the specified name at the default batch  
30551 server. The means by which *qselect* determines the default server is  
30552 implementation-defined.
- 30553 If the option-argument describes only a batch server, the *qselect* utility shall select  
30554 batch jobs from all the batch queues at that batch server.
- 30555 If the option-argument describes both a batch queue and a batch server, the *qselect*  
30556 utility shall select only batch jobs from the specified batch queue at the specified

|       |                     |                                                                                                        |
|-------|---------------------|--------------------------------------------------------------------------------------------------------|
| 30557 |                     | server.                                                                                                |
| 30558 | <b>-r y n</b>       | Restrict selection to batch jobs with the specified rerunability status.                               |
| 30559 |                     | The <i>qselect</i> utility shall select only batch jobs for which the value of the <i>Rerunable</i>    |
| 30560 |                     | attribute of the batch job matches the value of the option-argument.                                   |
| 30561 |                     | The <i>qselect</i> utility shall accept a value for the option-argument that consists of               |
| 30562 |                     | either the single character 'y' or the single character 'n'. The character 'y'                         |
| 30563 |                     | represents the value TRUE, and the character 'n' represents the value FALSE.                           |
| 30564 | <b>-s states</b>    | Restrict selection to batch jobs in the specified states.                                              |
| 30565 |                     | The <i>qselect</i> utility shall accept an option-argument that consists of any combination            |
| 30566 |                     | of the characters 'e', 'q', 'r', 'w', 'h', and 't'.                                                    |
| 30567 |                     | Conforming applications shall not repeat any character in the option-argument.                         |
| 30568 |                     | The <i>qselect</i> utility shall permit the repetition of characters in the option-argument,           |
| 30569 |                     | but shall not assign additional meaning to repeated characters.                                        |
| 30570 |                     | The <i>qselect</i> utility shall interpret the characters in the <i>states</i> option-argument as      |
| 30571 |                     | follows:                                                                                               |
| 30572 | e                   | Represents the EXITING state.                                                                          |
| 30573 | q                   | Represents the QUEUED state.                                                                           |
| 30574 | r                   | Represents the RUNNING state.                                                                          |
| 30575 | t                   | Represents the TRANSITING state.                                                                       |
| 30576 | h                   | Represents the HELD state.                                                                             |
| 30577 | w                   | Represents the WAITING state.                                                                          |
| 30578 |                     | For each character in the <i>states</i> option-argument, the <i>qselect</i> utility shall select batch |
| 30579 |                     | jobs in the corresponding state.                                                                       |
| 30580 | <b>-u user_list</b> | Restrict selection to batch jobs owned by the specified user names.                                    |
| 30581 |                     | The <i>qselect</i> utility shall select only the batch jobs of those users specified in the            |
| 30582 |                     | <i>user_list</i> option-argument.                                                                      |
| 30583 |                     | The <i>qselect</i> utility shall accept a <i>user_list</i> option-argument that conforms to the        |
| 30584 |                     | following syntax:                                                                                      |
| 30585 |                     | <i>username</i> [@ <i>host</i> ][, , <i>username</i> [@ <i>host</i> ], , . . . ]                       |
| 30586 |                     | The <i>qselect</i> utility shall accept only one user name that is missing a corresponding             |
| 30587 |                     | host name. The <i>qselect</i> utility shall accept only one user name per named host.                  |
| 30588 | <b>OPERANDS</b>     |                                                                                                        |
| 30589 |                     | None.                                                                                                  |
| 30590 | <b>STDIN</b>        |                                                                                                        |
| 30591 |                     | Not used.                                                                                              |
| 30592 | <b>INPUT FILES</b>  |                                                                                                        |
| 30593 |                     | None.                                                                                                  |

30594 **ENVIRONMENT VARIABLES**

30595 The following environment variables shall affect the execution of *qselect*:

30596 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 30597 If *LANG* is unset or null, the corresponding value from the implementation-  
 30598 defined default locale shall be used. If any of the internationalization variables  
 30599 contains an invalid setting, the utility shall behave as if none of the variables had  
 30600 been defined.

30601 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 30602 internationalization variables.

30603 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 30604 characters (for example, single-byte as opposed to multi-byte characters in  
 30605 arguments).

30606 *LC\_MESSAGES*  
 30607 Determine the locale that should be used to affect the format and contents of  
 30608 diagnostic messages written to standard error.

30609 *LC\_TIME* Determine the format and contents of date and time strings written by *qselect*.

30610 *LOGNAME* Determine the login name of the user.

30611 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
 30612 is not set, an unspecified system default timezone is used.

30613 **ASYNCHRONOUS EVENTS**

30614 Default.

30615 **STDOUT**

30616 The *qselect* utility shall write zero or more batch *job\_identifiers* to standard output.

30617 The *qselect* utility shall separate the batch *job\_identifiers* written to standard output by white  
 30618 space.

30619 The *qselect* utility shall write batch *job\_identifiers* in the following format:

30620 *sequence\_number.server\_name@server*

30621 **STDERR**

30622 Used only for diagnostic messages.

30623 **OUTPUT FILES**

30624 None.

30625 **EXTENDED DESCRIPTION**

30626 None.

30627 **EXIT STATUS**

30628 The following exit values shall be returned:

30629 0 Successful completion.

30630 >0 An error occurred.

30631 **CONSEQUENCES OF ERRORS**

30632 Default.

30633 **APPLICATION USAGE**

30634 None.

30635 **EXAMPLES**

30636 The following example shows how a user might use the *qselect* utility in conjunction with the  
 30637 *qdel* utility to delete all of his or her jobs in the queued state without affecting any jobs that are  
 30638 already running:

30639 `qdel $(qselect -s q)`

30640 or:

30641 `qselect -s q || xargs qdel`30642 **RATIONALE**

30643 The *qselect* utility allows users to acquire a list of job identifiers that match user-specified  
 30644 selection criteria. The list of identifiers returned by the *qselect* utility conforms to the syntax of  
 30645 the batch job identifier list processed by a utility such as *qmove*, *qdel*, and *qrls*. The *qselect* utility is  
 30646 thus a powerful tool for causing another batch system utility to act upon a set of jobs that match  
 30647 a list of selection criteria.

30648 The options of the *qselect* utility let the user apply a number of useful filters for selecting jobs.  
 30649 Each option further restricts the selection of jobs. Many of the selection options allow the  
 30650 specification of a relational operator. The FORTRAN-like syntax of the operator—that is,  
 30651 ".lt.", was chosen rather than the C-like "<=" meta-characters.

30652 The *-a* option allows users to restrict the selected jobs to those that have been submitted (or  
 30653 altered) to wait until a particular time. The time period is determined by the argument of this  
 30654 option, which includes both a time and an operator—it is thus possible to select jobs waiting  
 30655 until a specific time, jobs waiting until after a certain time, or those waiting for a time before the  
 30656 specified time.

30657 The *-A* option allows users to restrict the selected jobs to those that have been submitted (or  
 30658 altered) to charge a particular account.

30659 The *-c* option allows users to restrict the selected jobs to those whose checkpointing interval  
 30660 falls within the specified range.

30661 The *-l* option allows users to select those jobs whose resource limits fall within the range  
 30662 indicated by the value of the option. For example, a user could select those jobs for which the  
 30663 CPU time limit is greater than two hours.

30664 The *-N* option allows users to select jobs by job name. For instance, all the parts of a task that  
 30665 have been divided in parallel jobs might be given the same name, and thus manipulated as a  
 30666 group by means of this option.

30667 The *-q* option allows users to select jobs in a specified queue.

30668 The *-r* option allows users to select only those jobs with a specified rerun criteria. For instance, a  
 30669 user might select only those jobs that can be rerun for use with the *qrerun* utility.

30670 The *-s* option allows users to select only those jobs that are in a certain state.

30671 The *-u* option allows users to select jobs that have been submitted to execute under a particular  
 30672 account.

30673 The selection criteria provided by the options of the *qselect* utility allow users to select jobs based  
 30674 on all the appropriate attributes that can be assigned to jobs by the *qsub* utility. When  
 30675 implementors extend the *qsub* utility, or another utilities, using the *-W* option, they may likewise  
 30676 elect to extend the *qselect* utility to allow additional selection criteria.



30677 Historically, the *qselect* utility has not been a part of existing practice; it is an improvement that  
30678 has been introduced in this volume of IEEE Std. 1003.1-200x.

30679 **FUTURE DIRECTIONS**

30680 None.

30681 **SEE ALSO**

30682 *qdel*, *qrerun*, *qrls*, *qselect*, *qsub*, *touch*, Chapter 3 (on page 2313)

30683 **CHANGE HISTORY**

30684 Derived from IEEE Std. 1003.2d-1994.

## 30685 NAME

30686 qsig — signal batch jobs

## 30687 SYNOPSIS

30688 BE qsig [-s *signal*] *job\_identifier* ...

30689

## 30690 DESCRIPTION

30691 To signal a batch job is to send a signal to the session leader of the batch job. A batch job is  
 30692 signaled by sending a request to the batch server that manages the batch job. The *qsig* utility is a  
 30693 user-accessible batch client that requests the signaling of a batch job.

30694 The *qsig* utility shall signal those batch jobs for which a batch *job\_identifier* is presented to the  
 30695 utility. The *qsig* utility shall not signal any batch jobs whose batch *job\_identifiers* are not  
 30696 presented to the utility.

30697 The *qsig* utility shall signal batch jobs in the order in which the corresponding batch  
 30698 *job\_identifiers* are presented to the utility. If the *qsig* utility fails to process a batch *job\_identifier*  
 30699 successfully, the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

30700 The *qsig* utility shall signal batch jobs by sending a *Signal Job Request* to the batch server that  
 30701 manages the batch job.

30702 For each successfully processed batch *job\_identifier*, the *qsig* utility shall have received a  
 30703 completion reply to each *Signal Job Request* sent to a batch server at the time the utility exits.

## 30704 OPTIONS

30705 The *qsig* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30706 12.2, Utility Syntax Guidelines.

30707 The following option shall be supported by the implementation:

30708 **-s *signal*** Define the signal to be sent to the batch job.

30709 The *qsig* utility shall accept a *signal* option-argument that is either a symbolic  
 30710 signal name or an unsigned integer signal number (see the POSIX.1-1990 standard,  
 30711 Section 3.3.1.1). The *qsig* utility shall accept signal names for which the SIG prefix  
 30712 has been omitted.

30713 If the *signal* option-argument is a signal name, the *qsig* utility shall send that name.

30714 If the *signal* option-argument is a number, the *qsig* utility shall send the signal  
 30715 value represented by the number.

30716 If the **-s** option is not presented to the *qsig* utility, the utility shall send the signal  
 30717 SIGTERM to each signaled batch job.

## 30718 OPERANDS

30719 The *qsig* utility shall accept one or more operands that conform to the syntax for a batch  
 30720 *job\_identifier* (see Section 3.3.1 (on page 2336)).

## 30721 STDIN

30722 Not used.

## 30723 INPUT FILES

30724 None.

30725 **ENVIRONMENT VARIABLES**

30726 The following environment variables shall affect the execution of *qsig*:

30727 *LANG* Provide a default value for the internationalization variables that are unset or null.  
30728 If *LANG* is unset or null, the corresponding value from the implementation-  
30729 defined default locale shall be used. If any of the internationalization variables  
30730 contains an invalid setting, the utility shall behave as if none of the variables had  
30731 been defined.

30732 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
30733 internationalization variables.

30734 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
30735 characters (for example, single-byte as opposed to multi-byte characters in  
30736 arguments).

30737 *LC\_MESSAGES*  
30738 Determine the locale that should be used to affect the format and contents of  
30739 diagnostic messages written to standard error.

30740 *LC\_TIME* Determine the format and contents of date and time strings written by *qsig*.

30741 *LOGNAME* Determine the login name of the user.

30742 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
30743 is not set, an unspecified system default timezone is used.

30744 **ASYNCHRONOUS EVENTS**

30745 Default.

30746 **STDOUT**

30747 An implementation of the *qsig* utility may write informative messages to standard output.

30748 **STDERR**

30749 Used only for diagnostic messages.

30750 **OUTPUT FILES**

30751 None.

30752 **EXTENDED DESCRIPTION**

30753 None.

30754 **EXIT STATUS**

30755 The following exit values shall be returned:

30756 0 Successful completion.

30757 >0 An error occurred.

30758 **CONSEQUENCES OF ERRORS**

30759 In addition to the default behavior, the *qsig* utility shall not be required to write a diagnostic  
30760 message to standard error when the error reply received from a batch server indicates that the  
30761 batch *job\_identifier* does not exist on the server. Whether or not the *qsig* utility waits to output the  
30762 diagnostic message while attempting to locate the batch job on other servers is implementation-  
30763 defined.

30764 **APPLICATION USAGE**

30765 None.

30766 **EXAMPLES**

30767 None.

30768 **RATIONALE**30769 The *qsig* utility allows users to signal batch jobs.

30770 A user may be unable to signal a batch job with the *kill* utility of the operating system for a  
30771 number of reasons. First, the process ID of the batch job may be unknown to the user. Second,  
30772 the processes of the batch job may be on a remote node. However, by virtue of communication  
30773 between batch nodes, the *qsig* utility can arrange for the signaling of a process.

30774 Because a batch job that is not running cannot be signaled, and because the signal may not  
30775 terminate the batch job, the *qsig* utility is not a substitute for the *qdel* utility.

30776 The options of the *qsig* utility allow the user to specify the signal that is to be sent to the batch  
30777 job.

30778 The *-s* option allows users to specify a signal by name or by number, and thus override the  
30779 default signal. The POSIX.1-1990 standard defines signals by both name and number.

30780 The *qsig* utility is a new utility, *vis-a-vis* existing practice; it has been defined in this volume of  
30781 IEEE Std. 1003.1-200x in response to user-perceived shortcomings in existing practice.

30782 **FUTURE DIRECTIONS**

30783 None.

30784 **SEE ALSO**30785 *kill*, *qdel*, Chapter 3 (on page 2313)30786 **CHANGE HISTORY**

30787 Derived from IEEE Std. 1003.2d-1994.

30788 **NAME**

30789 qstat — show status of batch jobs

30790 **SYNOPSIS**30791 BE qstat [-f] *job\_identifier* ...30792 qstat -Q [-f] *destination* ...30793 qstat -B [-f] *server\_name* ...

30794

30795 **DESCRIPTION**

30796 The status of a batch job, batch queue, or batch server is obtained by a request to the server. The  
 30797 *qstat* utility is a user-accessible batch client that requests the status of one or more batch jobs,  
 30798 batch queues, or servers, and writes the status information to standard output.

30799 For each successfully processed batch *job\_identifier*, the *qstat* utility shall display information  
 30800 about the corresponding batch job.

30801 For each successfully processed destination, the *qstat* utility shall display information about the  
 30802 corresponding batch queue.

30803 For each successfully processed server name, the *qstat* utility shall display information about the  
 30804 corresponding server.

30805 The *qstat* utility shall acquire batch job status information by sending a *Job Status Request* to a  
 30806 batch server. The *qstat* utility shall acquire batch queue status information by sending a *Queue*  
 30807 *Status Request* to a batch server. The *qstat* utility shall acquire server status information by  
 30808 sending a *Server Status Request* to a batch server.

30809 **OPTIONS**

30810 The *qstat* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30811 12.2, Utility Syntax Guidelines.

30812 The following options shall be supported by the implementation:

30813 **-f** Specify that a full display is produced.

30814 The minimum contents of a full display are specified in the STDOUT section.

30815 Additional contents and format of a full display are implementation-defined.

30816 **-Q** Specify that the operand is a destination.

30817 The *qstat* utility shall display information about each batch queue at each  
 30818 destination identified as an operand.

30819 **-B** Specify that the operand is a server name.

30820 The *qstat* utility shall display information about each server identified as an  
 30821 operand.

30822 **OPERANDS**

30823 If the **-Q** option is presented to the *qstat* utility, the utility shall accept one or more operands that  
 30824 conform to the syntax for a destination (see Section 3.3.2 (on page 2337)).

30825 If the **-B** option is presented to the *qstat* utility, the utility shall accept one or more *server\_name*  
 30826 operands.

30827 If neither the **-B** nor the **-Q** option is presented to the *qstat* utility, the utility shall accept one or  
 30828 more operands that conform to the syntax for a batch *job\_identifier* (see Section 3.3.1 (on page  
 30829 2336)).

30830 **STDIN**

30831 Not used.

30832 **INPUT FILES**

30833 None.

30834 **ENVIRONMENT VARIABLES**30835 The following environment variables shall affect the execution of *qstat*:

30836 *COLUMNS* Override the system-selected horizontal screen size. See the Base Definitions  
30837 volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid  
30838 values and results when it is unset or null.

30839 *HOME* Determine the path name of the user's home directory.

30840 *LANG* Provide a default value for the internationalization variables that are unset or null.  
30841 If *LANG* is unset or null, the corresponding value from the implementation-  
30842 defined default locale shall be used. If any of the internationalization variables  
30843 contains an invalid setting, the utility shall behave as if none of the variables had  
30844 been defined.

30845 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
30846 internationalization variables.

30847 *LC\_COLLATE*

30848 Determine the locale for the behavior of ranges, equivalence classes and multi-  
30849 character collating elements within regular expressions.

30850 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
30851 characters (for example, single-byte as opposed to multi-byte characters in  
30852 arguments).

30853 *LC\_MESSAGES*

30854 Determine the locale that should be used to affect the format and contents of  
30855 diagnostic messages written to standard error.

30856 *LC\_NUMERIC*

30857 Determine the locale for selecting the radix character used when writing floating-  
30858 point formatted output.

30859 *LC\_TIME* Determine the format and contents of date and time strings written by *qstat*.

30860 *LINES* Override the system-selected vertical screen size, used as the number of lines in a  
30861 screenful and the vertical screen size in visual mode. See the Base Definitions  
30862 volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables for valid  
30863 values and results when it is unset or null.

30864 *LOGNAME* Determine the login name of the user.

30865 *TERM* Determine the terminal type. If this variable is unset or null, and if the *-T* option is  
30866 not specified, an unspecified default terminal type shall be used.

30867 *TZ* Determine the timezone in which the time and date are written. If the *TZ* variable  
30868 is not set, an unspecified system default timezone is used.

30869 **ASYNCHRONOUS EVENTS**

30870 Default.

30871 **STDOUT**

30872 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is not  
 30873 specified, the *qstat* utility shall display the following items on a single line, in the stated order,  
 30874 with white space between each item, for each successfully processed operand:

- 30875 • The batch *job\_identifier*
- 30876 • The batch job name
- 30877 • The *Job\_Owner* attribute
- 30878 • The CPU time used by the batch job
- 30879 • The batch job state
- 30880 • The batch job location

30881 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is specified,  
 30882 the *qstat* utility shall display the following items for each success fully processed operand:

- 30883 • The batch *job\_identifier*
- 30884 • The batch job name
- 30885 • The *Job\_Owner* attribute
- 30886 • The execution user ID
- 30887 • The CPU time used by the batch job
- 30888 • The batch job state
- 30889 • The batch job location
- 30890 • Additional implementation-defined information, if any, about the batch job or batch queue

30891 If an operand presented to the *qstat* utility is a destination, the *-Q* option is specified, and the *-f*  
 30892 option is not specified, the *qstat* utility shall display the following items on a single line, in the  
 30893 stated order, with white space between each item, for each successfully processed operand:

- 30894 • The batch queue name
- 30895 • The maximum number of batch jobs that are allowed to run in the batch queue concurrently
- 30896 • The total number of batch jobs in the batch queue
- 30897 • The status of the batch queue
- 30898 • For each state, the number of batch jobs in that state in the batch queue and the name of the  
 30899 state
- 30900 • The type of batch queue (execution or routing)

30901 If the operands presented to the *qstat* utility are destinations, the *-Q* option is specified, and the  
 30902 *-f* option is specified, the *qstat* utility shall display the following items for each successfully  
 30903 processed operand:

- 30904 • The batch queue name
- 30905 • The maximum number of batch jobs that are allowed to run in the batch queue concurrently
- 30906 • The total number of batch jobs in the batch queue
- 30907 • The status of the batch queue

- 30908           • For each state, the number of batch jobs in that state in the batch queue and the name of the  
30909           state
- 30910           • The type of batch queue (execution or routing)
- 30911           • Additional implementation-defined information, if any, about the batch queue
- 30912           If the operands presented to the *qstat* utility are batch server names, the **-B** option is specified,  
30913           and the **-f** option is not specified, the *qstat* utility shall display the following items on a single  
30914           line, in the stated order, with white space between each item, for each successfully processed  
30915           operand:
- 30916           • The batch server name
- 30917           • The maximum number of batch jobs that are allowed to run in the batch queue concurrently
- 30918           • The total number of batch jobs managed by the batch server
- 30919           • The status of the batch server
- 30920           • For each state, the number of batch jobs in that state and the name of the state
- 30921           If the operands presented to the *qstat* utility are server names, the **-B** option is specified, and the  
30922           **-f** option is specified, the *qstat* utility shall display the following items for each successfully  
30923           processed operand:
- 30924           • The server name
- 30925           • The maximum number of batch jobs that are allowed to run in the batch queue concurrently
- 30926           • The total number of batch jobs managed by the server
- 30927           • The status of the server
- 30928           • For each state, the number of batch jobs in that state and the name of the state
- 30929           • Additional implementation-defined information, if any, about the server
- 30930 **STDERR**
- 30931           Used only for diagnostic messages.
- 30932 **OUTPUT FILES**
- 30933           None.
- 30934 **EXTENDED DESCRIPTION**
- 30935           None.
- 30936 **EXIT STATUS**
- 30937           The following exit values shall be returned:
- 30938           0   Successful completion.
- 30939           >0  An error occurred.
- 30940 **CONSEQUENCES OF ERRORS**
- 30941           In addition to the default behavior, the *qstat* utility shall not be required to write a diagnostic  
30942           message to standard error when the error reply received from a batch server indicates that the  
30943           batch *job\_identifier* does not exist on the server. Whether or not the *qstat* utility waits to output  
30944           the diagnostic message while attempting to locate the batch job on other servers is  
30945           implementation-defined.



**30946 APPLICATION USAGE**

30947           None.

**30948 EXAMPLES**

30949           None.

**30950 RATIONALE**

30951           The *qstat* utility allows users to display the status of jobs and listing the batch jobs in queues.

30952           The operands of the *qstat* utility may be either job identifiers, queues (specified as destination  
30953 identifiers), or batch server names. The **-Q** and **-B** options, or absence thereof, indicate the  
30954 nature of the operands.

30955           The other options of the *qstat* utility allow the user to control the amount of information  
30956 displayed and the format in which it is displayed. Should a user wish to display the status of a  
30957 set of jobs that match a selection criteria, the *qselect* utility may be used to acquire such a list.

30958           The **-f** option allows users to request a “full” display in an implementation-defined format. |

30959           Historically, the *qstat* utility has been a part of the NQS and its derivatives, the existing practice |  
30960 on which it is based.

**30961 FUTURE DIRECTIONS**

30962           None.

**30963 SEE ALSO**

30964           *qselect*, Chapter 3 (on page 2313)

**30965 CHANGE HISTORY**

30966           Derived from IEEE Std. 1003.2d-1994.

## 30967 NAME

30968 qsub — submit a script

## 30969 SYNOPSIS

```

30970 BE qsub [-a date_time][-A account_string][-c interval]
30971 [-C directive_prefix][-e path_name][-h][-j join_list][-k keep_list]
30972 [-m mail_options][-M mail_list][-N name]
30973 [-o path_name][-p priority][-q destination][-r y|n]
30974 [-S path_name_list][-u user_list][-v variable_list][-V]
30975 [-z][script]
30976

```

## 30977 DESCRIPTION

30978 To submit a script is to create a batch job that executes the script. A script is submitted by a  
 30979 request to a batch server. The *qsub* utility is a user-accessible batch client that submits a script.

30980 Upon successful completion, the *qsub* utility shall have created a batch job that will execute the  
 30981 submitted script.

30982 The *qsub* utility shall submit a script by sending a *Queue Job Request* to a batch server.

30983 The *qsub* utility shall place the value of the following environment variables in the *Variable\_List*  
 30984 attribute of the batch job: *HOME*, *LANG*, *LOGNAME*, *PATH*, *MAIL*, *SHELL*, and *TZ*. The name  
 30985 of the environment variable shall be the current name prefixed with the string *PBS\_O\_*.

30986 **Note:** If the current value of the *HOME* variable in the environment space of the *qsub* utility  
 30987 is */aa/bb/cc*, then *qsub* shall place *PBS\_O\_HOME=/aa/bb/cc* in the *Variable\_List*  
 30988 attribute of the batch job.

30989 In addition to the variables described above, the *qsub* utility shall add the following variables  
 30990 with the indicated values to the variable list:

30991 *PBS\_O\_WORKDIR* The absolute path of the current working directory of the *qsub* utility process.

30992 *PBS\_O\_HOST* The name of the host on which the *qsub* utility is running.

## 30993 OPTIONS

30994 The *qsub* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 30995 12.2, Utility Syntax Guidelines.

30996 The following options shall be supported by the implementation:

30997 **-a date\_time** Define the time at which a batch job becomes eligible for execution.

30998 The *qsub* utility shall accept an option-argument that conforms to the syntax of the  
 30999 *date\_time* operand of the *touch* utility.

31000

**Table 4-18** Environment Variable Values (Utilities)

31001

31002

31003

31004

31005

31006

31007

31008

31009

31010

| Variable Name        | Value at qsub Time        |
|----------------------|---------------------------|
| <i>PBS_O_HOME</i>    | <i>HOME</i>               |
| <i>PBS_O_HOST</i>    | Client host name          |
| <i>PBS_O_LANG</i>    | <i>LANG</i>               |
| <i>PBS_O_LOGNAME</i> | <i>LOGNAME</i>            |
| <i>PBS_O_PATH</i>    | <i>PATH</i>               |
| <i>PBS_O_MAIL</i>    | <i>MAIL</i>               |
| <i>PBS_O_SHELL</i>   | <i>SHELL</i>              |
| <i>PBS_O_TZ</i>      | <i>TZ</i>                 |
| <i>PBS_O_WORKDIR</i> | Current working directory |

31011

31012

31013

**Note:** The server that initiates execution of the batch job will add other variables to the batch job's environment; see Section 3.2.2.1 (on page 2319).

31014

31015

31016

31017

The *qsub* utility shall set the *Execution\_Time* attribute of the batch job to the number of seconds since the Epoch that is equivalent to the local time expressed by the value of the *date\_time* option-argument. The Epoch is defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 3.151, Epoch.

31018

31019

31020

If the *-a* option is not presented to the *qsub* utility, the utility shall set the *Execution\_Time* attribute of the batch job to a time (number of seconds since the Epoch) that is earlier than the time at which the utility exits.

31021

**-A** *account\_string*

31022

31023

Define the account to which the resource consumption of the batch job should be charged.

31024

The syntax of the *account\_string* option-argument is unspecified.

31025

31026

The *qsub* utility shall set the *Account\_Name* attribute of the batch job to the value of the *account\_string* option-argument.

31027

31028

If the *-A* option is not presented to the *qsub* utility, the utility shall omit the *Account\_Name* attribute from the attributes of the batch job.

31029

**-c** *interval*

Define whether the batch job should be checkpointed, and if so, how often.

31030

31031

The *qsub* utility shall accept a value for the interval option-argument that is one of the following:

31032

31033

*n* No checkpointing shall be performed on the batch batch job (NO\_CHECKPOINT).

31034

31035

*s* Checkpointing shall be performed only when the batch server is shut down (CHECKPOINT\_AT\_SHUTDOWN).

31036

31037

31038

*c* Automatic periodic checkpointing shall be performed at the *Minimum\_Cpu\_Interval* attribute of the batch queue, in units of CPU minutes (CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL).

31039

31040

31041

31042

*c=minutes* Automatic periodic checkpointing shall be performed every minutes of CPU time, or every *Minimum\_Cpu\_Interval* minutes, whichever is greater. The *minutes* argument shall conform to the syntax for unsigned integers and shall be greater than zero.

- 31043 The *qsub* utility shall set the *Checkpoint* attribute of the batch job to the value of the  
31044 *interval* option-argument.
- 31045 If the *-c* option is not presented to the *qsub* utility, the utility shall set the  
31046 *Checkpoint* attribute of the batch job to the single character 'u'  
31047 (CHECKPOINT\_UNSPECIFIED).
- 31048 *-C directive\_prefix*  
31049 Define the prefix that declares a directive to the *qsub* utility within the script.
- 31050 The *directive\_prefix* is not a batch job attribute; it affects the behavior of the *qsub*  
31051 utility.
- 31052 If the *-C* option is presented to the *qsub* utility, and the value of the *directive\_prefix*  
31053 option-argument is the null string, the utility shall not scan the script file for  
31054 directives. If the *-C* option is not presented to the *qsub* utility, then the value of the  
31055 *PBS\_DPREFIX* environment variable is used. If the environment variable is not  
31056 defined, then #PBS encoded in the portable character set is the default.
- 31057 *-e path\_name* Define the path to be used for the standard error stream of the batch job.
- 31058 The *qsub* utility shall accept a *path\_name* option-argument which can be preceded  
31059 by a host name element of the form *hostname:*.
- 31060 If the *path\_name* option-argument constitutes an absolute path name, the *qsub*  
31061 utility shall set the *Error\_Path* attribute of the batch job to the value of the  
31062 *path\_name* option-argument.
- 31063 If the *path\_name* option-argument constitutes a relative path name and no host  
31064 name element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the  
31065 batch job to the value of the absolute path name derived by expanding the  
31066 *path\_name* option-argument relative to the current directory of the process  
31067 executing *qsub*.
- 31068 If the *path\_name* option-argument constitutes a relative path name and a host name  
31069 element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the batch  
31070 job to the value of the *path\_name* option-argument without expansion. The host  
31071 name element shall be included.
- 31072 If the *path\_name* option-argument does not include a host name element, the *qsub*  
31073 utility shall prefix the path name with *hostname:*, where *hostname* is the name of the  
31074 host upon which the *qsub* utility is being executed.
- 31075 If the *-e* option is not presented to the *qsub* utility, the utility shall set the  
31076 *Error\_Path* attribute of the batch job to the host name and path of the current  
31077 directory of the submitting process and the default file name.
- 31078 The default file name for standard error has the following format:  
31079 *job\_name.e**sequence\_number*
- 31080 *-h* Specify that a USER hold is applied to the batch job.
- 31081 The *qsub* utility shall set the value of the *Hold\_Types* attribute of the batch job to the  
31082 value USER.
- 31083 If the *-h* option is not presented to the *qsub* utility, the utility shall set the  
31084 *Hold\_Types* attribute of the batch job to the value NO\_HOLD.
- 31085 *-j join\_list* Define which streams of the batch job are to be merged. The *qsub -j* option shall  
31086 accept a value for the *join\_list* option-argument that is a string of alphanumeric

31087 characters in the portable character set (see the Base Definitions volume of  
 31088 IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

31089 The *qsub* utility shall accept a *join\_list* option-argument that consists of one or  
 31090 more of the characters 'e' and 'o' or the single character 'n'.

31091 All of the other batch job output streams specified will be merged into the output  
 31092 stream represented by the character listed first in the *join\_list* option-argument.

31093 For each unique character in the *join\_list* option-argument, the *qsub* utility shall  
 31094 add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 31095 a different batch job stream to join:

31096 *e* The standard error of the batch batch job (JOIN\_STD\_ERROR).

31097 *o* The standard output of the batch batch job (JOIN\_STD\_OUTPUT).

31098 An existing *Join\_Path* attribute can be cleared by the following join type:

31099 **n** NO\_JOIN

31100 If **n** is specified, then no files are joined. The *qsub* utility shall consider it an error if  
 31101 any join type other than **n** is combined with join type **n**.

31102 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 31103 'n' within the *join\_list* option-argument. The *qsub* utility shall permit the  
 31104 repetition of characters, but shall not assign additional meaning to the repeated  
 31105 characters.

31106 An implementation may define other join types. The conformance document for an  
 31107 implementation shall describe any additional batch job streams, how they are  
 31108 specified, their internal behavior, and how they affect the behavior of the utility.

31109 If the **-j** option is not presented to the *qsub* utility, the utility shall set the value of  
 31110 the *Join\_Path* attribute of the batch job to NO\_JOIN.

31111 **-k keep\_list** Define which output of the batch job to retain on the execution host.

31112 The *qsub* **-k** option shall accept a value for the *keep\_list* option-argument that is a  
 31113 string of alphanumeric characters in the portable character set (see the Base  
 31114 Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).

31115 The *qsub* utility shall accept a *keep\_list* option-argument that consists of one or  
 31116 more of the characters 'e' and 'o' or the single character 'n'.

31117 For each unique character in the *keep\_list* option-argument, the *qsub* utility shall  
 31118 add a value to the *Keep\_Files* attribute of the batch job as follows, each representing  
 31119 a different batch job stream to keep:

31120 *e* The standard error of the batch batch job (KEEP\_STD\_ERROR).

31121 *o* The standard output of the batch batch job (KEEP\_STD\_OUTPUT).

31122 If both *e* and *o* are specified, then both files are retained. An existing *Keep\_Files*  
 31123 attribute can be cleared by the following keep type:

31124 **n** NO\_KEEP

31125 If **n** is specified, then no files are retained. The *qsub* utility shall consider it an error  
 31126 if any keep type other than **n** is combined with keep type **n**.

31127 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 31128 'n' within the *keep\_list* option-argument. The *qsub* utility shall permit the

- 31129 repetition of characters, but shall not assign additional meaning to the repeated  
31130 characters.
- 31131 An implementation may define other keep types. The conformance document for  
31132 an implementation shall describe any additional keep types, how they are  
31133 specified, their internal behavior, and how they affect the behavior of the utility. If  
31134 the `-k` option is not presented to the *qsub* utility, the utility shall set the *Keep\_Files*  
31135 attribute of the batch job to the value `NO_KEEP`.
- 31136 **-m** *mail\_options*
- 31137 Define the points in the execution of the batch job at which the batch server that  
31138 manages the batch job shall send mail about a change in the state of the batch job.
- 31139 The *qsub -m* option shall accept a value for the *mail\_options* option-argument that  
31140 is a string of alphanumeric characters in the portable character set (see the Base  
31141 Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character Set).
- 31142 The *qsub* utility shall accept a value for the *mail\_options* option-argument that is a  
31143 string of one or more of the characters 'e', 'b', and 'a', or the single character  
31144 'n'.
- 31145 For each unique character in the *mail\_options* option-argument, the *qsub* utility shall  
31146 add a value to the *Mail\_Users* attribute of the batch job as follows, each  
31147 representing a different time during the life of a batch job at which to send mail:
- 31148 e MAIL\_AT\_EXIT
- 31149 b MAIL\_AT\_BEGINNING
- 31150 a MAIL\_AT\_ABORT
- 31151 If any of these characters are duplicated in the *mail\_options* option-argument, the  
31152 duplicates shall be ignored.
- 31153 An existing *Mail\_Points* attribute can be cleared by the following mail type:
- 31154 n NO\_MAIL
- 31155 If **n** is specified, then mail is not sent. The *qsub* utility shall consider it an error if  
31156 any mail type other than **n** is combined with mail type **n**.
- 31157 Strictly conforming applications shall not repeat any of the characters 'e', 'b',  
31158 'a', or 'n' within the *mail\_options* option-argument.
- 31159 The *qsub* utility shall permit the repetition of characters, but shall not assign  
31160 additional meaning to the repeated characters. An implementation may define  
31161 other mail types. The conformance document for an implementation shall describe  
31162 any additional mail types, how they are specified, their internal behavior, and how  
31163 they affect the behavior of the utility.
- 31164 If the `-m` option is not presented to the *qsub* utility, the utility shall set the  
31165 *Mail\_Points* attribute to the value `MAIL_AT_ABORT`.
- 31166 **-M** *mail\_list* Define the list of users to which a batch server that executes the batch job shall  
31167 send mail, if the server sends mail about the batch job.
- 31168 The syntax of the *mail\_list* option-argument is unspecified.
- 31169 If the implementation of the *qsub* utility uses a name service to locate users, the  
31170 utility should accept the syntax used by the name service.

- 31171 If the implementation of the *qsub* utility does not use a name service to locate  
31172 users, the implementation should accept the following syntax for user names:
- 31173 *mail\_address*[ , , *mail\_address* , , ... ]
- 31174 The interpretation of *mail\_address* is implementation-defined.
- 31175 The *qsub* utility shall set the *Mail\_Users* attribute of the batch job to the value of the  
31176 *mail\_list* option-argument.
- 31177 If the **-M** option is not presented to the *qsub* utility, the utility shall place only the  
31178 user name and host name for the current process in the *Mail\_Users* attribute of the  
31179 batch job.
- 31180 **-N name** Define the name of the batch job.
- 31181 The *qsub* **-N** option shall accept a value for the *name* option-argument that is a  
31182 string of up to 15 alphanumeric characters in the portable character set (see the  
31183 Base Definitions volume of IEEE Std. 1003.1-200x, Section 6.1, Portable Character  
31184 Set) where the first character is alphabetic.
- 31185 The *qsub* utility shall set the value of the *Job\_Name* attribute of the batch job to the  
31186 value of the *name* option-argument.
- 31187 If the **-N** option is not presented to the *qsub* utility, the utility shall set the  
31188 *Job\_Name* attribute of the batch job to the name of the *script* argument from which  
31189 the directory specification if any, has been removed.
- 31190 If the **-N** option is not presented to the *qsub* utility, and the script is read from  
31191 standard input, the utility shall set the *Job\_Name* attribute of the batch job to the  
31192 value STDIN.
- 31193 **-o path\_name** Define the path for the standard output of the batch job.
- 31194 The *qsub* utility shall accept a *path\_name* option-argument that conforms to the  
31195 syntax of the *path\_name* element defined in the POSIX.1-1990 standard, which can  
31196 be preceded by a host name element of the form *hostname*:
- 31197 If the *path\_name* option-argument constitutes an absolute path name, the *qsub*  
31198 utility shall set the *Output\_Path* attribute of the batch job to the value of the  
31199 *path\_name* option-argument without expansion.
- 31200 If the *path\_name* option-argument constitutes a relative path name and no host  
31201 name element is specified, the *qsub* utility shall set the *Output\_Path* attribute of the  
31202 batch job to the path name derived by expanding the value of the *path\_name*  
31203 option-argument relative to the current directory of the process executing the *qsub*.
- 31204 If the *path\_name* option-argument constitutes a relative path name and a host name  
31205 element is specified, the *qsub* utility shall set the *Output\_Path* attribute of the batch  
31206 job to the value of the *path\_name* option-argument without expansion.
- 31207 If the *path\_name* option-argument does not specify a host name element, the *qsub*  
31208 utility shall prefix the path name with *hostname*:, where *hostname* is the name of the  
31209 host upon which the *qsub* utility is executing.
- 31210 If the **-o** option is not presented to the *qsub* utility, the utility shall set the  
31211 *Output\_Path* attribute of the batch job to the host name and path of the current  
31212 directory of the submitting process and the default file name.
- 31213 The default file name for standard output has the following format:

|       |                          |                                                                                                                                                                                                                                      |
|-------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31214 |                          | <i>job_name.osequence_number</i>                                                                                                                                                                                                     |
| 31215 | <b>-p priority</b>       | Define the priority the batch job should have relative to other batch jobs owned by the batch server.                                                                                                                                |
| 31216 |                          |                                                                                                                                                                                                                                      |
| 31217 |                          | The <i>qsub</i> utility shall set the <i>Priority</i> attribute of the batch job to the value of the <i>priority</i> option-argument.                                                                                                |
| 31218 |                          |                                                                                                                                                                                                                                      |
| 31219 |                          | If the <b>-p</b> option is not presented to the <i>qsub</i> utility, the value of the <i>Priority</i> attribute is implementation-defined.                                                                                           |
| 31220 |                          |                                                                                                                                                                                                                                      |
| 31221 |                          | The <i>qsub</i> utility shall accept a value for the <i>priority</i> option-argument that conforms to the syntax for signed decimal integers, and which is not less than -1 024 and not greater than 1 023.                          |
| 31222 |                          |                                                                                                                                                                                                                                      |
| 31223 |                          |                                                                                                                                                                                                                                      |
| 31224 | <b>-q destination</b>    | Define the destination of the batch job.                                                                                                                                                                                             |
| 31225 |                          |                                                                                                                                                                                                                                      |
| 31226 |                          | The destination is not a batch job attribute; it determines the batch server, and possibly the batch queue, to which the <i>qsub</i> utility batch queues the batch job.                                                             |
| 31227 |                          |                                                                                                                                                                                                                                      |
| 31228 |                          | The <i>qsub</i> utility shall submit the script to the batch server named by the <i>destination</i> option-argument or the server that owns the batch queue named in the <i>destination</i> option-argument.                         |
| 31229 |                          |                                                                                                                                                                                                                                      |
| 31230 |                          |                                                                                                                                                                                                                                      |
| 31231 |                          | The <i>qsub</i> utility shall accept an option-argument for the <b>-q</b> option that conforms to the syntax for a destination (see Section 3.3.2 (on page 2337)).                                                                   |
| 31232 |                          |                                                                                                                                                                                                                                      |
| 31233 |                          | If the <b>-q</b> option is not presented to the <i>qsub</i> utility, the <i>qsub</i> utility shall submit the batch job to the default destination. The mechanism for determining the default destination is implementation-defined. |
| 31234 |                          |                                                                                                                                                                                                                                      |
| 31235 |                          |                                                                                                                                                                                                                                      |
| 31236 | <b>-r y   n</b>          | Define whether the batch job is rerunnable.                                                                                                                                                                                          |
| 31237 |                          |                                                                                                                                                                                                                                      |
| 31238 |                          | If the value of the option-argument is <i>y</i> , the <i>qsub</i> utility shall set the <i>Rerunnable</i> attribute of the batch job to TRUE.                                                                                        |
| 31239 |                          |                                                                                                                                                                                                                                      |
| 31240 |                          | If the value of the option-argument is <i>n</i> , the <i>qsub</i> utility shall set the <i>Rerunnable</i> attribute of the batch job to FALSE.                                                                                       |
| 31241 |                          |                                                                                                                                                                                                                                      |
| 31242 |                          | If the <b>-r</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>Rerunnable</i> attribute of the batch job to TRUE.                                                                                 |
| 31243 | <b>-S path_name_list</b> | Define the path name to the shell under which the batch job is to execute.                                                                                                                                                           |
| 31244 |                          |                                                                                                                                                                                                                                      |
| 31245 |                          | The <i>qsub</i> utility shall accept a <i>path_name_list</i> option-argument that conforms to the following syntax:                                                                                                                  |
| 31246 |                          |                                                                                                                                                                                                                                      |
| 31247 |                          | <i>pathname[@host][ , , pathname[@host] , , ... ]</i>                                                                                                                                                                                |
| 31248 |                          | The <i>qsub</i> utility shall allow only one path name for a given host name. The <i>qsub</i> utility shall allow only one path name that is missing a corresponding host name.                                                      |
| 31249 |                          |                                                                                                                                                                                                                                      |
| 31250 |                          | The <i>qsub</i> utility shall add a value to the <i>Shell_Path_List</i> attribute of the batch job for each entry in the <i>path_name_list</i> option-argument.                                                                      |
| 31251 |                          |                                                                                                                                                                                                                                      |
| 31252 |                          | If the <b>-S</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>Shell_Path_List</i> attribute of the batch job to the null string.                                                                 |
| 31253 |                          |                                                                                                                                                                                                                                      |
| 31254 |                          | The conformance document for an implementation shall describe the mechanism used to set the default shell and determine the current value of the default shell.                                                                      |
| 31255 |                          |                                                                                                                                                                                                                                      |



- 31256 An implementation shall provide a means for the installation to set the default  
31257 shell to the login shell of the user under which the batch job is to execute. See  
31258 Section 3.3.3 (on page 2337) for a means of removing *keyword=value* (and  
31259 *value@keyword*) pairs and other general rules for list-oriented batch job attributes.
- 31260     **-u *user\_list*** Define the user name under which the batch job is to execute.
- 31261 The *qsub* utility shall accept a *user\_list* option-argument that conforms to the  
31262 following syntax:
- 31263     *username[@host][, ,username[@host], , ...]*
- 31264 The *qsub* utility shall accept only one user name that is missing a corresponding  
31265 host name. The *qsub* utility shall accept only one user name per named host.
- 31266 The *qsub* utility shall add a value to the *User\_List* attribute of the batch job for each  
31267 entry in the *user\_list* option-argument.
- 31268 If the **-u** option is not presented to the *qsub* utility, the utility shall set the *User\_List*  
31269 attribute of the batch job to the user name from which the utility is executing. See  
31270 Section 3.3.3 (on page 2337) for a means of removing *keyword=value* (and  
31271 *value@keyword*) pairs and other general rules for list-oriented batch job attributes.
- 31272     **-v *variable\_list***
- 31273 Add to the list of variables that are exported to the session leader of the batch job.
- 31274 A *variable\_list* is a set of strings of either the form *<variable>* or *<variable=value>*,  
31275 delimited by commas.
- 31276 If the **-v** option is presented to the *qsub* utility, the utility shall also add, to the  
31277 environment *Variable\_List* attribute of the batch job, every variable named in the  
31278 environment *variable\_list* option-argument and, optionally, values of specified  
31279 variables.
- 31280 If a value is not provided on the command line, the *qsub* utility shall set the value  
31281 of each variable in the environment *Variable\_List* attribute of the batch job to the  
31282 value of the corresponding environment variable for the process in which the  
31283 utility is executing; see Table 4-18 (on page 3013).
- 31284 A conforming application shall not repeat a variable in the environment  
31285 *variable\_list* option-argument.
- 31286 The *qsub* utility shall not repeat a variable in the environment *Variable\_List*  
31287 attribute of the batch job. See Section 3.3.3 (on page 2337) for a means of removing  
31288 *keyword=value* (and *value@keyword*) pairs and other general rules for list-oriented  
31289 batch job attributes.
- 31290     **-V** Specify that all of the environment variables of the process are exported to the  
31291 context of the batch job.
- 31292 The *qsub* utility shall place every environment variable in the process in which the  
31293 utility is executing in the list and shall set the value of each variable in the attribute  
31294 to the value of that variable in the process.
- 31295     **-z** Specify that the utility does not write the batch *job\_identifier* of the created batch  
31296 job to standard output.
- 31297 If the **-z** option is presented to the *qsub* utility, the utility shall not write the batch  
31298 *job\_identifier* of the created batch job to standard output.

31299 If the `-z` option is not presented to the `qsub` utility, the utility shall write the  
31300 identifier of the created batch job to standard output.

### 31301 OPERANDS

31302 The `qsub` utility shall accept a `script` operand that indicates the path to the script of the batch job.

31303 If the `script` operand is not presented to the `qsub` utility, or if the operand is the single-character  
31304 string `'-'`, the utility shall read the script from standard input.

31305 If the script represents a partial path, the `qsub` utility shall expand the path relative to the current  
31306 directory of the process executing the utility.

### 31307 STDIN

31308 The `qsub` utility reads the script of the batch job from standard input if the script operand is  
31309 omitted or is the single character `'-'`.

### 31310 INPUT FILES

31311 In addition to binding the file indicated by the `script` operand to the batch job, the `qsub` utility  
31312 reads the script file and acts on directives in the script.

### 31313 ENVIRONMENT VARIABLES

31314 The following environment variables shall affect the execution of `qsub`:

31315 **LANG** Provide a default value for the internationalization variables that are unset or null.  
31316 If `LANG` is unset or null, the corresponding value from the implementation-  
31317 defined default locale shall be used. If any of the internationalization variables  
31318 contains an invalid setting, the utility shall behave as if none of the variables had  
31319 been defined.

31320 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
31321 internationalization variables.

31322 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
31323 characters (for example, single-byte as opposed to multi-byte characters in  
31324 arguments).

31325 **LC\_MESSAGES**  
31326 Determine the locale that should be used to affect the format and contents of  
31327 diagnostic messages written to standard error.

31328 **LC\_TIME** Determine the format and contents of date and time strings written by `qsub`.

31329 **LOGNAME** Determine the login name of the user.

31330 **PBS\_DPREFIX**  
31331 Determine the default prefix for directives within the script.

31332 **SHELL** Determine the path name of the preferred command language interpreter of the  
31333 user.

31334 **TZ** Determine the timezone in which the time and date are written. If the `TZ` variable  
31335 is not set, an unspecified system default timezone is used.

### 31336 ASYNCHRONOUS EVENTS

31337 Once created, a batch job exists until it exits, aborts, or is deleted.

31338 After a batch job is created by the `qsub` utility, batch servers might route, execute, modify, or  
31339 delete the batch job.

31340 **STDOUT**

31341 The *qsub* utility writes the batch *job\_identifier* assigned to the batch job to standard output, unless  
31342 the **-z** option is specified.

31343 **STDERR**

31344 Used only for diagnostic messages.

31345 **OUTPUT FILES**

31346 None.

31347 **EXTENDED DESCRIPTION**31348 **Script Preservation**

31349 The *qsub* utility shall make the script available to the server executing the batch job in such a way  
31350 that the server executes the script as it exists at the time of submission.

31351 The *qsub* utility can send a copy of the script to the server with the *Queue Job Request* or store a  
31352 temporary copy of the script in a location specified to the server.

31353 **Option Specification**

31354 A script can contain directives to the *qsub* utility.

31355 The *qsub* utility shall scan the lines of the script for directives, skipping blank lines, until the first  
31356 line that begins with a string other than the directive string; if directives occur on subsequent  
31357 lines, the utility shall ignore those directives.

31358 Lines are separated by a <newline>. If the first line of the script begins with "#!" or a colon  
31359 (' : '), then it is skipped. The *qsub* utility shall process a line in the script as a directive if and  
31360 only if the string of characters from the first non-white-space character on the line until the first  
31361 <space> or <tab> character on the line match the directive prefix. If a line in the script contains a  
31362 directive and the final characters of the line are backslash ('\ ') and <newline>, then the next  
31363 line shall be interpreted as a continuation of that directive.

31364 The *qsub* utility shall process the options and option-arguments contained on the directive prefix  
31365 line using the same syntax as if the options were input on the *qsub* utility.

31366 The *qsub* utility shall continue to process a directive prefix line until after a <newline> is  
31367 encountered. An implementation may ignore lines which, according to the syntax of the shell  
31368 that will interpret the script, are comments. An implementation shall describe in the  
31369 conformance document the format of any shell comments that it will recognize.

31370 If an option is present in both a directive and the arguments to the *qsub* utility, the utility shall  
31371 ignore the option and the corresponding option-argument, if any, in the directive.

31372 If an option that is present in the directive is not present in the arguments to the *qsub* utility, the  
31373 utility shall process the option and the option-argument, if any.

31374 In order of preference, the *qsub* utility shall select the directive prefix from one of the following  
31375 sources:

- 31376 • If the **-C** option is presented to the utility, the value of the *directive\_prefix* option-argument
- 31377 • If the environment variable *PBS\_DPREFIX* is defined, the value of that variable
- 31378 • The four-character string "#PBS" encoded in the portable character set

31379 If the **-C** option is present in the script file it shall be ignored.

**31380 EXIT STATUS**

31381 The following exit values shall be returned:

31382 0 Successful completion.

31383 >0 An error occurred.

**31384 CONSEQUENCES OF ERRORS**

31385 Default.

**31386 APPLICATION USAGE**

31387 None.

**31388 EXAMPLES**

31389 None.

**31390 RATIONALE**

31391 The *qsub* utility allows users to create a batch job that will process the script specified as the  
31392 operand of the utility.

31393 The options of the *qsub* utility allow users to control many aspects of the queuing and execution  
31394 of a batch job.

31395 The **-a** option allows users to designate the time after which the batch job will become eligible to  
31396 run. By specifying an execution time, users can take advantage of resources at off-peak hours,  
31397 synchronize jobs with chronologically predictable events, and perhaps take advantage of off-  
31398 peak pricing of computing time. For these reasons and others, a timing option is existing practice  
31399 on the part of almost every batch system, including NQS.

31400 The **-A** option allows users to specify the account that will be charged for the batch job. Support  
31401 for account is not mandatory for conforming batch servers.

31402 The **-C** option allows users to prescribe the prefix for directives within the script file. The default  
31403 prefix "#PBS" may be inappropriate if the script will be interpreted with an alternate shell, as  
31404 specified by the **-S** option.

31405 The **-c** option allows users to establish the checkpointing interval for their jobs. A checkpointing  
31406 system, which is not defined by this volume of IEEE Std. 1003.1-200x, allows recovery of a batch  
31407 job at the most recent checkpoint in the event of a crash. Checkpointing is typically used for jobs  
31408 that consume expensive computing time or must meet a critical schedule. Users should be  
31409 allowed to make the tradeoff between the overhead of checkpointing and the risk to the timely  
31410 completion of the batch job; therefore, this volume of IEEE Std. 1003.1-200x provides the  
31411 checkpointing interval option. Support for checkpointing is optional for batch servers.

31412 The **-e** option allows users to redirect the standard error streams of their jobs to a non-default  
31413 path. For example, if the submitted script generally produces a great deal of useless error output,  
31414 a user might redirect the standard error output to the null device. Or, if the file system holding  
31415 the default location (the home directory of the user) has too little free space, the user might  
31416 redirect the standard error stream to a file in another file system.

31417 The **-h** option allows users to create a batch job that is held until explicitly released. The ability  
31418 to create a held job is useful when some external event must complete before the batch job can  
31419 execute. For example, the user might submit a held job and release it when the system load has  
31420 dropped.

31421 The **-j** option allows users to merge the standard error of a batch job into its standard output  
31422 stream, which has the advantage of showing the sequential relationship between output and  
31423 error messages.

- 31424 The **-m** option allows users to designate those points in the execution of a batch job at which  
31425 mail will be sent to the submitting user, or to the account(s) indicated by the **-M** option. By  
31426 requesting mail notification at points of interest in the life of a job, the submitting user, or other  
31427 designated users, can track the progress of a batch job.
- 31428 The **-N** option allows users to associate a name with the batch job. The job name in no way  
31429 affects the processing of the batch job, but rather serves as a mnemonic handle for users. For  
31430 example, the batch job name can help the user distinguish between multiple jobs listed by the  
31431 *qstat* utility.
- 31432 The **-o** option allows users to redirect the standard output stream. A user might, for example,  
31433 wish to redirect to the null device the standard output stream of a job that produces copious yet  
31434 superfluous output.
- 31435 The **-P** option allows users to designate the relative priority of a batch job for selection from a  
31436 queue.
- 31437 The **-q** option allows users to specify an initial queue for the batch job. If the user specifies a  
31438 routing queue, the batch server routes the batch job to another queue for execution or  
31439 further routing. If the user specifies a non-routing queue, the batch server of the queue  
31440 eventually executes the batch job.
- 31441 The **-r** option allows users to control whether the submitted job will be rerun if the controlling  
31442 batch node fails during execution of the batch job. The **-r** option likewise allows users to  
31443 indicate whether or not the batch job is eligible to be rerun by the *qrerun* utility. Some jobs cannot  
31444 be correctly rerun because of changes they make in the state of databases or other aspects of  
31445 their environment. This volume of IEEE Std. 1003.1-200x specifies that the default, if the **-r**  
31446 option is not presented to the utility, will be that the batch job cannot be rerun, since the result of  
31447 rerunning a non-rerunable job might be catastrophic.
- 31448 The **-S** option allows users to specify the program (usually a shell) that will be invoked to  
31449 process the script of the batch job. This option has been modified to allow a list of shell names  
31450 and locations associated with different hosts.
- 31451 The **-u** option is useful when the submitting user is authorized to use more than one account on  
31452 a given host, in which case the **-u** option allows the user to select from among those accounts.  
31453 The option-argument is a list of user-host pairs, so that the submitting user can provide different  
31454 user identifiers for different nodes in the event the batch job is routed. The **-u** option provides a  
31455 lot of flexibility to accommodate sites with complex account structures. Users that have the  
31456 same user identifier on all the hosts they are authorized to use will not need to use the **-u** option.
- 31457 The **-V** option allows users to export all their current environment variables, as of the time the  
31458 batch job is submitted, to the context of the processes of the batch job.
- 31459 The **-v** option allows users to export specific environment variables from their current process  
31460 to the processes of the batch job.
- 31461 The **-z** option allows users to suppress the writing of the batch job identifier to standard output.  
31462 The **-z** option is an existing NQS practice that has been standardized.
- 31463 Historically, the *qsub* utility has served the batch job-submission function in the NQS system, the  
31464 existing practice on which it is based. Some changes and additions have been made to the *qsub*  
31465 utility in this volume of IEEE Std. 1003.1-200x, *vis-a-vis* NQS, as a result of the growing pool of  
31466 experience with distributed batch systems.
- 31467 The set of features of the *qsub* utility as defined in this volume of IEEE Std. 1003.1-200x appears  
31468 to incorporate all the common existing practice on potentially POSIX-conformant platforms.  
31469 Where implementors wish to extend the functionality of their *qsub* utility, they may (as defined

31470 by IEEE Std. 1003.1-200x) use the **-W** option to provide implementation-defined extensions.

31471 **FUTURE DIRECTIONS**

31472 None.

31473 **SEE ALSO**

31474 *qrun, qstat, touch*, Chapter 3 (on page 2313)

31475 **CHANGE HISTORY**

31476 Derived from IEEE Std. 1003.2d-1994.

31477 **Issue 6**

31478 The **-I** option has been removed as there is no portable description of the resources that are  
31479 allowed or required by the batch job.

31480 **NAME**

31481 read — read a line from standard input

31482 **SYNOPSIS**

31483 read [-r] var...

31484 **DESCRIPTION**31485 The *read* utility shall read a single line from standard input.

31486 By default, unless the *-r* option is specified, backslash ('\`\`') shall act as an escape character, as  
 31487 described in Section 2.2.1 (on page 2236). If standard input is a terminal device and the invoking  
 31488 shell is interactive, *read* shall prompt for a continuation line when:

- 31489 • The shell reads an input line ending with a backslash, unless the *-r* option is specified.
- 31490 • A here-document is not terminated after a <newline> character is entered.

31491 The line shall be split into fields as in the shell (see Section 2.6.5 (on page 2249)); the first field  
 31492 shall be assigned to the first variable *var*, the second field to the second variable *var*, and so on. If  
 31493 there are fewer *var* operands specified than there are fields, the leftover fields and their  
 31494 intervening separators shall be assigned to the last *var*. If there are fewer fields than *vars*, the  
 31495 remaining *vars* shall be set to empty strings.

31496 The setting of variables specified by the *var* operands shall affect the current shell execution  
 31497 environment; see Section 2.13 (on page 2273). If it is called in a subshell or separate utility  
 31498 execution environment, such as one of the following:

```
31499 (read foo)
31500 nohup read ...
31501 find . -exec read ... \;
```

31502 it shall not affect the shell variables in the caller's environment.

31503 **OPTIONS**31504 The *read* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
31505 12.2, Utility Syntax Guidelines.

31506 The following option is supported:

31507 *-r* Do not treat a backslash character in any special way. Consider each backslash to  
 31508 be part of the input line.

31509 **OPERANDS**

31510 The following operand shall be supported:

31511 *var* The name of an existing or nonexisting shell variable.31512 **STDIN**

31513 The standard input shall be a text file.

31514 **INPUT FILES**

31515 None.

31516 **ENVIRONMENT VARIABLES**31517 The following environment variables shall affect the execution of *read*:

31518 *IFS* Determine the internal field separators used to delimit fields; see Section 2.5.3 (on  
 31519 page 2242).

31520 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 31521 If *LANG* is unset or null, the corresponding value from the implementation-  
 31522 defined default locale shall be used. If any of the internationalization variables

|       |                               |                                                                                                                        |
|-------|-------------------------------|------------------------------------------------------------------------------------------------------------------------|
| 31523 |                               | contains an invalid setting, the utility shall behave as if none of the variables had                                  |
| 31524 |                               | been defined.                                                                                                          |
| 31525 | <b>LC_ALL</b>                 | If set to a non-empty string value, override the values of all the other                                               |
| 31526 |                               | internationalization variables.                                                                                        |
| 31527 | <b>LC_CTYPE</b>               | Determine the locale for the interpretation of sequences of bytes of text data as                                      |
| 31528 |                               | characters (for example, single-byte as opposed to multi-byte characters in                                            |
| 31529 |                               | arguments).                                                                                                            |
| 31530 | <b>LC_MESSAGES</b>            |                                                                                                                        |
| 31531 |                               | Determine the locale that should be used to affect the format and contents of                                          |
| 31532 |                               | diagnostic messages written to standard error.                                                                         |
| 31533 | XSI <b>NLSPATH</b>            | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                  |
| 31534 | <b>PS2</b>                    | Provide the prompt string that an interactive shell shall write to standard error                                      |
| 31535 |                               | when a line ending with a backslash is read and the <b>-r</b> option was not specified, or                             |
| 31536 |                               | if a here-document is not terminated after a <newline> character is entered.                                           |
| 31537 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                        |
| 31538 |                               | Default.                                                                                                               |
| 31539 | <b>STDOUT</b>                 |                                                                                                                        |
| 31540 |                               | Not used.                                                                                                              |
| 31541 | <b>STDERR</b>                 |                                                                                                                        |
| 31542 |                               | Used for diagnostic messages and prompts for continued input.                                                          |
| 31543 | <b>OUTPUT FILES</b>           |                                                                                                                        |
| 31544 |                               | None.                                                                                                                  |
| 31545 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                        |
| 31546 |                               | None.                                                                                                                  |
| 31547 | <b>EXIT STATUS</b>            |                                                                                                                        |
| 31548 |                               | The following exit values shall be returned:                                                                           |
| 31549 | 0                             | Successful completion.                                                                                                 |
| 31550 | >0                            | End-of-file was detected or an error occurred.                                                                         |
| 31551 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                        |
| 31552 |                               | Default.                                                                                                               |
| 31553 | <b>APPLICATION USAGE</b>      |                                                                                                                        |
| 31554 |                               | The <i>read</i> utility has historically been a shell built-in.                                                        |
| 31555 |                               | The <b>-r</b> option is included to enable <i>read</i> to subsume the purpose of the <i>line</i> utility, which is not |
| 31556 |                               | included in IEEE Std. 1003.1-200x.                                                                                     |
| 31557 |                               | The results are undefined if an end-of-file is detected following a backslash at the end of a line                     |
| 31558 |                               | when <b>-r</b> is not specified.                                                                                       |
| 31559 | <b>EXAMPLES</b>               |                                                                                                                        |
| 31560 |                               | The following command:                                                                                                 |
| 31561 |                               | <code>while read -r xx yy</code>                                                                                       |
| 31562 |                               | <code>do</code>                                                                                                        |
| 31563 |                               | <code>    printf "%s %s\n" "\$yy" "\$xx"</code>                                                                        |
| 31564 |                               | <code>done &lt; input_file</code>                                                                                      |



31565            prints a file with the first field of each line moved to the end of the line.

31566 **RATIONALE**

31567            The *read* utility historically has been a shell built-in. It was separated off into its own utility to  
31568            take advantage of the richer description of functionality introduced by this volume of  
31569            IEEE Std. 1003.1-200x.

31570            Since *read* affects the current shell execution environment, it is generally provided as a shell  
31571            regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
31572            of the following:

```
31573 (read foo)
31574 nohup read ...
31575 find . -exec read ... \;
```

31576            it does not affect the shell variables in the environment of the caller.

31577 **FUTURE DIRECTIONS**

31578            None.

31579 **SEE ALSO**

31580            None.

31581 **CHANGE HISTORY**

31582            First released in Issue 2.

31583 **Issue 4**

31584            Relocated from the *sh* description for alignment with the ISO/IEC 9945-2: 1993 standard.

## 31585 NAME

31586 renice — set nice values of running processes

## 31587 SYNOPSIS

31588 UP `renice -n increment [-g | -p | -u] ID ...`

31589

## 31590 DESCRIPTION

31591 The *renice* utility shall request that the nice values (see the Base Definitions volume of  
31592 IEEE Std. 1003.1-200x, Section 3.241, Nice Value) of one or more running processes be changed.  
31593 By default, the applicable processes are specified by their process IDs. When a process group is  
31594 specified (see `-g`), the request applies to all processes in the process group.

31595 The nice value shall be bounded in an implementation-defined manner. If the requested  
31596 *increment* would raise or lower the nice value of the executed utility beyond implementation-  
31597 defined limits, then the limit whose value was exceeded shall be used.

31598 When a user is *reniced*, the request applies to all processes whose saved set-user-ID matches the  
31599 user ID corresponding to the user.

31600 Regardless of which options are supplied or any other factor, *renice* shall not alter the nice values  
31601 of any process unless the user requesting such a change has appropriate privileges to do so for  
31602 the specified process. If the user lacks appropriate privileges to perform the requested action, the  
31603 utility shall return an error status.

31604 The saved set-user-ID of the user's process shall be checked instead of its effective user ID when  
31605 *renice* attempts to determine the user ID of the process in order to determine whether the user  
31606 has appropriate privileges.

## 31607 OPTIONS

31608 The *renice* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
31609 12.2, Utility Syntax Guidelines.

31610 The following options shall be supported:

31611 `-g` Interpret all operands as unsigned decimal integer process group IDs.

31612 `-n increment` Specify how the nice value of the specified process or processes is to be adjusted.  
31613 The *increment* option-argument is a positive or negative decimal integer that shall  
31614 be used to modify the nice value of the specified process or processes.

31615 Positive *increment* values shall cause a lower nice value. Negative *increment* values  
31616 may require appropriate privileges and shall cause a higher nice value.

31617 `-p` Interpret all operands as unsigned decimal integer process IDs. The `-p` option is  
31618 the default if no options are specified.

31619 `-u` Interpret all operands as users. If a user exists with a user name equal to the  
31620 operand, then the user ID of that user is used in further processing. Otherwise, if  
31621 the operand represents an unsigned decimal integer, it shall be used as the numeric  
31622 user ID of the user.

## 31623 OPERANDS

31624 The following operands shall be supported:

31625 *ID* A process ID, process group ID, or user name/user ID, depending on the option  
31626 selected.

31627 **STDIN**

31628 Not used.

31629 **INPUT FILES**

31630 None.

31631 **ENVIRONMENT VARIABLES**31632 The following environment variables shall affect the execution of *renice*:

31633 *LANG* Provide a default value for the internationalization variables that are unset or null.  
31634 If *LANG* is unset or null, the corresponding value from the implementation-  
31635 defined default locale shall be used. If any of the internationalization variables  
31636 contains an invalid setting, the utility shall behave as if none of the variables had  
31637 been defined.

31638 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
31639 internationalization variables.

31640 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
31641 characters (for example, single-byte as opposed to multi-byte characters in  
31642 arguments).

31643 *LC\_MESSAGES*  
31644 Determine the locale that should be used to affect the format and contents of  
31645 diagnostic messages written to standard error.

31646 *XSI* *NLS\_PATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

31647 **ASYNCHRONOUS EVENTS**

31648 Default.

31649 **STDOUT**

31650 Not used.

31651 **STDERR**

31652 Used only for diagnostic messages.

31653 **OUTPUT FILES**

31654 None.

31655 **EXTENDED DESCRIPTION**

31656 None.

31657 **EXIT STATUS**

31658 The following exit values shall be returned:

31659 0 Successful completion.

31660 &gt;0 An error occurred.

31661 **CONSEQUENCES OF ERRORS**

31662 Default.

31663 **APPLICATION USAGE**

31664 None.

31665 **EXAMPLES**

31666 1. Adjust the nice value so that process IDs 987 and 32 would have a lower nice value:

31667 `renice -n 5 -p 987 32`31668 2. Adjust the nice value so that group IDs 324 and 76 would have a higher nice value, if the  
31669 user has the appropriate privileges to do so:31670 `renice -n -4 -g 324 76`31671 3. Adjust the nice value so that numeric user ID 8 and user **sas** would have a lower nice  
31672 value:31673 `renice -n 4 -u 8 sas`31674 Useful nice value increments on historical systems include 19 or 20 (the affected processes run  
31675 only when nothing else in the system attempts to run) and any negative number (to make  
31676 processes run faster).31677 **RATIONALE**31678 The *gid*, *pid*, and *user* specifications do not fit either the definition of operand or option-  
31679 argument. However, for clarity, they have been included in the OPTIONS section, rather than  
31680 the OPERANDS section.31681 The definition of nice value is not intended to suggest that all processes in a system have  
31682 priorities that are comparable. Scheduling policy extensions such as the realtime priorities in  
31683 POSIX.4 make the notion of a single underlying priority for all scheduling policies problematic.  
31684 Some systems may implement the *nice*-related features to affect all processes on the system,  
31685 others to affect just the general time-sharing activities implied by this volume of  
31686 IEEE Std. 1003.1-200x, and others may have no effect at all. Because of the use of  
31687 “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are  
31688 possible.31689 Originally, this utility was written in the historical manner, using the term “nice value”. This  
31690 was always a point of concern with users because it was never intuitively obvious what this  
31691 meant. With a newer version of *renice*, which used the term “system scheduling priority”, it was  
31692 hoped that novice users could better understand what this utility was meant to do. Also, it  
31693 would be easier to document what the utility was meant to do. Unfortunately, the addition of  
31694 the POSIX realtime scheduling capabilities introduced the concepts of process and thread  
31695 scheduling priorities that were totally unaffected by the *nice/renice* utilities or the  
31696 *nice()/setpriority()* functions. Continuing to use the term “system scheduling priority” would  
31697 have incorrectly suggested that these utilities and functions were indeed affecting these realtime  
31698 priorities. It was decided to revert to the historical term “nice value” to reference this unrelated  
31699 process attribute.31700 Although this utility has use by system administrators (and in fact appears in the system  
31701 administration portion of the BSD documentation), the standard developers considered that it  
31702 was very useful for individual end users to control their own processes.31703 **FUTURE DIRECTIONS**

31704 None.

31705 **SEE ALSO**31706 *nice*31707 **CHANGE HISTORY**

31708 First released in Issue 4.

31709 **Issue 5**31710 In the SYNOPSIS, an ellipsis is added to the `-u` option in all three obsolescent forms.31711 **Issue 6**

31712 This utility is now marked as part of the User Portability Utilities option.

31713 The APPLICATION USAGE section is added.

31714 The obsolescent forms of the SYNOPSIS are removed.

31715 Text previously conditional on `POSIX_SAVED_IDS` is mandatory in this issue. This is a FIPS  
31716 requirement.

## 31717 NAME

31718 rm — remove directory entries

## 31719 SYNOPSIS

31720 rm [-fiRr] *file*...

## 31721 DESCRIPTION

31722 The *rm* utility shall remove the directory entry specified by each *file* argument.

31723 If either of the files dot or dot-dot are specified as the basename portion of an operand (that is,  
31724 the final path name component), *rm* shall write a diagnostic message to standard error and do  
31725 nothing more with such operands.

31726 For each *file* the following steps shall be taken:

- 31727 1. If the *file* does not exist:
  - 31728 a. If the **-f** option is not specified, write a diagnostic message to standard error.
  - 31729 b. Go on to any remaining *files*.
- 31730 2. If *file* is of type directory, the following steps shall be taken:
  - 31731 a. If neither the **-R** option nor the **-r** option is specified, write a diagnostic message to  
31732 standard error, do nothing more with *file*, and go on to any remaining files.
  - 31733 b. If the **-f** option is not specified, and either the permissions of *file* do not permit  
31734 writing and the standard input is a terminal or the **-i** option is specified, write a  
31735 prompt to standard error and read a line from the standard input. If the response is  
31736 not affirmative, do nothing more with the current file and go on to any remaining  
31737 files.
  - 31738 c. For each entry contained in *file*, other than dot or dot-dot, the four steps listed here  
31739 (1-4) shall be taken with the entry as if it were a *file* operand. The *rm* utility shall not  
31740 traverse directories by following symbolic links into other parts of the hierarchy, but  
31741 shall remove the links themselves.
  - 31742 d. If the **-i** option is specified, write a prompt to standard error and read a line from the  
31743 standard input. If the response is not affirmative, do nothing more with the current  
31744 file, and go on to any remaining files.
- 31745 3. If *file* is not of type directory, the **-f** option is not specified, and either the permissions of  
31746 *file* do not permit writing and the standard input is a terminal or the **-i** option is specified,  
31747 write a prompt to the standard error and read a line from the standard input. If the  
31748 response is not affirmative, do nothing more with the current file and go on to any  
31749 remaining files.
- 31750 4. If the current file is a directory, *rm* shall perform actions equivalent to the *rmdir*()  
31751 function defined in the System Interfaces volume of IEEE Std. 1003.1-200x called with a path name  
31752 of the current file used as the *path* argument. If the current file is not a directory, *rm* shall  
31753 perform actions equivalent to the *unlink*() function defined in the System Interfaces  
31754 volume of IEEE Std. 1003.1-200x called with a path name of the current file used as the *path*  
31755 argument.

31756 If this fails for any reason, *rm* shall write a diagnostic message to standard error, do  
31757 nothing more with the current file, and go on to any remaining files.

31758 The *rm* utility shall be able to descend to arbitrary depths in a file hierarchy, and shall not fail  
31759 due to path length limitations (unless an operand specified by the user exceeds system  
31760 limitations).

31761 **OPTIONS**

31762 The *rm* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
31763 12.2, Utility Syntax Guidelines.

31764 The following options shall be supported:

31765 **-f** Do not prompt for confirmation. Do not write diagnostic messages or modify the  
31766 exit status in the case of nonexistent operands. Any previous occurrences of the **-i**  
31767 option shall be ignored.

31768 **-i** Prompt for confirmation as described previously. Any previous occurrences of the  
31769 **-f** option shall be ignored.

31770 **-R** Remove file hierarchies. See the DESCRIPTION.

31771 **-r** Equivalent to **-R**.

31772 **OPERANDS**

31773 The following operand shall be supported:

31774 *file* A path name of a directory entry to be removed.

31775 **STDIN**

31776 Used to read an input line in response to each prompt specified in the STDOUT section.  
31777 Otherwise, the standard input shall not be used.

31778 **INPUT FILES**

31779 None.

31780 **ENVIRONMENT VARIABLES**

31781 The following environment variables shall affect the execution of *rm*:

31782 *LANG* Provide a default value for the internationalization variables that are unset or null.  
31783 If *LANG* is unset or null, the corresponding value from the implementation-  
31784 defined default locale shall be used. If any of the internationalization variables  
31785 contains an invalid setting, the utility shall behave as if none of the variables had  
31786 been defined.

31787 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
31788 internationalization variables.

31789 *LC\_COLLATE*

31790 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
31791 character collating elements used in the extended regular expression defined for  
31792 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

31793 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
31794 characters (for example, single-byte as opposed to multi-byte characters in  
31795 arguments) and the behavior of character classes within regular expressions used  
31796 in the extended regular expression defined for the **yesexpr** locale keyword in the  
31797 *LC\_MESSAGES* category.

31798 *LC\_MESSAGES*

31799 Determine the locale for the processing of affirmative responses that should be  
31800 used to affect the format and contents of diagnostic messages written to standard  
31801 error.

31802 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

31803 **ASYNCHRONOUS EVENTS**

31804 Default.

31805 **STDOUT**

31806 Not used.

31807 **STDERR**

31808 Prompts shall be written to standard error under the conditions specified in the DESCRIPTION  
31809 and OPTIONS sections. The prompts shall contain the *file* path name, but their format is  
31810 otherwise unspecified. The standard error also shall be used for diagnostic messages.

31811 **OUTPUT FILES**

31812 None.

31813 **EXTENDED DESCRIPTION**

31814 None.

31815 **EXIT STATUS**

31816 The following exit values shall be returned:

31817 0 All of the named directory entries for which *rm* performed actions equivalent to *rmdir()* or  
31818 *unlink()* functions were removed.

31819 &gt;0 An error occurred.

31820 **CONSEQUENCES OF ERRORS**

31821 Default.

31822 **APPLICATION USAGE**

31823 The *rm* utility is forbidden to remove the names dot and dot-dot in order to avoid the  
31824 consequences of inadvertently doing something like:

31825 `rm -r .*`

31826 Some systems do not permit the removal of the last link to an executable binary file that is being  
31827 executed; see the [EBUSY] error in the *unlink()* function defined in the System Interfaces volume  
31828 of IEEE Std. 1003.1-200x. Thus, the *rm* utility can fail to remove such files.

31829 The *-i* option causes *rm* to prompt and read the standard input even if the standard input is not  
31830 a terminal, but in the absence of *-i* the mode prompting is not done when the standard input is  
31831 not a terminal.

31832 **EXAMPLES**

31833 1. The following command:

31834 `rm a.out core`31835 removes the directory entries: **a.out** and **core**.

31836 2. The following command:

31837 `rm -Rf junk`31838 removes the directory **junk** and all its contents, without prompting.31839 **RATIONALE**

31840 The *-i* option causes *rm* to prompt and read the standard input even if the standard input is not  
31841 a terminal, but, in the absence of *-i*, the mode prompting is not done when the standard input is  
31842 not a terminal.

31843 For absolute clarity, paragraphs (2b) and (3) in the DESCRIPTION of *rm* describing the behavior  
31844 when prompting for confirmation, should be interpreted in the following manner:



```
31845 if ((NOT f_option) AND
31846 ((not_writable AND input_is_terminal) OR i_option))
```

31847 The exact format of the interactive prompts is unspecified. Only the general nature of the  
31848 contents of prompts are specified because implementations may desire more descriptive  
31849 prompts than those used on historical implementations. Therefore, an application not using the  
31850 `-f` option, or using the `-i` option, relies on the system to provide the most suitable dialog directly  
31851 with the user, based on the behavior specified.

31852 The `-r` option is historical practice on all known systems. The synonym `-R` option is provided  
31853 for consistency with the other utilities in this volume of IEEE Std. 1003.1-200x that provide  
31854 options requesting recursive descent through the file hierarchy.

31855 The behavior of the `-f` option in historical versions of *rm* is inconsistent. In general, along with  
31856 “forcing” the unlink without prompting for permission, it always causes diagnostic messages to  
31857 be suppressed and the exit status to be unmodified for nonexistent operands and files that  
31858 cannot be unlinked. In some versions, however, the `-f` option suppresses usage messages and  
31859 system errors as well. Suppressing such messages is not a service to either shell scripts or users.

31860 It is less clear that error messages regarding files that cannot be unlinked (removed) should be  
31861 suppressed. Although this is historical practice, this volume of IEEE Std. 1003.1-200x does not  
31862 permit the `-f` option to suppress such messages.

31863 When given the `-r` and `-i` options, historical versions of *rm* prompt the user twice for each  
31864 directory, once before removing its contents and once before actually attempting to delete the  
31865 directory entry that names it. This allows the user to “prune” the file hierarchy walk. Historical  
31866 versions of *rm* were inconsistent in that some did not do the former prompt for directories  
31867 named on the command line and others had obscure prompting behavior when the `-i` option  
31868 was specified and the permissions of the file did not permit writing. The POSIX Shell and  
31869 Utilities *rm* differs little from historic practice, but does require that prompts be consistent.  
31870 Historical versions of *rm* were also inconsistent in that prompts were done to both standard  
31871 output and standard error. This volume of IEEE Std. 1003.1-200x requires that prompts be done  
31872 to standard error, for consistency with *cp* and *mv*, and to allow historical extensions to *rm* that  
31873 provide an option to list deleted files on standard output.

31874 The *rm* utility is required to descend to arbitrary depths so that any file hierarchy may be  
31875 deleted. This means, for example, that the *rm* utility cannot run out of file descriptors during its  
31876 descent (that is, if the number of file descriptors is limited, *rm* cannot be implemented in the  
31877 historical fashion where one file descriptor is used per directory level). Also, *rm* is not permitted  
31878 to fail because of path length restrictions, unless an operand specified by the user is longer than  
31879 `{PATH_MAX}`.

31880 The *rm* utility removes symbolic links themselves, not the files they refer to, as a consequence of  
31881 the dependence on the `unlink()` functionality, per the DESCRIPTION. When removing  
31882 hierarchies with `-r` or `-R`, the prohibition on following symbolic links has to be made explicit.

#### 31883 FUTURE DIRECTIONS

31884 None.

#### 31885 SEE ALSO

31886 *rmdir*, the System Interfaces volume of IEEE Std. 1003.1-200x, `remove()`, `unlink()`

#### 31887 CHANGE HISTORY

31888 First released in Issue 2.

31889 **Issue 4**

31890 Aligned with the ISO/IEC 9945-2: 1993 standard.

31891 **Issue 5**

31892 FUTURE DIRECTIONS section added.

31893 **Issue 6**

31894 Text is added to clarify actions relating to symbolic links as specified in the IEEE P1003.2b draft  
31895 standard.

31896 **NAME**31897 rmdel — remove a delta from an SCCS file (**DEVELOPMENT**)31898 **SYNOPSIS**31899 XSI `rmdel -r SID file...`

31900

31901 **DESCRIPTION**

31902 The *rmdel* utility shall remove the delta specified by the SID from each named SCCS file. The  
 31903 delta to be removed shall be the most recent delta in its branch in the delta chain of each named  
 31904 SCCS file. In addition, the application shall ensure that the SID specified is not that of a version  
 31905 being edited for the purpose of making a delta; that is, if a *p-file* (see *get* (on page 2685)) exists for  
 31906 the named SCCS file, the SID specified shall not appear in any entry of the *p-file*.

31907 Removal of a delta shall be restricted to:

- 31908 1. The user who made the delta
- 31909 2. The owner of the SCCS file
- 31910 3. The owner of the directory containing the SCCS file

31911 **OPTIONS**

31912 The *rmdel* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 31913 12.2, Utility Syntax Guidelines.

31914 The following option shall be supported:

31915 **-r** *SID* Specify the SCCS identification string (*SID*) of the delta to be deleted.31916 **OPERANDS**

31917 The following operand shall be supported:

31918 *file* A path name of an existing SCCS file or a directory. If *file* is a directory, the *rmdel*  
 31919 utility shall behave as though each file in the directory were specified as a named  
 31920 file, except that non-SCCS files (last component of the path name does not begin  
 31921 with *s.*) and unreadable files shall be silently ignored.

31922 If exactly one *file* operand appears, and it is *'-'*, the standard input shall be read;  
 31923 each line of the standard input is taken to be the name of an SCCS file to be  
 31924 processed. Non-SCCS files and unreadable files shall be silently ignored.

31925 **STDIN**

31926 The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
 31927 line of the text file shall be interpreted as an SCCS path name.

31928 **INPUT FILES**

31929 The SCCS files are files of unspecified format.

31930 **ENVIRONMENT VARIABLES**31931 The following environment variables shall affect the execution of *rmdel*:

31932 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 31933 If *LANG* is unset or null, the corresponding value from the implementation-  
 31934 defined default locale shall be used. If any of the internationalization variables  
 31935 contains an invalid setting, the utility shall behave as if none of the variables had  
 31936 been defined.

31937 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 31938 internationalization variables.

- 31939        *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
31940 characters (for example, single-byte as opposed to multi-byte characters in  
31941 arguments and input files).
- 31942        *LC\_MESSAGES*  
31943                Determine the locale that should be used to affect the format and contents of  
31944 diagnostic messages written to standard error.
- 31945        *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 31946 **ASYNCHRONOUS EVENTS**  
31947        Default.
- 31948 **STDOUT**  
31949        Not used.
- 31950 **STDERR**  
31951        Used only for diagnostic messages.
- 31952 **OUTPUT FILES**  
31953        The SCCS files are files of unspecified format. During processing of a *file*, a temporary *x-file*, as  
31954 described in *admin* (on page 2340), may be created and deleted; a locking *z-file*, as described in  
31955 *get* (on page 2685), may be created and deleted.
- 31956 **EXTENDED DESCRIPTION**  
31957        None.
- 31958 **EXIT STATUS**  
31959        The following exit values shall be returned:  
31960        0 Successful completion.  
31961        >0 An error occurred.
- 31962 **CONSEQUENCES OF ERRORS**  
31963        Default.
- 31964 **APPLICATION USAGE**  
31965        None.
- 31966 **EXAMPLES**  
31967        None.
- 31968 **RATIONALE**  
31969        None.
- 31970 **FUTURE DIRECTIONS**  
31971        None.
- 31972 **SEE ALSO**  
31973        *delta*, *get*, *prs*
- 31974 **CHANGE HISTORY**  
31975        First released in Issue 2.
- 31976 **Issue 4**  
31977        Format reorganized.  
31978        Utility Syntax Guidelines support mandated.  
31979        Internationalized environment variable support mandated.

31980 **Issue 6**

31981 The normative text is reworded to avoid use of the term “must” for application requirements. |

31982 The normative text is reworded to emphasise the term “shall” for implementation requirements. |

31983 **NAME**

31984            rmdir — remove directories

31985 **SYNOPSIS**31986            rmdir [-p] *dir...*31987 **DESCRIPTION**31988            The *rmdir* utility shall remove the directory entry specified by each *dir* operand, which, in order  
31989            to succeed, the application shall ensure refers to an empty directory.31990            Directories shall be processed in the order specified. If a directory and a subdirectory of that  
31991            directory are specified in a single invocation of the *rmdir* utility, the application shall specify the  
31992            subdirectory before the parent directory so that the parent directory will be empty when the  
31993            *rmdir* utility tries to remove it.31994 **OPTIONS**31995            The *rmdir* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
31996            12.2, Utility Syntax Guidelines.

31997            The following option shall be supported:

31998            **-p**            Remove all directories in a path name. For each *dir* operand:

- 31999                            1. The directory entry it names shall be removed.
- 
- 32000                            2. If the
- dir*
- operand includes more than one path name component, effects
- 
- 32001                            equivalent to the following command shall occur:

32002                            rmdir -p \$(dirname *dir*)32003 **OPERANDS**

32004            The following operand shall be supported:

32005            *dir*            A path name of an empty directory to be removed.32006 **STDIN**

32007            Not used.

32008 **INPUT FILES**

32009            None.

32010 **ENVIRONMENT VARIABLES**32011            The following environment variables shall affect the execution of *rmdir*:32012            **LANG**            Provide a default value for the internationalization variables that are unset or null.  
32013                            If *LANG* is unset or null, the corresponding value from the implementation-  
32014                            defined default locale shall be used. If any of the internationalization variables  
32015                            contains an invalid setting, the utility shall behave as if none of the variables had  
32016                            been defined.32017            **LC\_ALL**            If set to a non-empty string value, override the values of all the other  
32018                            internationalization variables.32019            **LC\_CTYPE**        Determine the locale for the interpretation of sequences of bytes of text data as  
32020                            characters (for example, single-byte as opposed to multi-byte characters in  
32021                            arguments).32022            **LC\_MESSAGES**32023                            Determine the locale that should be used to affect the format and contents of  
32024                            diagnostic messages written to standard error.

- 32025 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 32026 **ASYNCHRONOUS EVENTS**
- 32027 Default.
- 32028 **STDOUT**
- 32029 Not used.
- 32030 **STDERR**
- 32031 Used only for diagnostic messages.
- 32032 **OUTPUT FILES**
- 32033 None.
- 32034 **EXTENDED DESCRIPTION**
- 32035 None.
- 32036 **EXIT STATUS**
- 32037 The following exit values shall be returned:
- 32038 0 Each directory entry specified by a *dir* operand was removed successfully.
- 32039 >0 An error occurred.
- 32040 **CONSEQUENCES OF ERRORS**
- 32041 Default.
- 32042 **APPLICATION USAGE**
- 32043 The definition of an empty directory is one that contains, at most, directory entries for dot and dot-dot.
- 32044
- 32045 **EXAMPLES**
- 32046 If a directory **a** in the current directory is empty except it contains a directory **b** and **a/b** is empty
- 32047 except it contains a directory **c**:
- 32048 `rmdir -p a/b/c`
- 32049 removes all three directories.
- 32050 **RATIONALE**
- 32051 On historical System V systems, the `-p` option also caused a message to be written to the
- 32052 standard output. The message indicated whether the whole path was removed or whether part
- 32053 of the path remained for some reason. The **STDERR** section requires this diagnostic when the
- 32054 entire path specified by a *dir* operand is not removed, but does not allow the status message
- 32055 reporting success to be written as a diagnostic.
- 32056 The *rmdir* utility on System V also included an `-s` option that suppressed the informational
- 32057 message output by the `-p` option. This option has been omitted because the informational
- 32058 message is not specified by this volume of IEEE Std. 1003.1-200x.
- 32059 **FUTURE DIRECTIONS**
- 32060 None.
- 32061 **SEE ALSO**
- 32062 *rm*, the System Interfaces volume of IEEE Std. 1003.1-200x, *remove()*, *rmdir()*, *unlink()*
- 32063 **CHANGE HISTORY**
- 32064 First released in Issue 2.

32065 **Issue 4**

32066

Separated from the *rm* description and aligned with the ISO/IEC 9945-2: 1993 standard.

32067 **Issue 6**

32068

The normative text is reworded to avoid use of the term “must” for application requirements.



32069 **NAME**32070 `sact` — print current SCCS file-editing activity (**DEVELOPMENT**)32071 **SYNOPSIS**32072 XSI `sact file...`

32073

32074 **DESCRIPTION**

32075 The *sact* utility shall inform the user of any impending deltas to a named SCCS file by writing a  
 32076 list to standard output. This situation occurs when *get -e* has been executed previously without  
 32077 a subsequent execution of *delta*.

32078 **OPTIONS**

32079 None.

32080 **OPERANDS**

32081 The following operand shall be supported:

32082 *file* A path name of an existing SCCS file or a directory. If *file* is a directory, the *sact*  
 32083 utility shall behave as though each file in the directory were specified as a named  
 32084 file, except that non-SCCS files (last component of the path name does not begin  
 32085 with *s*.) and unreadable files shall be silently ignored.

32086 If a single instance *file* is specified as *'-'*, the standard input shall be read; each  
 32087 line of the standard input shall be taken to be the name of an SCCS file to be  
 32088 processed. Non-SCCS files and unreadable files shall be silently ignored.

32089 **STDIN**

32090 The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
 32091 line of the text file shall be interpreted as an SCCS path name.

32092 **INPUT FILES**

32093 Any SCCS files interrogated are files of an unspecified format.

32094 **ENVIRONMENT VARIABLES**32095 The following environment variables shall affect the execution of *sact*:

32096 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 32097 If *LANG* is unset or null, the corresponding value from the implementation-  
 32098 defined default locale shall be used. If any of the internationalization variables  
 32099 contains an invalid setting, the utility shall behave as if none of the variables had  
 32100 been defined.

32101 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 32102 internationalization variables.

32103 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 32104 characters (for example, single-byte as opposed to multi-byte characters in  
 32105 arguments and input files).

32106 *LC\_MESSAGES*

32107 Determine the locale that should be used to affect the format and contents of  
 32108 diagnostic messages written to standard error.

32109 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

32110 **ASYNCHRONOUS EVENTS**

32111 Default.

32112 **STDOUT**

32113 The output for each named file shall consist of a line in the following format:

32114 "%sΔ%sΔ%sΔ%sΔ%s\n", &lt;SID&gt;, &lt;new SID&gt;, &lt;login&gt;, &lt;date&gt;, &lt;time&gt;

32115 <SID> Specifies the SID of a delta that currently exists in the SCCS file to which changes  
32116 are made to make the new delta.

32117 &lt;new SID&gt; Specifies the SID for the new delta to be created.

32118 <login> Contains the login name of the user who makes the delta (that is, who executed a  
32119 *get* for editing).32120 <date> Contains the date that *get -e* was executed, in the format used by the *prs :D:* data  
32121 keyword.32122 <time> Contains the time that *get -e* was executed, in the format used by the *prs :T:* data  
32123 keyword.32124 If there is more than one named file or if a directory or standard input is named, each path name  
32125 shall be written before each of the preceding lines:

32126 "\n%s:\n", &lt;pathname&gt;

32127 **STDERR**32128 Used only for optional informative messages concerning SCCS files with no impending deltas,  
32129 and for diagnostic messages.32130 **OUTPUT FILES**

32131 None.

32132 **EXTENDED DESCRIPTION**

32133 None.

32134 **EXIT STATUS**

32135 The following exit values shall be returned:

32136 0 Successful completion.

32137 &gt;0 An error occurred.

32138 **CONSEQUENCES OF ERRORS**

32139 Default.

32140 **APPLICATION USAGE**

32141 None.

32142 **EXAMPLES**

32143 None.

32144 **RATIONALE**

32145 None.

32146 **FUTURE DIRECTIONS**

32147 None.

32148 **SEE ALSO**32149 *delta, get, unget*32150 **CHANGE HISTORY**

32151 First released in Issue 2.

32152 **Issue 4**

32153 Format reorganized.

32154 Utility Syntax Guidelines support mandated.

32155 Internationalized environment variable support mandated.

32156 **Issue 4, Version 2**32157 The `STDERR` section encompasses informative messages concerning SCCS files with no  
32158 impending deltas.32159 **Issue 6**

32160 The normative text is reworded to emphasise the term “shall” for implementation requirements.

## 32161 NAME

32162 `sccs` — front end for the SCCS subsystem (**DEVELOPMENT**)

## 32163 SYNOPSIS

32164 XSI `sccs [-r][-d path][-p path] command [options...][operands...]`

32165

## 32166 DESCRIPTION

32167 The `sccs` utility is a front end to the SCCS programs. It also includes the capability to run set-  
32168 user-id to another user to provide additional protection.32169 The `sccs` utility shall invoke the specified *command* with the specified *options* and *operands*. By  
32170 default, each of the *operands* shall be modified by prefixing it with the string **SCCS/s**.32171 The *command* can be the name of one of the SCCS utilities in this volume of IEEE Std. 1003.1-200x  
32172 (*admin*, *delta*, *get*, *prs*, *rmdel*, *sact*, *unget*, *val*, or *what*) or one of the pseudo-utilities listed in the  
32173 EXTENDED DESCRIPTION section.

## 32174 OPTIONS

32175 The `sccs` utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
32176 12.2, Utility Syntax Guidelines, except that *options* operands are actually options to be passed to  
32177 the utility named by *command*. When the portion of the command:32178 `command [options ... ] [operands ... ]`32179 is considered, all of the pseudo-utilities used as *command* shall support the Utility Syntax  
32180 Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the  
32181 Guidelines to the extent indicated by their individual OPTIONS sections.32182 The following options shall be supported preceding the *command* operand:32183 **-d path** A path name of a directory to be used as a root directory for the SCCS files. The  
32184 default is the current directory. The **-d** option takes precedence over the  
32185 **PROJECTDIR** variable. See **-p**.32186 **-p path** A path name of a directory in which the SCCS files are located. The default is the  
32187 **SCCS** directory.32188 The **-p** option differs from the **-d** option in that the **-d** option-argument is  
32189 prefixed to the entire path name and the **-p** option-argument is inserted before the  
32190 final component of the path name. For example:32191 `sccs -d /x -p y get a/b`

32192 converts to:

32193 `get /x/a/y/s.b`

32194 This allows the creation of aliases such as:

32195 `alias syssccs="sccs -d /usr/src"`

32196 which is used as:

32197 `syssccs get cmd/who.c`32198 **-r** Invoke *command* with the real user ID of the process, not any effective user ID that  
32199 the `sccs` utility is set to. Certain commands (*admin*, **check**, **clean**, **diffs**, **info**, *rmdel*,  
32200 and **tell**) cannot be run set-user-ID by all users, since this would allow anyone to  
32201 change the authorizations. These commands are always run as the real user.

32202 **OPERANDS**

32203 The following operands shall be supported:

32204 *command* An SCCS utility name or the name of one of the pseudo-utilities listed in the  
32205 EXTENDED DESCRIPTION section.

32206 *options* An option or option-argument to be passed to *command*.

32207 *operands* An operand to be passed to *command*.

32208 **STDIN**

32209 See the utility description for the specified *command*.

32210 **INPUT FILES**

32211 See the utility description for the specified *command*.

32212 **ENVIRONMENT VARIABLES**

32213 The following environment variables shall affect the execution of *sccs*:

32214 *LANG* Provide a default value for the internationalization variables that are unset or null.  
32215 If *LANG* is unset or null, the corresponding value from the implementation-  
32216 defined default locale shall be used. If any of the internationalization variables  
32217 contains an invalid setting, the utility shall behave as if none of the variables had  
32218 been defined.

32219 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
32220 internationalization variables.

32221 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
32222 characters (for example, single-byte as opposed to multi-byte characters in  
32223 arguments and input files).

32224 *LC\_MESSAGES*

32225 Determine the locale that should be used to affect the format and contents of  
32226 diagnostic messages written to standard error.

32227 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

32228 *PROJECTDIR*

32229 Provide a default value for the *-d path* option. If the value of *PROJECTDIR* begins  
32230 with a slash, it shall be considered an absolute path name; otherwise, the value of  
32231 *PROJECTDIR* is treated as a user name and that user's initial working directory  
32232 shall be examined for a subdirectory *src* or *source*. If such a directory is found, it  
32233 shall be used. Otherwise, the value shall be used as a relative path name.

32234 Additional environment variable effects may be found in the utility description for the specified  
32235 *command*.

32236 **ASYNCHRONOUS EVENTS**

32237 Default.

32238 **STDOUT**

32239 See the utility description for the specified *command*.

32240 **STDERR**

32241 See the utility description for the specified *command*.

32242 **OUTPUT FILES**32243 See the utility description for the specified *command*.32244 **EXTENDED DESCRIPTION**32245 The following pseudo-utilities shall be supported as *command* operands. All options referred to  
32246 in the following list are values given in the *options* operands following *command*.32247 **check** Equivalent to **info**, except that nothing shall be printed if nothing is being edited, and a  
32248 non-zero exit status shall be returned if anything is being edited. The intent is to have  
32249 this included in an “install” entry in a makefile to ensure that everything is included  
32250 into the SCCS file before a version is installed.32251 **clean** Remove everything from the current directory that can be recreated from SCCS files,  
32252 but do not remove any files being edited. If the **-b** option is given, branches shall be  
32253 ignored in the determination of whether they are being edited; this is dangerous if  
32254 branches are kept in the same directory.32255 **create** Create an SCCS file, taking the initial contents from the file of the same name. Any  
32256 options to *admin* are accepted. If the creation is successful, the original files shall be  
32257 renamed by prefixing the basenames with a comma. These renamed files should be  
32258 removed after it has been verified that the SCCS files have been created successfully.32259 **delget** Perform a *delta* on the named files and then *get* new versions. The new versions shall  
32260 have ID keywords expanded and shall not be editable. Any **-m**, **-p**, **-r**, **-s**, and **-y**  
32261 options shall be passed to *delta*, and any **-b**, **-c**, **-e**, **-i**, **-k**, **-l**, **-s**, and **-x** options shall be  
32262 passed to *get*.32263 **deledit** Equivalent to **delget**, except that the *get* phase shall include the **-e** option. This option  
32264 is useful for making a checkpoint of the current editing phase. The same options are  
32265 passed to *delta* as described above, and all the options listed for *get* above except **-e** are  
32266 passed to **edit**.32267 **diffs** Write a difference listing between the current version of the files checked out for  
32268 editing and the versions in SCCS format. Any **-r**, **-c**, **-i**, **-x**, and **-t** options shall be  
32269 passed to *get*; any **-l**, **-s**, **-e**, **-f**, **-h**, and **-b** options shall be passed to *diff*. A **-C** option  
32270 shall be passed to *diff* as **-c**.32271 **edit** Equivalent to *get -e*.32272 **fix** Remove the named delta, but leave a copy of the delta with the changes that were in it.  
32273 It is useful for fixing small compiler bugs, and so on. The application shall ensure that it  
32274 is followed by a **-r** *SID* option. Since **fix** doesn't leave audit trails, it should be used  
32275 carefully.32276 **info** Write a listing of all files being edited. If the **-b** option is given, branches (that is, SIDs  
32277 with two or fewer components) shall be ignored. If a **-u** *user* option is given, then only  
32278 files being edited by the named user shall be listed. A **-U** option shall be equivalent to  
32279 **-u**<*current user*>.32280 **print** Write out verbose information about the named files, equivalent to *sccs prs*.32281 **tell** Write a <newline>-separated list of the files being edited to standard output. Takes the  
32282 **-b**, **-u**, and **-U** options like **info** and **check**.32283 **unedit** This is the opposite of an **edit** or a *get -e*. It should be used with caution, since any  
32284 changes made since the *get* are lost.

32285 **EXIT STATUS**

32286 The following exit values shall be returned:

32287 0 Successful completion.

32288 &gt;0 An error occurred.

32289 **CONSEQUENCES OF ERRORS**

32290 Default.

32291 **APPLICATION USAGE**

32292 Many of the SCCS utilities take directory names as operands as well as specific file names. The  
 32293 pseudo-utilities supported by `sccs` are not described as having this capability, but are not  
 32294 prohibited from doing so.

32295 **EXAMPLES**

32296 1. To get a file for editing, edit it and produce a new delta:

32297 `sccs get -e file.c`32298 `ex file.c`32299 `sccs delta file.c`

32300 2. To get a file from another directory:

32301 `sccs -p /usr/src/sccs/s. get cc.c`

32302 or:

32303 `sccs get /usr/src/sccs/s.cc.c`

32304 3. To make a delta of a large number of files in the current directory:

32305 `sccs delta *.c`

32306 4. To get a list of files being edited that are not on branches:

32307 `sccs info -b`

32308 5. To delta everything being edited by the current user:

32309 `sccs delta $(sccs tell -U)`

32310 6. In a makefile, to get source files from an SCCS file if it does not already exist:

32311 `SRCS = <list of source files>`32312 `$(SRCS):`32313 `sccs get $(REL) $@`32314 **RATIONALE**

32315 SCCS and its associated utilities are part of the XSI Development Utilities option within the XSI  
 32316 extension.

32317 SCCS is an abbreviation for Source Code Control System. It is a maintenance and enhancement  
 32318 tracking tool. When a file is put under SCCS, the source code control system maintains the file  
 32319 and, when changes are made, identifies and stores them in the file with the original source code  
 32320 and/or documentation. As other changes are made, they too are identified and retained in the  
 32321 file.

32322 Retrieval of the original and any set of changes is possible. Any version of the file as it develops  
 32323 can be reconstructed for inspection or additional modification. History data can be stored with  
 32324 each version, documenting why the changes were made, who made them, and when they were  
 32325 made.

32326 **FUTURE DIRECTIONS**

32327 None.

32328 **SEE ALSO**32329 *admin, delta, get, make, prs, rmdel, sact, unget, val, what*32330 **CHANGE HISTORY**

32331 First released in Issue 4.

32332 **Issue 6**

32333 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from  
32334 “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of  
32335 *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

32336 The normative text is reworded to avoid use of the term “must” for application requirements. |

32337 The normative text is reworded to emphasise the term “shall” for implementation requirements. |



32338 **NAME**32339 `sed` — stream editor32340 **SYNOPSIS**32341 `sed [-n] script[file...]`32342 `sed [-n][-e script][...][-f script_file][...][file...]`32343 **DESCRIPTION**

32344 The *sed* utility is a stream editor that shall read one or more text files, make editing changes  
 32345 according to a script of editing commands, and write the results to standard output. The script  
 32346 shall be obtained from either the *script* operand string or a combination of the option-arguments  
 32347 from the *-e script* and *-f script\_file* options.

32348 **OPTIONS**

32349 The *sed* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 32350 12.2, Utility Syntax Guidelines, except that the order of presentation of the *-e* and *-f* options is  
 32351 significant.

32352 The following options shall be supported:

32353 *-e script* Add the editing commands specified by the *script* option-argument to the end of  
 32354 the script of editing commands. The *script* option-argument shall have the same  
 32355 properties as the *script* operand, described in the OPERANDS section.

32356 *-f script\_file* Add the editing commands in the file *script\_file* to the end of the script.

32357 *-n* Suppress the default output (in which each line, after it is examined for editing, is  
 32358 written to standard output). Only lines explicitly selected for output are written.

32359 Multiple *-e* and *-f* options may be specified. All commands shall be added to the script in the  
 32360 order specified, regardless of their origin.

32361 **OPERANDS**

32362 The following operands shall be supported:

32363 *file* A path name of a file whose contents are read and edited. If multiple *file* operands  
 32364 are specified, the named files shall be read in the order specified and the  
 32365 concatenation shall be edited. If no *file* operands are specified, the standard input  
 32366 shall be used.

32367 *script* A string to be used as the script of editing commands. The application shall not  
 32368 present a *script* that violates the restrictions of a text file except that the final  
 32369 character need not be a <newline> character.

32370 **STDIN**

32371 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES  
 32372 section.

32373 **INPUT FILES**

32374 The input files shall be text files. The *script\_files* named by the *-f* option shall consist of editing  
 32375 commands.

32376 **ENVIRONMENT VARIABLES**

32377 The following environment variables shall affect the execution of *sed*:

32378 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 32379 If *LANG* is unset or null, the corresponding value from the implementation-  
 32380 defined default locale shall be used. If any of the internationalization variables  
 32381 contains an invalid setting, the utility shall behave as if none of the variables had

- 32382                    been defined.
- 32383            *LC\_ALL*    If set to a non-empty string value, override the values of all the other  
32384                    internationalization variables.
- 32385            *LC\_COLLATE*  
32386                    Determine the locale for the behavior of ranges, equivalence classes, and multi-  
32387                    character collating elements within regular expressions.
- 32388            *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
32389                    characters (for example, single-byte as opposed to multi-byte characters in  
32390                    arguments and input files), and the behavior of character classes within regular  
32391                    expressions.
- 32392            *LC\_MESSAGES*  
32393                    Determine the locale that should be used to affect the format and contents of  
32394                    diagnostic messages written to standard error.
- 32395 XSI            *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 32396 **ASYNCHRONOUS EVENTS**  
32397                    Default.
- 32398 **STDOUT**  
32399                    The input files shall be written to standard output, with the editing commands specified in the  
32400                    script applied. If the *-n* option is specified, only those input lines selected by the script shall be  
32401                    written to standard output.
- 32402 **STDERR**  
32403                    Used only for diagnostic messages.
- 32404 **OUTPUT FILES**  
32405                    The output files shall be text files whose formats are dependent on the editing commands given.
- 32406 **EXTENDED DESCRIPTION**  
32407                    The *script* shall consist of editing commands of the following form:  
32408                    [ *address* [ , *address* ] ] *function*
- 32409                    where *function* represents a single-character command verb from the list in **Editing Commands**  
32410                    in **sed** (on page 3053), followed by any applicable arguments.
- 32411                    Zero or more <blank> characters shall be accepted before the first address and before function.  
32412                    Any number of semicolons shall be accepted before the first address.
- 32413                    In default operation, *sed* cyclically shall copy a line of input, less its terminating <newline>, into  
32414                    a pattern space (unless there is something left after a **D** command), apply in sequence all  
32415                    commands whose addresses select that pattern space, and at the end of the script copy the  
32416                    pattern space to standard output (except when *-n* is specified) and delete the pattern space.  
32417                    Whenever the pattern space is written to standard output or a named file, *sed* shall immediately  
32418                    follow it with a <newline>.
- 32419                    Some of the editing commands use a hold space to save all or part of the pattern space for  
32420                    subsequent retrieval. The pattern and hold spaces shall each be able to hold at least 8 192 bytes.

32421 **Addresses in sed**

32422 An address is either a decimal number that counts input lines cumulatively across files, a '\$' character that addresses the last line of input, or a context address (which consists of a BRE, as described in **Regular Expressions in sed**, preceded and followed by a delimiter, usually a slash).

32425 An editing command with no addresses shall select every pattern space.

32426 An editing command with one address shall select each pattern space that matches the address.

32427 An editing command with two addresses shall select the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line shall be selected.) Starting at the first line following the selected range, *sed* shall look again for the first address. Thereafter, the process shall be repeated. Omitting either or both of the address components in the following form produces undefined results:

32433 [ *address* [ , *address* ] ]

32434 **Regular Expressions in sed**

32435 The *sed* utility shall support the BREs described in the Base Definitions volume of IEEE Std. 1003.1-200x, Section 9.3, Basic Regular Expressions, with the following additions:

- 32437 • In a context address, the construction "`\cBREc`", where *c* is any character other than  
32438 backslash or <newline>, shall be identical to "`/BRE/`". If the character designated by *c*  
32439 appears following a backslash, then it shall be considered to be that literal character, which  
32440 shall not terminate the BRE. For example, in the context address "`\xabc\xdefx`", the  
32441 second *x* stands for itself, so that the BRE is "`abcxdef`".
- 32442 • The escape sequence '`\n`' shall match a <newline> embedded in the pattern space. A literal  
32443 <newline> character shall not be used in the BRE of a context address or in the substitute  
32444 function.
- 32445 • If an RE is empty (that is, no pattern is specified) *sed* shall behave as if the last RE used in the  
32446 last command applied (either as an address or as part of a substitute command) was  
32447 specified.

32448 **Editing Commands in sed**

32449 In the following list of editing commands, the maximum number of permissible addresses for  
32450 each function is indicated by [*0addr*], [*1addr*], or [*2addr*], representing zero, one, or two  
32451 addresses.

32452 The argument *text* shall consist of one or more lines. Each embedded <newline> in the text shall  
32453 be preceded by a backslash. Other backslashes in text shall be removed, and the following  
32454 character shall be treated literally.

32455 The *r* and *w* command verbs, and the *w* flag to the *s* command, take an optional *rfile* (or *wfile*)  
32456 parameter, separated from the command verb letter or flag by one or more <blank> characters;  
32457 implementations may allow zero separation as an extension.

32458 The argument *rfile* or the argument *wfile* shall terminate the editing command. Each *wfile* shall be  
32459 created before processing begins. Implementations shall support at least ten *wfile* arguments in  
32460 the script; the actual number (greater than or equal to 10) that shall be supported by the  
32461 implementation is unspecified. The use of the *wfile* parameter shall cause that file to be initially  
32462 created, if it does not exist, or shall replace the contents of an existing file.

32463 The **b**, **r**, **s**, **t**, **w**, **y**, and **:** command verbs shall accept additional arguments. The following  
 32464 synopses indicate which arguments shall be separated from the command verbs by a single  
 32465 <space>.

32466 The **a** and **r** commands schedule text for later output. The text specified for the **a** command, and  
 32467 the contents of the file specified for the **r** command, shall be written to standard output just  
 32468 before the next attempt to fetch a line of input when executing the **N** or **n** commands, or when  
 32469 reaching the end of the script. If written when reaching the end of the script, and the **-n** option  
 32470 was not specified, the text shall be written after copying the pattern space to standard output.  
 32471 The contents of the file specified for the **r** command shall be as of the time the output is written,  
 32472 not the time the **r** command is applied. The text shall be output in the order in which the **a** and **r**  
 32473 commands were applied to the input.

32474 Command verbs other than **{**, **a**, **b**, **c**, **i**, **r**, **t**, **w**, **:**, and **#** can be followed by a semicolon, optional  
 32475 <blank> characters, and another command verb. However, when the **s** command verb is used  
 32476 with the **w** flag, following it with another command in this manner produces undefined results.

32477 A function can be preceded by one or more **'!'** characters, in which case the function shall be  
 32478 applied if the addresses do not select the pattern space. Zero or more <blank> characters shall be  
 32479 accepted before the first **'!'** character. It is unspecified whether <blank> characters can follow a  
 32480 **'!'** character, and conforming applications shall not follow a **'!'** character with <blank>  
 32481 characters.

32482 **[2addr]{function**  
 32483 **function**  
 32484 ...  
 32485 **}** Execute a list of *sed* functions only when the pattern space is selected. The list of  
 32486 *sed* functions shall be surrounded by braces and separated by <newline>s, as  
 32487 follows. The braces can be preceded or followed by <blank> characters. The  
 32488 functions can be preceded by <blank> characters, but shall not be followed by  
 32489 <blank> characters. The <right-brace> shall be preceded by a <newline> and can  
 32490 be preceded or followed by <blank> characters.

32491 **[1addr]a\**  
 32492 **text** Write text to standard output as described previously.

32493 **[2addr]b [label]**  
 32494 Branch to the **:** function bearing the *label*. If *label* is not specified, branch to the end  
 32495 of the script. The implementation shall support *labels* recognized as unique up to  
 32496 at least 8 characters; the actual length (greater than or equal to 8) that shall be  
 32497 supported by the implementation is unspecified. It is unspecified whether  
 32498 exceeding a label length causes an error or a silent truncation.

32499 **[2addr]c\**  
 32500 **text** Delete the pattern space. With a 0 or 1 address or at the end of a 2-address range,  
 32501 place *text* on the output and start the next cycle.

32502 **[2addr]d** Delete the pattern space and start the next cycle.

32503 **[2addr]D** Delete the initial segment of the pattern space through the first <newline> and  
 32504 start the next cycle.

32505 **[2addr]g** Replace the contents of the pattern space by the contents of the hold space.

32506 **[2addr]G** Append to the pattern space a <newline> followed by the contents of the hold  
 32507 space.

|       |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 32508 | <b>[2addr]h</b>                       | Replace the contents of the hold space with the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 32509 | <b>[2addr]H</b>                       | Append to the hold space a <newline> followed by the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 32510 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32511 | <b>[1addr]i\</b>                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32512 | <i>text</i>                           | Write <i>text</i> to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 32513 | <b>[2addr]l</b>                       | (The letter ell.) Write the pattern space to standard output in a visually unambiguous form. The characters listed in the Base Definitions volume of IEEE Std. 1003.1-200x, Table 5-1, Escape Sequences and Associated Actions ('\\', '\a', '\b', '\f', '\r', '\t', '\v') shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-printable characters not in that table shall be written as one three-digit octal number (with a preceding backslash) for each byte in the character (most significant byte first). If the size of a byte on the system is greater than 9 bits, the format used for non-printable characters is implementation-defined. |
| 32514 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32515 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32516 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32517 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32518 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32519 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32520 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32521 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32522 |                                       | Long lines shall be folded, with the point of folding indicated by writing a backslash followed by a <newline>; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line shall be marked with a '\$'.                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 32523 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32524 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32525 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32526 | <b>[2addr]n</b>                       | Write the pattern space to standard output if the default output has not been suppressed, and replace the pattern space with the next line of input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 32527 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32528 |                                       | If no next line of input is available, the <b>n</b> command verb shall branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 32529 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32530 | <b>[2addr]N</b>                       | Append the next line of input to the pattern space, using an embedded <newline> character to separate the appended material from the original material. Note that the current line number changes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 32531 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32532 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32533 |                                       | If no next line of input is available, the <b>N</b> command verb shall branch to the end of the script and quit without starting a new cycle or copying the pattern space to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 32534 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32535 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32536 | <b>[2addr]p</b>                       | Write the pattern space to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 32537 | <b>[2addr]P</b>                       | Write the pattern space, up to the first <newline>, to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 32538 | <b>[1addr]q</b>                       | Branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 32539 | <b>[1addr]r rfile</b>                 | Copy the contents of <i>rfile</i> to standard output as described previously. If <i>rfile</i> does not exist or cannot be read, it shall be treated as if it were an empty file, causing no error condition.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 32540 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32541 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32542 | <b>[2addr]s/BRE/replacement/flags</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32543 |                                       | Substitute the replacement string for instances of the BRE in the pattern space. Any character other than backslash or <newline> can be used instead of a slash to delimit the BRE and the replacement. Within the BRE and the replacement, the BRE delimiter itself can be used as a literal character if it is preceded by a backslash.                                                                                                                                                                                                                                                                                                                                                             |
| 32544 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32545 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32546 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32547 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32548 |                                       | An ampersand ('&') appearing in the replacement shall be replaced by the string matching the BRE. The special meaning of '&' in this context can be suppressed by preceding it by a backslash. The characters "\n", where <i>n</i> is a digit, shall be replaced by the text matched by the corresponding backreference expression. For each backslash ('\')                                                                                                                                                                                                                                                                                                                                          |
| 32549 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32550 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32551 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 32552 |                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

32553 encountered in scanning **replacement** from beginning to end, the  
 32554 backslash shall be discarded and the following character  
 32555 shall lose its special meaning (if any). It is unspecified  
 32556 what special meaning is given to any character other than  
 32557 '&', '\', or digits.

32558 A line can be split by substituting a <newline> character  
 32559 into it. The application shall escape the <newline> in the  
 32560 replacement by preceding it by a backslash. A substitution  
 32561 shall be considered to have been performed even if the  
 32562 replacement string is identical to the string that it  
 32563 replaces. Any backslash used to alter the default meaning of  
 32564 a subsequent character shall be discarded from the BRE or the  
 32565 replacement before evaluating the BRE or using the  
 32566 replacement.

32567 The value of *flags* shall be zero or more of:

32568 **n** Substitute for the *n*th occurrence only of the BRE found within the  
 32569 pattern space.

32570 **g** Globally substitute for all non-overlapping instances of the BRE  
 32571 rather than just the first one. If both **g** and **n** are specified, the results  
 32572 are unspecified.

32573 **p** Write the pattern space to standard output if a replacement was  
 32574 made.

32575 **w wfile** Write. Append the pattern space to *wfile* if a replacement was made.  
 32576 A conforming application shall precede the *wfile* argument with one  
 32577 or more <blank> characters. If the *w* flag is not the last flag value  
 32578 given in a concatenation of multiple flag values, the results are  
 32579 undefined.

32580 **[2addr]t [label]**  
 32581 Test. Branch to the : command verb bearing the *label* if any substitutions have been  
 32582 made since the most recent reading of an input line or execution of a **t**. If *label* is  
 32583 not specified, branch to the end of the script.

32584 **[2addr]w wfile**  
 32585 Append (write) the pattern space to *wfile*.

32586 **[2addr]x** Exchange the contents of the pattern and hold spaces.

32587 **[2addr]y/string1/string2/**  
 32588 Replace all occurrences of characters in *string1* with the corresponding characters  
 32589 in *string2*. If a backslash followed by an 'n' appear in *string1* or *string2*, the two  
 32590 characters shall be handled as a single <newline> character. If the number of  
 32591 characters in *string1* and *string2* are not equal, or if any of the characters in *string1*  
 32592 appear more than once, the results are undefined. Any character other than  
 32593 backslash or <newline> can be used instead of slash to delimit the strings. If the  
 32594 delimiter is not *n*, within *string1* and *string2*, the delimiter itself can be used as a  
 32595 literal character if it is preceded by a backslash. If a backslash character is  
 32596 immediately followed by a backslash character in *string1* or *string2*, the two  
 32597 backslash characters shall be counted as a single literal backslash character. The  
 32598 meaning of a backslash followed by any character that is not 'n', a backslash, or  
 32599 the delimiter character is undefined.

32600        **[0addr]:label** Do nothing. This command bears a *label* to which the **b** and **t** commands branch.

32601        **[1addr]=**     Write the following to standard output:

32602                "%d\n", <current line number>

32603        **[0addr]**     Ignore this empty command.

32604        **[0addr]#**    Ignore the '#' and the remainder of the line (treat them as a comment), with the single exception that if the first two characters in the script are "#n", the default output shall be suppressed; this shall be the equivalent of specifying **-n** on the command line.

#### 32608 EXIT STATUS

32609        The following exit values shall be returned:

32610        **0**    Successful completion.

32611        **>0**   An error occurred.

#### 32612 CONSEQUENCES OF ERRORS

32613        Default.

#### 32614 APPLICATION USAGE

32615        Regular expressions match entire strings, not just individual lines, but a <newline> character is matched by '\n' in a *sed* RE; a <newline> character is not allowed by the general definition of regular expression in IEEE Std. 1003.1-200x. Also note that '\n' cannot be used to match a <newline> character at the end of an arbitrary input line; <newline> characters appear in the pattern space as a result of the **N** editing command.

#### 32620 EXAMPLES

32621        This *sed* script simulates the BSD *cat -s* command, squeezing excess blank lines from standard input.

```
32623 sed -n '
32624 # Write non-empty lines.
32625 ./ {
32626 p
32627 d
32628 }
32629 # Write a single empty line, then look for more empty lines.
32630 /^$/ p
32631 # Get next line, discard the held <newline> (empty line),
32632 # and look for more empty lines.
32633 :Empty
32634 /^$/ {
32635 N
32636 s/./ /
32637 b Empty
32638 }
32639 # Write the non-empty line before going back to search
32640 # for the first in a set of empty lines.
32641 p
32642 '
```

## 32643 RATIONALE

32644 This volume of IEEE Std. 1003.1-200x requires implementations to support at least ten distinct  
 32645 *wfiles*, matching historical practice on many implementations. Implementations are encouraged  
 32646 to support more, but portable applications should not exceed this limit.

32647 The exit status codes specified here are different from those in System V. System V returns 2 for  
 32648 garbled *sed* commands, but returns zero with its usage message or if the input file could not be  
 32649 opened. The standard developers considered this to be a bug.

32650 The manner in which the **l** command writes non-printable characters was changed to avoid the  
 32651 historical backspace-overstrike method, and other requirements to achieve unambiguous output  
 32652 were added. See the RATIONALE for *ed* (on page 2546) for details of the format chosen, which is  
 32653 the same as that chosen for *sed*.

32654 This volume of IEEE Std. 1003.1-200x requires implementations to provide pattern and hold  
 32655 spaces of at least 8 192 bytes, larger than the 4 000 bytes spaces used by some historical  
 32656 implementations, but less than the 20 480 bytes limit used in an early proposal. Implementations  
 32657 are encouraged to allocate dynamically larger pattern and hold spaces as needed.

32658 The requirements for acceptance of <blank> and <space> characters in command lines has been  
 32659 made more explicit than in early proposals to describe clearly the historical practice and to  
 32660 remove confusion about the phrase “protect initial blanks [*sic*] and tabs from the stripping that  
 32661 is done on every script line” that appears in much of the historical documentation of the *sed*  
 32662 utility description of text. (Not all implementations are known to have stripped <blank>  
 32663 characters from text lines, although they all have allowed leading <blank> characters preceding  
 32664 the address on a command line.)

32665 The treatment of ‘#’ comments differs from the SVID which only allows a comment as the first  
 32666 line of the script, but matches BSD-derived implementations. The comment character is treated  
 32667 as a command, and it has the same properties in terms of being accepted with leading <blank>  
 32668 characters; the BSD implementation has historically supported this.

32669 Early proposals required that a *script\_file* have at least one non-comment line. Some historical  
 32670 implementations have behaved in unexpected ways if this were not the case. The standard  
 32671 developers considered that this was incorrect behavior and that application developers should  
 32672 not have to avoid this feature. A correct implementation of this volume of IEEE Std. 1003.1-200x  
 32673 shall permit *script\_files* that consist only of comment lines.

32674 Early proposals indicated that if **-e** and **-f** options were intermixed, all **-e** options were  
 32675 processed before any **-f** options. This has been changed to process them in the order presented  
 32676 because it matches historical practice and is more intuitive.

32677 The treatment of the **p** flag to the **s** command differs between System V and BSD-based systems  
 32678 when the default output is suppressed. In the two examples:

```
32679 echo a | sed 's/a/A/p'
```

```
32680 echo a | sed -n 's/a/A/p'
```

32681 This volume of IEEE Std. 1003.1-200x, BSD, System V documentation, and the SVID indicate that  
 32682 the first example should write two lines with **A**, whereas the second should write one. Some  
 32683 System V systems write the **A** only once in both examples because the **p** flag is ignored if the **-n**  
 32684 option is not specified.

32685 This is a case of a diametrical difference between systems that could not be reconciled through  
 32686 the compromise of declaring the behavior to be unspecified. The SVID/BSD/System V  
 32687 documentation behavior was adopted for this volume of IEEE Std. 1003.1-200x because:



- 32688       • No known documentation for any historic system describes the interaction between the **p**  
32689       flag and the **-n** option.
- 32690       • The selected behavior is more correct as there is no technical justification for any interaction  
32691       between the **p** flag and the **-n** option. A relationship between **-n** and the **p** flag might imply  
32692       that they are only used together, but this ignores valid scripts that interrupt the cyclical  
32693       nature of the processing through the use of the **D**, **d**, **q**, or branching commands. Such scripts  
32694       rely on the **p** suffix to write the pattern space because they do not make use of the default  
32695       output at the “bottom” of the script.
- 32696       • Because the **-n** option makes the **p** flag unnecessary, any interaction would only be useful if  
32697       *sed* scripts were written to run both with and without the **-n** option. This is believed to be  
32698       unlikely. It is even more unlikely that programmers have coded the **p** flag expecting it to be  
32699       unnecessary. Because the interaction was not documented, the likelihood of a programmer  
32700       discovering the interaction and depending on it is further decreased.
- 32701       • Finally, scripts that break under the specified behavior produce too much output instead of  
32702       too little, which is easier to diagnose and correct.
- 32703       The form of the substitute command that uses the **n** suffix was limited to the first 512 matches in  
32704       an early proposal. This limit has been removed because there is no reason an editor processing  
32705       lines of {LINE\_MAX} length should have this restriction. The command **s/a/A/2047** should be  
32706       able to substitute the 2 047th occurrence of **a** on a line.
- 32707       The **b**, **t**, and **:** commands are documented to ignore leading white space, but no mention is  
32708       made of trailing white space. Historical implementations of *sed* assigned different locations to  
32709       the labels '**x**' and "**x**". This is not useful, and leads to subtle programming errors, but it is  
32710       historical practice, and changing it could theoretically break working scripts. Implementors are  
32711       encouraged to provide warning messages about labels that are never used or jumps to labels  
32712       that do not exist.
- 32713       Historically, the *sed* **!** and **}** editing commands did not permit multiple commands on a single  
32714       line using a semicolon as a command delimiter. Implementations are permitted, but not  
32715       required, to support this extension.
- 32716       **FUTURE DIRECTIONS**
- 32717       None.
- 32718       **SEE ALSO**
- 32719       *awk*, *ed*, *grep*
- 32720       **CHANGE HISTORY**
- 32721       First released in Issue 2.
- 32722       **Issue 4**
- 32723       Aligned with the ISO/IEC 9945-2: 1993 standard.
- 32724       **Issue 5**
- 32725       FUTURE DIRECTIONS section added.
- 32726       **Issue 6**
- 32727       The following new requirements on POSIX implementations derive from alignment with the  
32728       Single UNIX Specification:
- 32729       • Implementations are required to support at least ten *wfile* arguments in an editing command.
- 32730       The EXTENDED DESCRIPTION is changed to align with the IEEE P1003.2b draft standard.

## 32731 NAME

32732 sh — shell, the standard command language interpreter

## 32733 SYNOPSIS

32734 sh [-abCefhimnuvx][-o option][+abCefhimnuvx][+o option]  
 32735 [command\_file [argument...]]

32736 sh -c[-abCefhimnuvx][-o option][+abCefhimnuvx][+o option]command\_string  
 32737 [command\_name [argument...]]

32738 sh -s[-abCefhimnuvx][-o option][+abCefhimnuvx][+o option][argument]

## 32739 DESCRIPTION

32740 The *sh* utility is a command language interpreter that shall execute commands read from a  
 32741 command line string, the standard input, or a specified file. The application shall ensure that the  
 32742 commands to be executed are expressed in the language described in Chapter 2 (on page 2235).

32743 Path name expansion does not fail due to the size of a file.

## 32744 Notes to Reviewers

32745 *This section with side shading will not appear in the final copy. - Ed.*

32746 D3, XCU, ERN 215: Text here is unclear. There is nothing under the stat command that permits it  
 32747 to fail on a very large file.

32748 Shell input and output redirections have an implementation-defined offset maximum that is  
 32749 established in the open file description.

## 32750 OPTIONS

32751 The *sh* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,  
 32752 Utility Syntax Guidelines, with an extension for support of a leading plus sign ('+') as noted  
 32753 below.

32754 The **-a**, **-b**, **-C**, **-e**, **-f**, **-m**, **-n**, **-o option**, **-u**, **-v**, and **-x** options are described as part of the *set*  
 32755 utility in Section 2.15 (on page 2276). The option letters derived from the *set* special built-in shall  
 32756 also be accepted with a leading plus sign ('+') instead of a leading hyphen (meaning the reverse  
 32757 case of the option as described in this volume of IEEE Std. 1003.1-200x).

32758 The following additional options shall be supported:

32759 **-c** Read commands from the *command\_string* operand. Set the value of special  
 32760 parameter 0 (see Section 2.5.2 (on page 2241)) from the value of the *command\_name*  
 32761 operand and the positional parameters (\$1, \$2, and so on) in sequence from the  
 32762 remaining *argument* operands. No commands shall be read from the standard  
 32763 input.

32764 **-i** Specify that the shell is *interactive*; see below. An implementation may treat  
 32765 specifying the **-i** option as an error if the real user ID of the calling process does  
 32766 not equal the effective user ID or if the real group ID does not equal the effective  
 32767 group ID.

32768 **-s** Read commands from the standard input.

32769 If there are no operands and the **-c** option is not specified, the **-s** option shall be assumed.

32770 If the **-i** option is present, or if there are no operands and the shell's standard input and standard  
 32771 error are attached to a terminal, the shell is considered to be *interactive*.

32772 **OPERANDS**

32773 The following operands shall be supported:

32774 – A single hyphen is treated as the first operand and then ignored. If both ‘-’ and  
32775 “—” are given as arguments, or if other operands precede the single hyphen, the  
32776 results are undefined.

32777 *argument* The positional parameters (\$1, \$2, and so on) shall be set to *arguments*, if any.

32778 *command\_file* The path name of a file containing commands. If the path name contains one or  
32779 more slash characters, the implementation attempts to read that file; the file need  
32780 not be executable. If the path name does not contain a slash character:

32781 • The implementation shall attempt to read that file from the current working  
32782 directory; the file need not be executable.

32783 • If the file is not in the current working directory, the implementation may  
32784 perform a search for an executable file using the value of *PATH*, as described in  
32785 Section 2.9.1.1 (on page 2257).

32786 Special parameter 0 (see Section 2.5.2 (on page 2241)) shall be set to the value of  
32787 *command\_file*. If *sh* is called using a synopsis form that omits *command\_file*, special  
32788 parameter 0 shall be set to the value of the first argument passed to *sh* from its  
32789 parent (for example, *argv*[0] for a C program), which is normally a path name used  
32790 to execute the *sh* utility.

32791 *command\_name*

32792 A string assigned to special parameter 0 when executing the commands in  
32793 *command\_string*. If *command\_name* is not specified, special parameter 0 shall be set  
32794 to the value of the first argument passed to *sh* from its parent (for example, *argv*[0]  
32795 for a C program), which is normally a path name used to execute the *sh* utility.

32796 *command\_string*

32797 A string that shall be interpreted by the shell as one or more commands, as if the  
32798 string were the argument to the *system*() function defined in the System Interfaces  
32799 volume of IEEE Std. 1003.1-200x. If the *command\_string* operand is an empty string,  
32800 *sh* shall exit with a zero exit status.

32801 **STDIN**

32802 The standard input shall be used only if one of the following is true:

32803 • The *-s* option is specified.

32804 • The *-c* option is not specified and no operands are specified.

32805 • The script executes one or more commands that require input from standard input (such as a  
32806 *read* command that does not redirect its input).

32807 See the INPUT FILES section.

32808 When the shell is using standard input and it invokes a command that also uses standard input,  
32809 the shell shall ensure that the standard input file pointer points directly after the command it has  
32810 read when the command begins execution. It shall not read ahead in such a manner that any  
32811 characters intended to be read by the invoked command are consumed by the shell (whether  
32812 interpreted by the shell or not) or that characters that are not read by the invoked command are  
32813 not seen by the shell. When the command expecting to read standard input is started  
32814 asynchronously by an interactive shell, it is unspecified whether characters are read by the  
32815 command or interpreted by the shell.

32816 If the standard input to *sh* is a FIFO or terminal device and is set to non-blocking reads, then *sh*  
 32817 shall enable blocking reads on standard input. This shall remain in effect when the command  
 32818 completes.

#### 32819 INPUT FILES

32820 The input file shall be a text file, except that line lengths shall be unlimited. If the input file is  
 32821 empty or consists solely of blank lines or comments, or both, *sh* shall exit with a zero exit status.

#### 32822 ENVIRONMENT VARIABLES

32823 The following environment variables shall affect the execution of *sh*:

32824 *ENV* This variable, when and only when an interactive shell is invoked, shall be  
 32825 subjected to parameter expansion (see Section 2.6.2 (on page 2245)) by the shell,  
 32826 and the resulting value shall be used as a path name of a file containing shell  
 32827 commands to execute in the current environment. The file need not be executable.  
 32828 If the expanded value of *ENV* is not an absolute path name, the results are  
 32829 unspecified. *ENV* shall be ignored if the real and effective user IDs or real and  
 32830 effective group IDs of the process are different.

32831 *FCEDIT* This variable, when expanded by the shell, determines the default value for the *-e*  
 32832 *editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be used  
 32833 as the editor. This volume of IEEE Std. 1003.1-200x specifies the effects of this  
 32834 variable only for systems supporting the User Portability Utilities option.

32835 *HISTFILE* Determine a path name naming a command history file. If the *HISTFILE* variable is  
 32836 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 32837 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 32838 and write access to, or create, the history file, it shall use an unspecified  
 32839 mechanism that allows the history to operate properly. (References to history  
 32840 "file" in this section shall be understood to mean this unspecified mechanism in  
 32841 such cases.) An implementation may choose to access this variable only when  
 32842 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 32843 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 32844 by the user, the file named by the *ENV* variable, or implementation-defined system  
 32845 start-up files. (The initialization process for the history file can be dependent on the  
 32846 system start-up files, in that they may contain commands that effectively preempt  
 32847 the user's settings of *HISTFILE* and *HISTSIZE*. For example, function definition  
 32848 commands are recorded in the history file, unless the *set -o nolog* option is set. If  
 32849 the system administrator includes function definitions in some system start-up file  
 32850 called before the *ENV* file, the history file is initialized before the user gets a chance  
 32851 to influence its characteristics.) In some historical shells, the history file is  
 32852 initialized just after the *ENV* file has been processed. Therefore, it is  
 32853 implementation-defined whether changes made to *HISTFILE* after the history file  
 32854 has been initialized are effective. Implementations may choose to disable the  
 32855 history list mechanism for users with appropriate privileges who do not set  
 32856 *HISTFILE*; the specific circumstances under which this occurs are  
 32857 implementation-defined. If more than one instance of the shell is using the same  
 32858 history file, it is unspecified how updates to the history file from those shells  
 32859 interact. As entries are deleted from the history file, they shall be deleted oldest  
 32860 first. It is unspecified when history file entries are physically removed from the  
 32861 history file. This volume of IEEE Std. 1003.1-200x specifies the effects of this  
 32862 variable only for systems supporting the User Portability Utilities option.

32863 *HISTSIZE* Determine a decimal number representing the limit to the number of previous  
 32864 commands that are accessible. If this variable is unset, an unspecified default

- 32865 greater than or equal to 128 shall be used. The maximum number of commands in  
 32866 the history list is unspecified, but shall be at least 128. An implementation may  
 32867 choose to access this variable only when initializing the history file, as described  
 32868 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSZ*  
 32869 after the history file has been initialized are effective.
- 32870 *HOME* Determine the path name of the user's home directory. The contents of *HOME* are  
 32871 used in Tilde Expansion as described in Section 2.6.1 (on page 2244). This volume  
 32872 of IEEE Std. 1003.1-200x specifies the effects of this variable only for systems  
 32873 supporting the User Portability Utilities option.
- 32874 *IFS* *Input field separators*: a string treated as a list of characters that shall be used for  
 32875 field splitting and to split lines into words with the *read* command. See Section  
 32876 2.6.5 (on page 2249). If *IFS* is not set, the shell shall behave as if the value of *IFS*  
 32877 were the <space>, <tab>, and <newline> characters. Implementations may ignore  
 32878 the value of *IFS* in the environment at the time *sh* is invoked, treating *IFS* as if it  
 32879 were not set.
- 32880 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 32881 If *LANG* is unset or null, the corresponding value from the implementation-  
 32882 defined default locale shall be used. If any of the internationalization variables  
 32883 contains an invalid setting, the utility shall behave as if none of the variables had  
 32884 been defined.
- 32885 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 32886 internationalization variables.
- 32887 *LC\_COLLATE*  
 32888 Determine the behavior of range expressions, equivalence classes and multi-  
 32889 character collating elements within pattern matching.
- 32890 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 32891 characters (for example, single-byte as opposed to multi-byte characters in  
 32892 arguments and input files), which characters are defined as letters (character class  
 32893 **alpha**), and the behavior of character classes within pattern matching.
- 32894 *LC\_MESSAGES*  
 32895 Determine the locale that should be used to affect the format and contents of  
 32896 diagnostic messages written to standard error.
- 32897 *MAIL* Determine a path name of the user's mailbox file for purposes of incoming mail  
 32898 notification. If this variable is set, the shell shall inform the user if the file named by  
 32899 the variable is created or if its modification time has changed. Informing the user  
 32900 shall be accomplished by writing a string of unspecified format to standard error  
 32901 prior to the writing of the next primary prompt string after the completion of an  
 32902 interval defined by the *MAILCHECK* variable. The user shall be informed only if  
 32903 *MAIL* is set and *MAILPATH* is not set. This volume of IEEE Std. 1003.1-200x  
 32904 specifies the effects of this variable only for systems supporting the User  
 32905 Portability Utilities option.
- 32906 *MAILCHECK*  
 32907 Establish a decimal integer value that specifies how often (in seconds) the shell  
 32908 shall check for the arrival of mail in the files specified by the *MAILPATH* or *MAIL*  
 32909 variables. The default value shall be 600 seconds. If set to zero, the shell shall check  
 32910 before issuing each primary prompt. This volume of IEEE Std. 1003.1-200x  
 32911 specifies the effects of this variable only for systems supporting the User  
 32912 Portability Utilities option.

- 32913 **MAILPATH** Provide a list of path names and optional messages separated by colons. If this  
 32914 variable is set, the shell shall inform the user if any of the files named by the  
 32915 variable are created or if any of their modification times change. (See the preceding  
 32916 entry for *MAIL* for descriptions of mail arrival and user informing.) Each path  
 32917 name can be followed by '%' and a string that shall be subjected to parameter  
 32918 expansion and written to standard error when the modification time changes. If a  
 32919 '%' character in the path name is preceded by a backslash, it shall be treated as a  
 32920 literal '%' in the path name. The default message is unspecified.
- 32921 The *MAILPATH* environment variable takes precedence over the *MAIL* variable.  
 32922 This volume of IEEE Std. 1003.1-200x specifies the effects of this variable only for  
 32923 systems supporting the User Portability Utilities option.
- 32924 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 32925 **PATH** Establish a string formatted as described in the Base Definitions volume of  
 32926 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables, used to effect command  
 32927 interpretation; see Section 2.9.1.1 (on page 2257).
- 32928 **PWD** This variable shall represent an absolute path name of the current working  
 32929 directory. Assignments to this variable may be ignored unless the value is an  
 32930 absolute path name of the current working directory and there are no file name  
 32931 components of dot or dot-dot.
- 32932 **ASYNCHRONOUS EVENTS**
- 32933 Default.
- 32934 **STDOUT**
- 32935 See the *STDERR* section.
- 32936 **STDERR**
- 32937 Except as otherwise stated (by the descriptions of any invoked utilities or in interactive mode),  
 32938 standard error is used only for diagnostic messages.
- 32939 **OUTPUT FILES**
- 32940 None.
- 32941 **EXTENDED DESCRIPTION**
- 32942 See Chapter 2. The following additional capabilities are supported on systems supporting the  
 32943 User Portability Utilities option.
- 32944 **Command History List**
- 32945 When the *sh* utility is being used interactively, it shall maintain a list of commands previously  
 32946 entered from the terminal in the file named by the *HISTFILE* environment variable. The type,  
 32947 size, and internal format of this file are unspecified. Multiple *sh* processes can share access to the  
 32948 file for a user, if file access permissions allow this; see the description of the *HISTFILE*  
 32949 environment variable.

32950 **Command Line Editing**

32951 When *sh* is being used interactively from a terminal, the current command and the command  
 32952 history (see *fc* (on page 2646)) can be edited using *vi*-mode command line editing. This mode  
 32953 uses commands, described below, similar to a subset of those described in the *vi* utility.  
 32954 Implementations may offer other command line editing modes corresponding to other editing  
 32955 utilities.

32956 The command *set -o vi* shall enable *vi*-mode editing and place *sh* into *vi* insert mode (see  
 32957 **Command Line Editing (vi-mode)**). This command also shall disable any other editing mode  
 32958 that the implementation may provide. The command *set +o vi* disables *vi*-mode editing.

32959 Certain block-mode terminals may be unable to support shell command line editing. If a  
 32960 terminal is unable to provide either edit mode, it need not be possible to *set -o vi* when using the  
 32961 shell on this terminal.

32962 In the following sections, the characters *erase*, *interrupt*, *kill*, and *end-of-file* are those set by the  
 32963 *stty* utility.

32964 **Command Line Editing (vi-mode)**

32965 With *vi*-mode enabled, *sh* can be switched between insert mode and command mode.

32966 When in insert mode, an entered character shall be inserted into the command line, except as  
 32967 noted in **vi Line Editing Insert Mode**. Upon entering *sh* and after termination of the previous  
 32968 command, *sh* shall be in insert mode.

32969 Typing an escape character shall switch *sh* into command mode (see **vi Line Editing Command**  
 32970 **Mode** (on page 3066)). In command mode, an entered character shall either invoke a defined  
 32971 operation, is used as part of a multi-character operation, or is treated as an error. A character that  
 32972 is not recognized as part of an editing command shall terminate any specific editing command  
 32973 and shall alert the terminal. Typing the *interrupt* character in command mode shall cause *sh* to  
 32974 terminate command line editing on the current command line, reissue the prompt on the next  
 32975 line of the terminal, and reset the command history (see *fc* (on page 2646)) so that the most  
 32976 recently executed command is the previous command (that is, the command that was being  
 32977 edited when it was interrupted is not reentered into the history).

32978 In the following sections, the phrase “move the cursor to the beginning of the word” shall mean  
 32979 “move the cursor to the first character of the current word” and the phrase “move the cursor to  
 32980 the end of the word” shall mean “move the cursor to the last character of the current word”. The  
 32981 phrase “beginning of the command line” indicates the point between the end of the prompt  
 32982 string issued by the shell (or the beginning of the terminal line, if there is no prompt string) and  
 32983 the first character of the command text.

32984 **vi Line Editing Insert Mode**

32985 While in insert mode, any character typed shall be inserted in the current command line, unless  
 32986 it is from the following set.

32987 <newline> Execute the current command line being edited.

32988 *erase* Delete the character previous to the current cursor position and move the current  
 32989 cursor position back one character. In insert mode, characters shall be erased from  
 32990 both the screen and the buffer when backspacing.

32991 *interrupt* Terminate command line editing with the same effects as described for  
 32992 interrupting command mode; see **Command Line Editing (vi-mode)**.

|       |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 32993 | <i>kill</i>                         | Clear all the characters from the input line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 32994 | <control>-V                         | Insert the next character input, even if the character is otherwise a special insert mode character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 32995 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 32996 | <control>-W                         | Delete the characters from the one preceding the cursor to the preceding word boundary. The word boundary in this case is the closer to the cursor of either the beginning of the line or a character that is in neither the <b>blank</b> nor <b>punct</b> character classification of the current locale.                                                                                                                                                                                                                                                                   |
| 32997 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 32998 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 32999 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33000 | <i>end-of-file</i>                  | Interpreted as the end of input in <i>sh</i> . This interpretation shall occur only at the beginning of an input line. If <i>end-of-file</i> is entered other than at the beginning of the line, the results are unspecified.                                                                                                                                                                                                                                                                                                                                                |
| 33001 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33002 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33003 | <ESC>                               | Place <i>sh</i> into command mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 33004 | <b>vi Line Editing Command Mode</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33005 |                                     | In command mode for the command line editing feature, decimal digits not beginning with 0 that precede a command letter shall be remembered. Some commands use these decimal digits as a count number that affects the operation.                                                                                                                                                                                                                                                                                                                                            |
| 33006 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33007 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33008 |                                     | The term <i>motion command</i> represents one of the commands:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 33009 | <space>                             | 0 b F l W ^ \$ ; E f T w   , B e h t                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 33010 |                                     | Any command that modifies the current line shall cause a copy of the current line to be made at the end of the command history, the current line shall become that copy, and the edit is performed on that copy.                                                                                                                                                                                                                                                                                                                                                             |
| 33011 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33012 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33013 |                                     | Any command that is preceded by <i>count</i> shall take a count (the numeric value of any preceding decimal digits). Unless otherwise noted, this count shall cause the specified operation to repeat by the number of times specified by the count. Also unless otherwise noted, a <i>count</i> that is out of range is considered an error condition and shall alert the terminal, but neither the cursor position, nor the command line, shall change.                                                                                                                    |
| 33014 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33015 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33016 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33017 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33018 |                                     | The terms <i>word</i> and <i>bigword</i> are used as defined in the <i>vi</i> description. The term <i>save buffer</i> corresponds to the term <i>unnamed buffer</i> in <i>vi</i> .                                                                                                                                                                                                                                                                                                                                                                                          |
| 33019 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33020 |                                     | The following commands shall be recognized in command mode:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 33021 | <newline>                           | Execute the current command line being edited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 33022 | <control>-L                         | Redraw the current command line. Position the cursor at the same location on the new command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 33023 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33024 | #                                   | Insert the character '#' at the beginning of the current command line and treat the current command line as a comment. This line shall be entered into the command history; see <i>fc</i> (on page 2646).                                                                                                                                                                                                                                                                                                                                                                    |
| 33025 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33026 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33027 | =                                   | Display the possible shell word expansions (see Section 2.6 (on page 2244)) of the bigword at the current command line position. These expansions shall be displayed on subsequent terminal lines. If the bigword contains none of the characters '?', '*', or '[', an asterisk('*') shall be implicitly assumed at the end. If any directories are matched, these expansions shall have a '/' character appended. After the expansion, the line shall be redrawn, the cursor is repositioned at the current cursor position, and <i>sh</i> shall be placed in command mode. |
| 33028 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33029 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33030 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33031 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33032 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33033 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33034 | \                                   | Perform path name expansion (see Section 2.6.6 (on page 2249)) on the current bigword, up to the largest set of characters that can be matched uniquely. If the                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33035 |                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



33036 bigword contains none of the characters '?', '\*', or '[', an asterisk ('\*') shall  
 33037 be implicitly assumed at the end. This maximal expansion then shall replace the  
 33038 original bigword in the command line, and the cursor shall be placed after this  
 33039 expansion. If the resulting bigword completely and uniquely matches a directory, a  
 33040 '/' character shall be inserted directly after the bigword. If some other file is  
 33041 completely matched, a single <space> character shall be inserted after the bigword.  
 33042 After this operation, *sh* shall be placed in insert mode.

33043 \* Perform path name expansion on the current bigword and insert all expansions  
 33044 into the command to replace the current bigword, with each expansion separated  
 33045 by a single <space> character. If at the end of the line, the current cursor position  
 33046 shall be moved to the first column position following the expansions and *sh* shall  
 33047 be placed in insert mode. Otherwise, the current cursor position shall be the last  
 33048 column position of the first character after the expansions and *sh* shall be placed in  
 33049 insert mode. If the current bigword contains none of the characters '?', '\*', or  
 33050 '[', before the operation, an asterisk shall be implicitly assumed at the end.

33051 @*letter* Insert the value of the alias named *\_letter*. The symbol *letter* represents a single  
 33052 alphabetic character from the portable character set; implementations may support  
 33053 additional characters as an extension. If the alias *\_letter* contains other editing  
 33054 commands, these commands shall be performed as part of the insertion. If no alias  
 33055 *\_letter* is enabled, this command shall have no effect.

33056 [*count*]~ Convert, if the current character is a lowercase letter, to the equivalent uppercase  
 33057 letter and *vice versa*, as prescribed by the current locale. The current cursor position  
 33058 then shall be advanced by one character. If the cursor was positioned on the last  
 33059 character of the line, the case conversion shall occur, but the cursor shall not  
 33060 advance. If the '~' command is preceded by a *count*, that number of characters  
 33061 shall be converted, and the cursor shall be advanced to the character position after  
 33062 the last character converted. If the *count* is larger than the number of characters  
 33063 after the cursor, this shall not be considered an error; the cursor shall advance to  
 33064 the last character on the line.

33065 [*count*]. Repeat the most recent non-motion command, even if it was executed on an earlier  
 33066 command line. If the previous command was preceded by a *count*, and no count is  
 33067 given on the '.' command, the count from the previous command shall be  
 33068 included as part of the repeated command. If the '.' command is preceded by a  
 33069 *count*, this shall override any *count* argument to the previous command. The *count*  
 33070 specified in the '.' command shall become the count for subsequent '.'  
 33071 commands issued without a count.

33072 [*number*]v Invoke the *vi* editor to edit the current command line in a temporary file. When the  
 33073 editor exits, the commands in the temporary file shall be executed. If a *number* is  
 33074 prefixed to the command, it specifies the command number in the command  
 33075 history to be edited, rather than the current command line.

33076 [*count*]l (ell)  
 33077 [*count*]<space>  
 33078 Move the current cursor position to the next character position. If the cursor was  
 33079 positioned on the last character of the line, the terminal shall be alerted and the  
 33080 cursor shall not be advanced. If the *count* is larger than the number of characters  
 33081 after the cursor, this shall not be considered an error; the cursor shall advance to  
 33082 the last character on the line.

33083 [*count*]h Move the current cursor position to the *count*th (default 1) previous character  
 33084 position. If the cursor was positioned on the first character of the line, the terminal

|       |                  |                                                                                               |
|-------|------------------|-----------------------------------------------------------------------------------------------|
| 33085 |                  | shall be alerted and the cursor shall not be moved. If the count is larger than the           |
| 33086 |                  | number of characters before the cursor, this shall not be considered an error; the            |
| 33087 |                  | cursor shall move to the first character on the line.                                         |
| 33088 | <b>[count]w</b>  | Move to the start of the next word. If the cursor was positioned on the last                  |
| 33089 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 33090 |                  | advanced. If the <i>count</i> is larger than the number of words after the cursor, this shall |
| 33091 |                  | not be considered an error; the cursor shall advance to the last character on the             |
| 33092 |                  | line.                                                                                         |
| 33093 | <b>[count]W</b>  | Move to the start of the next bigword. If the cursor was positioned on the last               |
| 33094 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 33095 |                  | advanced. If the <i>count</i> is larger than the number of bigwords after the cursor, this    |
| 33096 |                  | shall not be considered an error; the cursor shall advance to the last character on           |
| 33097 |                  | the line.                                                                                     |
| 33098 | <b>[count]e</b>  | Move to the end of the current word. If at the end of a word, move to the end of the          |
| 33099 |                  | next word. If the cursor was positioned on the last character of the line, the                |
| 33100 |                  | terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger |
| 33101 |                  | than the number of words after the cursor, this shall not be considered an error; the         |
| 33102 |                  | cursor shall advance to the last character on the line.                                       |
| 33103 | <b>[count]E</b>  | Move to the end of the current bigword. If at the end of a bigword, move to the               |
| 33104 |                  | end of the next bigword. If the cursor was positioned on the last character of the            |
| 33105 |                  | line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> |
| 33106 |                  | is larger than the number of bigwords after the cursor, this shall not be considered          |
| 33107 |                  | an error; the cursor shall advance to the last character on the line.                         |
| 33108 | <b>[count]b</b>  | Move to the beginning of the current word. If at the beginning of a word, move to             |
| 33109 |                  | the beginning of the previous word. If the cursor was positioned on the first                 |
| 33110 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 33111 |                  | moved. If the <i>count</i> is larger than the number of words preceding the cursor, this      |
| 33112 |                  | shall not be considered an error; the cursor shall return to the first character on the       |
| 33113 |                  | line.                                                                                         |
| 33114 | <b>[count]B</b>  | Move to the beginning of the current bigword. If at the beginning of a bigword,               |
| 33115 |                  | move to the beginning of the previous bigword. If the cursor was positioned on the            |
| 33116 |                  | first character of the line, the terminal shall be alerted and the cursor shall not be        |
| 33117 |                  | moved. If the <i>count</i> is larger than the number of bigwords preceding the cursor,        |
| 33118 |                  | this shall not be considered an error; the cursor shall return to the first character on      |
| 33119 |                  | the line.                                                                                     |
| 33120 | <b>^</b>         | Move the current cursor position to the first character on the input line that is not a       |
| 33121 |                  | <blank> character.                                                                            |
| 33122 | <b>\$</b>        | Move to the last character position on the current command line.                              |
| 33123 | <b>0</b>         | (Zero.) Move to the first character position on the current command line.                     |
| 33124 | <b>[count]  </b> | Move to the <i>count</i> th character position on the current command line. If no number      |
| 33125 |                  | is specified, move to the first position. The first character position shall be               |
| 33126 |                  | numbered 1. If the count is larger than the number of characters on the line, this            |
| 33127 |                  | shall not be considered an error; the cursor shall be placed on the last character on         |
| 33128 |                  | the line.                                                                                     |
| 33129 | <b>[count]fc</b> | Move to the first occurrence of the character 'c' that occurs after the current               |
| 33130 |                  | cursor position. If the cursor was positioned on the last character of the line, the          |
| 33131 |                  | terminal shall be alerted and the cursor shall not be advanced. If the character 'c'          |

|       |                       |                                                                                                          |
|-------|-----------------------|----------------------------------------------------------------------------------------------------------|
| 33132 |                       | does not occur in the line after the current cursor position, the terminal shall be                      |
| 33133 |                       | alerted and the cursor shall not be moved.                                                               |
| 33134 | <b>[count]Fc</b>      | Move to the first occurrence of the character 'c' that occurs before the current                         |
| 33135 |                       | cursor position. If the cursor was positioned on the first character of the line, the                    |
| 33136 |                       | terminal shall be alerted and the cursor shall not be moved. If the character 'c'                        |
| 33137 |                       | does not occur in the line before the current cursor position, the terminal shall be                     |
| 33138 |                       | alerted and the cursor shall not be moved.                                                               |
| 33139 | <b>[count]tc</b>      | Move to the character before the first occurrence of the character 'c' that occurs                       |
| 33140 |                       | after the current cursor position. If the cursor was positioned on the last character                    |
| 33141 |                       | of the line, the terminal shall be alerted and the cursor shall not be advanced. If the                  |
| 33142 |                       | character 'c' does not occur in the line after the current cursor position, the                          |
| 33143 |                       | terminal shall be alerted and the cursor shall not be moved.                                             |
| 33144 | <b>[count]Tc</b>      | Move to the character after the first occurrence of the character 'c' that occurs                        |
| 33145 |                       | before the current cursor position. If the cursor was positioned on the first                            |
| 33146 |                       | character of the line, the terminal shall be alerted and the cursor shall not be                         |
| 33147 |                       | moved. If the character 'c' does not occur in the line before the current cursor                         |
| 33148 |                       | position, the terminal shall be alerted and the cursor shall not be moved.                               |
| 33149 | <b>[count];</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that |
| 33150 |                       | previous command shall be ignored. Errors are those described for the repeated                           |
| 33151 |                       | command.                                                                                                 |
| 33152 | <b>[count],</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that |
| 33153 |                       | previous command shall be ignored. However, reverse the direction of that                                |
| 33154 |                       | command.                                                                                                 |
| 33155 | <b>a</b>              | Enter insert mode after the current cursor position. Characters that are entered                         |
| 33156 |                       | shall be inserted before the next character.                                                             |
| 33157 | <b>A</b>              | Enter insert mode after the end of the current command line.                                             |
| 33158 | <b>i</b>              | Enter insert mode at the current cursor position. Characters that are entered are                        |
| 33159 |                       | inserted before the current character.                                                                   |
| 33160 | <b>I</b>              | Enter insert mode at the beginning of the current command line.                                          |
| 33161 | <b>R</b>              | Enter insert mode, replacing characters from the command line beginning at the                           |
| 33162 |                       | current cursor position.                                                                                 |
| 33163 | <b>[count]cmotion</b> |                                                                                                          |
| 33164 |                       | Delete the characters between the current cursor position and the cursor position                        |
| 33165 |                       | that would result from the specified <i>motion</i> command. Then enter insert mode                       |
| 33166 |                       | before the first character following any deleted characters. If <i>count</i> is specified, it            |
| 33167 |                       | shall be applied to the motion command. A <i>count</i> shall be ignored for the following                |
| 33168 |                       | motion commands:                                                                                         |
| 33169 |                       | 0    ^    \$    c                                                                                        |
| 33170 |                       | If the <i>motion</i> command is the character 'c', the current command line shall be                     |
| 33171 |                       | cleared and insert mode shall be entered. If the <i>motion</i> command would move the                    |
| 33172 |                       | current cursor position toward the beginning of the command line, the character                          |
| 33173 |                       | under the current cursor position shall not be deleted. If the motion command                            |
| 33174 |                       | would move the current cursor position toward the end of the command line, the                           |
| 33175 |                       | character under the current cursor position shall be deleted. If the <i>count</i> is larger              |
| 33176 |                       | than the number of characters between the current cursor position and the end of                         |
| 33177 |                       | the command line toward which the motion command would move the cursor,                                  |

|       |                       |                                                                                                  |
|-------|-----------------------|--------------------------------------------------------------------------------------------------|
| 33178 |                       | this shall not be considered an error; all of the remaining characters in the                    |
| 33179 |                       | aforementioned range shall be deleted and insert mode shall be entered. If the                   |
| 33180 |                       | motion command is invalid, the terminal shall be alerted, the cursor shall not be                |
| 33181 |                       | moved, and no text shall be deleted.                                                             |
| 33182 | <b>C</b>              | Delete from the current character to the end of the line and enter insert mode at the            |
| 33183 |                       | new end-of-line.                                                                                 |
| 33184 | <b>S</b>              | Clear the entire current command line and enter insert mode.                                     |
| 33185 | <b>[count]rc</b>      | Replace the current character with the character 'c'. With a number <i>count</i> ,               |
| 33186 |                       | replace the current and the following <i>count</i> -1 characters. After this command, the        |
| 33187 |                       | current cursor position shall be on the last character that was changed. If the <i>count</i>     |
| 33188 |                       | is larger than the number of characters after the cursor, this shall not be considered           |
| 33189 |                       | an error; all of the remaining characters shall be changed.                                      |
| 33190 | <b>[count]_</b>       | Append a <space> character after the current character position and then append                  |
| 33191 |                       | the last bigword in the previous input line after the <space> character. Then enter              |
| 33192 |                       | insert mode after the last character just appended. With a number <i>count</i> , append          |
| 33193 |                       | the <i>count</i> th bigword in the previous line.                                                |
| 33194 | <b>[count]x</b>       | Delete the character at the current cursor position and place the deleted characters             |
| 33195 |                       | in the save buffer. If the cursor was positioned on the last character of the line, the          |
| 33196 |                       | character shall be deleted and the cursor position shall be moved to the previous                |
| 33197 |                       | character (the new last character). If the <i>count</i> is larger than the number of             |
| 33198 |                       | characters after the cursor, this shall not be considered an error; all the characters           |
| 33199 |                       | from the cursor to the end of the line shall be deleted.                                         |
| 33200 | <b>[count]X</b>       | Delete the character before the current cursor position and place the deleted                    |
| 33201 |                       | characters in the save buffer. The character under the current cursor position shall             |
| 33202 |                       | not change. If the cursor was positioned on the first character of the line, the                 |
| 33203 |                       | terminal shall be alerted, and the <b>X</b> command shall have no effect. If the line            |
| 33204 |                       | contained a single character, the <b>X</b> command shall have no effect. If the line             |
| 33205 |                       | contained no characters, the terminal shall be alerted and the cursor shall not be               |
| 33206 |                       | moved. If the <i>count</i> is larger than the number of characters before the cursor, this       |
| 33207 |                       | shall not be considered an error; all the characters from before the cursor to the               |
| 33208 |                       | beginning of the line shall be deleted.                                                          |
| 33209 | <b>[count]dmotion</b> |                                                                                                  |
| 33210 |                       | Delete the characters between the current cursor position and the character                      |
| 33211 |                       | position that would result from the <i>motion</i> command. A number <i>count</i> repeats the     |
| 33212 |                       | <i>motion</i> command <i>count</i> times. If the <i>motion</i> command would move toward the     |
| 33213 |                       | beginning of the command line, the character under the current cursor position                   |
| 33214 |                       | shall not be deleted. If the <i>motion</i> command is <b>d</b> , the entire current command line |
| 33215 |                       | shall be cleared. If the <i>count</i> is larger than the number of characters between the        |
| 33216 |                       | current cursor position and the end of the command line toward which the motion                  |
| 33217 |                       | command would move the cursor, this shall not be considered an error; all of the                 |
| 33218 |                       | remaining characters in the aforementioned range shall be deleted. The deleted                   |
| 33219 |                       | characters shall be placed in the save buffer.                                                   |
| 33220 | <b>D</b>              | Delete all characters from the current cursor position to the end of the line. The               |
| 33221 |                       | deleted characters shall be placed in the save buffer.                                           |
| 33222 | <b>[count]ymotion</b> |                                                                                                  |
| 33223 |                       | Yank (that is, copy) the characters from the current cursor position to the position             |
| 33224 |                       | resulting from the <i>motion</i> command into a save buffer. A number <i>count</i> shall be      |
| 33225 |                       | applied to the <i>motion</i> command. If the <i>motion</i> command would move toward the         |

|       |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 33226 |                               | beginning of the command line, the character under the current cursor position shall not be included in the set of yanked characters. If the <i>motion</i> command is <i>y</i> , the entire current command line shall be yanked into the save buffer. The current cursor position shall be unchanged. If the <i>count</i> is larger than the number of characters between the current cursor position and the end of the command line toward which the motion command would move the cursor, this shall not be considered an error; all of the remaining characters in the aforementioned range shall be yanked. |
| 33227 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33228 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33229 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33230 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33231 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33232 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33233 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33234 | <b>Y</b>                      | Yank the characters from the current cursor position to the end of the line into the save buffer. The current character position shall be unchanged.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 33235 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33236 | <b>[count]p</b>               | Put a copy of the current contents of the save buffer after the current cursor position. The current cursor position shall be advanced to the last character put from the save buffer. A <i>count</i> shall indicate how many copies of the save buffer shall be put.                                                                                                                                                                                                                                                                                                                                             |
| 33237 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33238 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33239 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33240 | <b>[count]P</b>               | Put a copy of the current contents of the save buffer before the current cursor position. The current cursor position shall be moved to the last character put from the save buffer. A <i>count</i> shall indicate how many copies of the save buffer shall be put.                                                                                                                                                                                                                                                                                                                                               |
| 33241 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33242 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33243 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33244 | <b>u</b>                      | Undo the last command that modified the text of the current command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 33245 | <b>U</b>                      | Undo all changes made to the current command line since first entering command mode on the line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 33246 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33247 | <b>[count]k</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33248 | <b>[count]-</b>               | Replace the current command line with the previous command line in the shell command history. The cursor shall be positioned on the first character of the new command. A count preceding the command shall have the same effect as executing the command <i>count</i> times. If a <b>k</b> or <b>-</b> command retreats past the maximum number of commands in effect for this shell (affected by the <i>HISTSIZE</i> environment variable), the terminal shall be alerted and the command shall have no effect.                                                                                                 |
| 33249 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33250 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33251 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33252 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33253 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33254 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33255 | <b>[count]j</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33256 | <b>[count]+</b>               | Replace the current command line with the next command line in the shell command history. The cursor shall be positioned on the first character of the new command. The command history position shall be remembered, and any <b>k</b> or <b>-</b> command, or <b>j</b> or <b>+</b> command, shall decrement or increment that position and then shall fetch the line at the new position. If a <b>j</b> or <b>+</b> command advances past the most recent line in the history, the current command line shall be restored to the contents before the first <b>k</b> or <b>-</b> .                                |
| 33257 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33258 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33259 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33260 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33261 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33262 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33263 | <b>[number]G</b>              | Replace the current command line with the contents of the oldest command line stored in the shell command history. With a number <i>number</i> , replace the current command line with the contents of command <i>number</i> in the history.                                                                                                                                                                                                                                                                                                                                                                      |
| 33264 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33265 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33266 | <b>/string&lt;newline&gt;</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33267 |                               | Move backward through the command history, searching for the specified <i>string</i> , beginning with the previous command line. If it is not found, the current command line shall be unchanged. If it is found in a previous line, this command shall behave equivalently to a set of <b>k</b> commands to reach that line. If <i>string</i> begins with '^', the characters after the '^' shall be matched only at the beginning of a line.                                                                                                                                                                    |
| 33268 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33269 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33270 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33271 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33272 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

33273 *?string*<newline>  
 33274 Move forward through the command history, searching for the specified string. If  
 33275 it is not found, the current command line shall be unchanged. If the string is found  
 33276 in the current command line, the current cursor position shall be moved to the  
 33277 beginning of that string. If it is found in the history, this command shall behave  
 33278 equivalently to a set of **j** commands to reach that line. If *string* begins with '^', the  
 33279 characters after the '^' shall be matched only at the beginning of a line.

33280 **n** Repeat the most recent / or ? command.

33281 **N** Repeat the most recent / or ? command, reversing the direction of the search.

33282 **EXIT STATUS**

33283 The following exit values shall be returned:

33284 0 The script to be executed consisted solely of zero or more blank lines or comments, or  
 33285 both.

33286 1-125 A non-interactive shell detected a syntax, redirection or variable assignment error.

33287 127 A specified *command\_file* could not be found by a non-interactive shell.

33288 Otherwise, the shell shall return the exit status of the last command it invoked or attempted to  
 33289 invoke (see also the *exit* utility in Section 2.15 (on page 2276)).

33290 **CONSEQUENCES OF ERRORS**

33291 See Section 2.8.1 (on page 2255).

33292 **APPLICATION USAGE**

33293 Standard input and standard error are the files that determine whether a shell is interactive  
 33294 when **-i** is not specified. For example:

33295 `sh > file`

33296 and:

33297 `sh 2> file`

33298 create interactive and non-interactive shells, respectively. Although both accept terminal input,  
 33299 the results of error conditions are different, as described in Section 2.8.1 (on page 2255); in the  
 33300 second example a redirection error encountered by a special built-in utility aborts the shell.

33301 A portable application must protect its first operand, if it starts with a plus sign, by preceding it  
 33302 with the "--" argument that denotes the end of the options.

33303 Applications should note that the standard *PATH* to the shell cannot be assumed to be either  
 33304 **/bin/sh** or **/usr/bin/sh**, and should be determined by interrogation of the *PATH* returned by  
 33305 *getconf PATH*, ensuring that the returned path name is an absolute path name and not a shell  
 33306 built in.

33307 For example, to determine the location of the standard *sh* utility:

33308 `command -v sh`

33309 On some systems this might return:

33310 `/usr/xpg4/bin/sh`

33311 Furthermore, on systems that support executable scripts (the "#!" construct), it is  
 33312 recommended that applications using executable scripts install them using *getconf -v* to  
 33313 determine the shell path name and update the "#!" script appropriately as it is being installed  
 33314 (for example, with *sed*). For example:

```

33315 #
33316 # Installation time script to install correct POSIX shell path name
33317 #
33318 # Get list of paths to check
33319 #
33320 Sifs=$IFS
33321 IFS=:
33322 set $(getconf PATH)
33323 IFS=$Sifs
33324 #
33325 # Check each path for 'sh'
33326 #
33327 for i in $@
33328 do
33329 if [-f ${i}/sh];
33330 then
33331 Pshell=${i}/sh
33332 fi
33333 done
33334 #
33335 # This is the list of scripts to update. They should be of the
33336 # form '${name}.source' and will be transformed to '${name}'.
33337 # Each script should begin:
33338 #
33339 # !INSTALLSHELLPATH -p
33340 #
33341 scripts="a b c"
33342 #
33343 # Transform each script
33344 #
33345 for i in ${scripts}
33346 do
33347 sed -e "s|INSTALLSHELLPATH|${Pshell}|" < ${i}.source > ${i}
33348 done

```

### 33349 EXAMPLES

- 33350 1. Execute a shell command from a string:
 

```
33351 sh -c "cat myfile"
```
- 33352 2. Execute a shell script from a file in the current directory:
 

```
33353 sh my_shell_cmds
```

### 33354 RATIONALE

33355 The *sh* utility and the *set* special built-in utility share a common set of options.

33356 The KornShell ignores the contents of *IFS* upon entry to the script. A conforming application cannot rely on importing *IFS*. One justification for this, beyond security considerations, is to assist possible future shell compilers. Allowing *IFS* to be imported from the environment prevents many optimizations that might otherwise be performed via dataflow analysis of the script itself.

33361 The text in the STDIN section about non-blocking reads concerns an instance of *sh* that has been invoked, probably by a C-language program, with standard input that has been opened using

33363 the O\_NONBLOCK flag; see *open()* in the System Interfaces volume of IEEE Std. 1003.1-200x. If  
33364 the shell did not reset this flag, it would immediately terminate because no input data would be  
33365 available yet and that would be considered the same as end-of-file.

33366 The options associated with a *restricted shell* (command name *rsh* and the *-r* option) were  
33367 excluded because the standard developers considered that the implied level of security could  
33368 not be achieved and they did not want to raise false expectations.

33369 On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the  
33370 name *-i*. When it is called by a sequence such as

33371 `sh -`

33372 or by:

33373 `#! usr/bin/sh -`

33374 the historical systems have assumed that no option letters follow. Thus, this volume of  
33375 IEEE Std. 1003.1-200x allows the single hyphen to mark the end of the options, in addition to the  
33376 use of the regular `"—"` argument, because it was considered that the older practice was so  
33377 pervasive. An alternative approach is taken by the KornShell, where real and effective  
33378 user/group IDs must match for an interactive shell; this behavior is specifically allowed by this  
33379 volume of IEEE Std. 1003.1-200x.

33380 **Note:** There are other problems with set-user-ID scripts that the two approaches described  
33381 here do not resolve.

33382 The default messages for the various *MAIL*-related messages are unspecified because they vary  
33383 across implementations. Typical messages are:

33384 `"you have mail\n"`

33385 or:

33386 `"you have new mail\n"`

33387 It is important that the descriptions of command line editing refer to the same shell as that in  
33388 IEEE Std. 1003.1-200x so that interactive users can also be application programmers without  
33389 having to deal with programmatic differences in their two environments. It is also essential that  
33390 the utility name *sh* be specified because this explicit utility name is too firmly rooted in historical  
33391 practice of application programs for it to change.

33392 Consideration was given to mandating a diagnostic message when attempting to set *vi*-mode on  
33393 terminals that do not support command line editing. However, it is not historical practice for the  
33394 shell to be cognizant of all terminal types and thus be able to detect inappropriate terminals in  
33395 all cases. Implementations are encouraged to supply diagnostics in this case whenever possible,  
33396 rather than leaving the user in a state where editing commands work incorrectly.

33397 In early proposals, the KornShell-derived *emacs* mode of command line editing was included,  
33398 even though the *emacs* editor itself was not. The community of *emacs* proponents was adamant  
33399 that the full *emacs* editor not be included in this volume of IEEE Std. 1003.1-200x because they  
33400 were concerned that an attempt to standardize this very powerful environment would  
33401 encourage vendors to ship versions conforming strictly to this volume of IEEE Std. 1003.1-200x,  
33402 but lacking the extensibility required by the community. The author of the original *emacs*  
33403 program also expressed his desire to omit the program. Furthermore, there were a number of  
33404 historical systems that did not include *emacs*, or included it without supporting it, but there were  
33405 very few that did not include and support *vi*. The shell *emacs* command line editing mode was  
33406 finally omitted from this volume of IEEE Std. 1003.1-200x because it became apparent that the  
33407 KornShell version and the editor being distributed with the GNU system had diverged in some



33408 respects. The author of *emacs* requested that the POSIX *emacs* mode either be deleted or have a  
 33409 significant number of unspecified conditions. Although the KornShell author agreed to consider  
 33410 changes to bring the shell into alignment, the standard developers decided to defer specification  
 33411 at this time, rather than attempting to agree on a specific subset of *emacs* late within the  
 33412 development of this volume of IEEE Std. 1003.1-200x. It is assumed that the *emacs* and KornShell  
 33413 developers will converge on a definition acceptable to both groups, and this may be used as a  
 33414 model for a future version of this volume of IEEE Std. 1003.1-200x. In the interim,  
 33415 implementations are free to offer additional command line editing modes based on the exact  
 33416 models of editors their users are most comfortable with.

33417 Early proposals had the following list entry in **vi Line Editing Insert Mode** (on page 3065):

33418 \ If followed by the *erase* or *kill* character, that character shall be inserted into the input line.  
 33419 Otherwise, the backslash itself shall be inserted into the input line.

33420 However, this is not actually a feature of *sh* command line editing insert mode, but one of some  
 33421 historical terminal line drivers. Some conforming implementations continue to do this when the  
 33422 *stty ixtexten* flag is set.

#### 33423 FUTURE DIRECTIONS

33424 None.

#### 33425 SEE ALSO

33426 *cd*, *echo*, *pwd*, *test*, *umask*, the System Interfaces volume of IEEE Std. 1003.1-200x, *dup()*, *exec*,  
 33427 *exit()*, *fork()*, *pipe()*, *signal()*, *system()*, *ulimit()*, *umask()*, *wait()*

#### 33428 CHANGE HISTORY

33429 First released in Issue 2.

#### 33430 Issue 4

33431 Aligned with the ISO/IEC 9945-2:1993 standard.

33432 Description of the shell command language and special built-ins moved to Chapter 2 (on page  
 33433 2235).

#### 33434 Issue 5

33435 FUTURE DIRECTIONS section added.

33436 Text is added to the DESCRIPTION for the Large File Summit proposal.

#### 33437 Issue 6

33438 The Open Group corrigenda item U029/2 has been applied, correcting the second SYNOPSIS.

33439 The Open Group corrigenda item U027/3 has been applied, correcting a typographical error.

33440 The following new requirements on POSIX implementations derive from alignment with the  
 33441 Single UNIX Specification:

33442 • The option letters derived from the set special built-in are also accepted with a leading plus  
 33443 sign ('+').

33444 • Large file extensions are added:

33445 — Path name expansion does not fail due to the size of a file.

33446 — Shell input and output redirections have an implementation-defined offset maximum  
 33447 that is established in the open file description.

33448 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
 33449 “directory referred to by the *HOME* environment variable”.

33450 Descriptions for the *ENV* and *PWD* environment variables are included to align with the  
33451 IEEE P1003.2b draft standard.  
33452 The normative text is reworded to avoid use of the term “must” for application requirements.

33453 **NAME**

33454           sleep — suspend execution for an interval

33455 **SYNOPSIS**33456           sleep *time*33457 **DESCRIPTION**33458           The *sleep* utility shall suspend execution for at least the integral number of seconds specified by  
33459           the *time* operand.33460 **OPTIONS**

33461           None.

33462 **OPERANDS**

33463           The following operand shall be supported:

33464           *time*           A non-negative decimal integer specifying the number of seconds for which to  
33465           suspend execution.33466 **STDIN**

33467           Not used.

33468 **INPUT FILES**

33469           None.

33470 **ENVIRONMENT VARIABLES**33471           The following environment variables shall affect the execution of *sleep*:33472           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
33473           If *LANG* is unset or null, the corresponding value from the implementation-  
33474           defined default locale shall be used. If any of the internationalization variables  
33475           contains an invalid setting, the utility shall behave as if none of the variables had  
33476           been defined.33477           *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
33478           internationalization variables.33479           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
33480           characters (for example, single-byte as opposed to multi-byte characters in  
33481           arguments).33482           *LC\_MESSAGES*33483           Determine the locale that should be used to affect the format and contents of  
33484           diagnostic messages written to standard error.33485 *XSI*           *NLS\_PATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.33486 **ASYNCHRONOUS EVENTS**33487           If the *sleep* utility receives a SIGALRM signal, one of the following actions shall be taken:

- 33488           1. Terminate normally with a zero exit status.
- 
- 33489           2. Effectively ignore the signal.
- 
- 33490           3. Provide the default behavior for signals described in the ASYNCHRONOUS EVENTS
- 
- 33491           section of Section 1.11 (on page 2224). This could include terminating with a non-zero exit
- 
- 33492           status.

33493           The *sleep* utility shall take the standard action for all other signals.

33494 **STDOUT**

33495 Not used.

33496 **STDERR**

33497 Used only for diagnostic messages.

33498 **OUTPUT FILES**

33499 None.

33500 **EXTENDED DESCRIPTION**

33501 None.

33502 **EXIT STATUS**

33503 The following exit values shall be returned:

33504 0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal  
33505 was received. See the ASYNCHRONOUS EVENTS section.

33506 &gt;0 An error occurred.

33507 **CONSEQUENCES OF ERRORS**

33508 Default.

33509 **APPLICATION USAGE**

33510 None.

33511 **EXAMPLES**33512 The *sleep* utility can be used to execute a command after a certain amount of time, as in:33513 (*sleep* 105; *command*) &

33514 or to execute a command every so often, as in:

33515 while true  
33516 do  
33517 *command*  
33518 *sleep* 37  
33519 done33520 **RATIONALE**33521 The exit status is allowed to be zero when *sleep* is interrupted by the SIGALRM signal because  
33522 most implementations of this utility rely on the arrival of that signal to notify them that the  
33523 requested finishing time has been successfully attained. Such implementations thus do not  
33524 distinguish this situation from the successful completion case. Other implementations are  
33525 allowed to catch the signal and go back to sleep until the requested time expires or to provide  
33526 the normal signal termination procedures.33527 As with all other utilities that take integral operands and do not specify subranges of allowed  
33528 values, *sleep* is required by this volume of IEEE Std. 1003.1-200x to deal with *time* requests of up  
33529 to 2 147 483 647 seconds. This may mean that some implementations have to make multiple calls  
33530 to the delay mechanism of the underlying operating system if its argument range is less than  
33531 this.33532 **FUTURE DIRECTIONS**

33533 None.

33534 **SEE ALSO**33535 *wait*, the System Interfaces volume of IEEE Std. 1003.1-200x, *alarm()*, *sleep()*

33536 **CHANGE HISTORY**

33537 First released in Issue 2.

33538 **Issue 4**

33539 Aligned with the ISO/IEC 9945-2:1993 standard.

## 33540 NAME

33541 sort — sort, merge, or sequence check text files

## 33542 SYNOPSIS

33543 sort [-m][-o *output*][-bdfinru][-t *char*][-k *keydef*]... [*file*...]33544 sort -c [-bdfinru][-t *char*][-k *keydef*]... *file*

## 33545 DESCRIPTION

33546 The *sort* utility shall perform one of the following functions:

- 33547 1. Sort lines of all the named files together and write the result to the specified output.
- 33548 2. Merge lines of all the named (presorted) files together and write the result to the specified  
33549 output.
- 33550 3. Check that a single input file is correctly presorted.

33551 Comparisons shall be based on one or more sort keys extracted from each line of input (or, if no  
33552 sort keys are specified, the entire line up to, but not including, the terminating <newline>  
33553 character), and shall be performed using the collating sequence of the current locale.

## 33554 OPTIONS

33555 The *sort* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
33556 12.2, Utility Syntax Guidelines, and the **-k** *keydef* option should follow the **-b**, **-d**, **-f**, **-i**, **-n**, and  
33557 **-r** options.

33558 The following options shall be supported:

- 33559 **-c** Check that the single input file is ordered as specified by the arguments and the  
33560 collating sequence of the current locale. No output shall be produced; only the exit  
33561 code shall be affected.
- 33562 **-m** Merge only; the input file shall be assumed to be already sorted.
- 33563 **-o** *output* Specify the name of an output file to be used instead of the standard output. This  
33564 file can be the same as one of the input *files*.
- 33565 **-u** Unique: suppress all but one in each set of lines having equal keys. If used with  
33566 the **-c** option, check that there are no lines with duplicate keys, in addition to  
33567 checking that the input file is sorted.

33568 The following options shall override the default ordering rules. When ordering options appear  
33569 independent of any key field specifications, the requested field ordering rules shall be applied  
33570 globally to all sort keys. When attached to a specific key (see **-k**), the specified ordering options  
33571 shall override all global ordering options for that key.

- 33572 **-d** Specify that only <blank> characters and alphanumeric characters, according to  
33573 the current setting of *LC\_CTYPE*, shall be significant in comparisons. The behavior  
33574 is undefined for a sort key to which **-i** or **-n** also applies.
- 33575 **-f** Consider all lowercase characters that have uppercase equivalents, according to  
33576 the current setting of *LC\_CTYPE*, to be the uppercase equivalent for the purposes  
33577 of comparison.
- 33578 **-i** Ignore all characters that are non-printable, according to the current setting of  
33579 *LC\_CTYPE*.
- 33580 **-n** Restrict the sort key to an initial numeric string, consisting of optional <blank>  
33581 characters, optional minus sign, and zero or more digits with an optional radix  
33582 character and thousands separators (as defined in the current locale), which shall

- 33583 be sorted by arithmetic value. An empty digit string shall be treated as zero.  
 33584 Leading zeros and signs on zeros shall not affect ordering.
- 33585 **-r** Reverse the sense of comparisons.
- 33586 The treatment of field separators can be altered using the options:
- 33587 **-b** Ignore leading <blank> characters when determining the starting and ending  
 33588 positions of a restricted sort key. If the **-b** option is specified before the first **-k**  
 33589 option, it shall be applied to all **-k** options. Otherwise, the **-b** option can be  
 33590 attached independently to each **-k** *field\_start* or *field\_end* option-argument (see  
 33591 below).
- 33592 **-t char** Use *char* as the field separator character; *char* shall not be considered to be part of a  
 33593 field (although it can be included in a sort key). Each occurrence of *char* shall be  
 33594 significant (for example, <*char*><*char*> delimits an empty field). If **-t** is not  
 33595 specified, <blank> characters shall be used as default field separators; each  
 33596 maximal non-empty sequence of <blank> characters that follows a non-<blank>  
 33597 character shall be a field separator.
- 33598 Sort keys can be specified using the options:
- 33599 **-k keydef** The *keydef* argument is a restricted sort key field definition. The format of this  
 33600 definition is:
- 33601 *field\_start*[*type*][, *field\_end*[*type*]]
- 33602 where *field\_start* and *field\_end* define a key field restricted to a portion of the line  
 33603 (see the EXTENDED DESCRIPTION section), and *type* is a modifier from the list of  
 33604 characters 'b', 'd', 'f', 'i', 'n', 'r'. The 'b' modifier shall behave like the  
 33605 **-b** option, but applies only to the *field\_start* or *field\_end* to which it is attached. The  
 33606 other modifiers shall behave like the corresponding options, but shall apply only  
 33607 to the key field to which they are attached; they shall have this effect if specified  
 33608 with *field\_start*, *field\_end*, or both. If any modifier is attached to a *field\_start* or to a  
 33609 *field\_end*, no option shall apply to either. Implementations shall support at least  
 33610 nine occurrences of the **-k** option, which shall be significant in command line  
 33611 order. If no **-k** option is specified, a default sort key of the entire line shall be used.
- 33612 When there are multiple key fields, later keys shall be compared only after all  
 33613 earlier keys compare equal. Except when the **-u** option is specified, lines that  
 33614 otherwise compare equal shall be ordered as if none of the options **-d**, **-f**, **-i**, **-n**, or  
 33615 **-k** were present (but with **-r** still in effect, if it was specified) and with all bytes in  
 33616 the lines significant to the comparison. The order in which lines that still compare  
 33617 equal are written is unspecified.
- 33618 **OPERANDS**
- 33619 The following operand shall be supported:
- 33620 *file* A path name of a file to be sorted, merged, or checked. If no *file* operands are  
 33621 specified, or if a *file* operand is '-', the standard input shall be used.
- 33622 **STDIN**
- 33623 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
 33624 See the INPUT FILES section.

33625 **INPUT FILES**

33626 The input files shall be text files, except that the *sort* utility shall add a <newline> character to  
 33627 the end of a file ending with an incomplete last line.

33628 **ENVIRONMENT VARIABLES**

33629 The following environment variables shall affect the execution of *sort*:

33630 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 33631 If *LANG* is unset or null, the corresponding value from the implementation-  
 33632 defined default locale shall be used. If any of the internationalization variables  
 33633 contains an invalid setting, the utility shall behave as if none of the variables had  
 33634 been defined.

33635 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 33636 internationalization variables.

33637 *LC\_COLLATE*

33638 Determine the locale for ordering rules.

33639 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 33640 characters (for example, single-byte as opposed to multi-byte characters in  
 33641 arguments and input files) and the behavior of character classification for the *-b*,  
 33642 *-d*, *-f*, *-i*, and *-n* options.

33643 *LC\_MESSAGES*

33644 Determine the locale that should be used to affect the format and contents of  
 33645 diagnostic messages written to standard error.

33646 *LC\_NUMERIC*

33647 Determine the locale for the definition of the radix character and thousands  
 33648 separator for the *-n* option.

33649 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

33650 **ASYNCHRONOUS EVENTS**

33651 Default.

33652 **STDOUT**

33653 Unless the *-o* or *-c* options are in effect, the standard output shall contain the sorted input.

33654 **STDERR**

33655 Used for diagnostic messages. A warning message about correcting an incomplete last line of an  
 33656 input file may be generated, but need not affect the final exit status.

33657 **OUTPUT FILES**

33658 If the *-o* option is in effect, the sorted input shall be written to the file *output*.

33659 **EXTENDED DESCRIPTION**

33660 The notation:

33661 *-k field\_start[type][,field\_end[type]]*

33662 shall define a key field that begins at *field\_start* and ends at *field\_end* inclusive, unless *field\_start*  
 33663 falls beyond the end of the line or after *field\_end*, in which case the key field is empty. A missing  
 33664 *field\_end* shall mean the last character of the line.

33665 A field comprises a maximal sequence of non-separating characters and, in the absence of option  
 33666 *-t*, any preceding field separator.

33667 The *field\_start* portion of the *keydef* option-argument shall have the form:



33668 *field\_number*[*.first\_character*]

33669 Fields and characters within fields shall be numbered starting with 1. The *field\_number* and  
33670 *first\_character* pieces, interpreted as positive decimal integers, shall specify the first character to  
33671 be used as part of a sort key. If *.first\_character* is omitted, it shall refer to the first character of the  
33672 field.

33673 The *field\_end* portion of the *keydef* option-argument shall have the form:

33674 *field\_number*[*.last\_character*]

33675 The *field\_number* shall be as described above for *field\_start*. The *last\_character* piece, interpreted  
33676 as a non-negative decimal integer, shall specify the last character to be used as part of the sort  
33677 key. If *last\_character* evaluates to zero or *.last\_character* is omitted, it shall refer to the last  
33678 character of the field specified by *field\_number*.

33679 If the **-b** option or **b** type modifier is in effect, characters within a field shall be counted from the  
33680 first non-<blank> character in the field. (This shall apply separately to *first\_character* and  
33681 *last\_character*.)

### 33682 EXIT STATUS

33683 The following exit values shall be returned:

33684 0 All input files were output successfully, or **-c** was specified and the input file was correctly  
33685 sorted.

33686 1 Under the **-c** option, the file was not ordered as specified, or if the **-c** and **-u** options were  
33687 both specified, two input lines were found with equal keys.

33688 >1 An error occurred.

### 33689 CONSEQUENCES OF ERRORS

33690 Default.

### 33691 APPLICATION USAGE

33692 The default value for **-t**, <blank> character, has different properties from, for example,  
33693 **-t"<space>"**. If a line contains:

33694 <space><space>foo

33695 the following treatment would occur with default separation as opposed to specifically selecting  
33696 a <space> character:

33697

| Field | Default           | <b>-t "&lt;space&gt;"</b> |
|-------|-------------------|---------------------------|
| 1     | <space><space>foo | <i>empty</i>              |
| 2     | <i>empty</i>      | <i>empty</i>              |
| 3     | <i>empty</i>      | foo                       |

33698

33699

33700

33701 The leading field separator itself is included in a field when **-t** is not used. For example, this  
33702 command returns an exit status of zero, meaning the input was already sorted:

33703 `sort -c -k 2 <<eof`

33704 `y<tab>b`

33705 `x<space>a`

33706 `eof`

33707 (assuming that a <tab> character precedes the <space> character in the current collating  
33708 sequence). The field separator is not included in a field when it is explicitly set via **-t**. This is  
33709 historical practice and allows usage such as:

```

33710 sort -t "|" -k 2n <<eof
33711 Atlanta|425022|Georgia
33712 Birmingham|284413|Alabama
33713 Columbia|100385|South Carolina
33714 eof

```

33715 where the second field can be correctly sorted numerically without regard to the non-numeric  
33716 field separator.

33717 The wording in the OPTIONS section clarifies that the **-b**, **-d**, **-f**, **-i**, **-n**, and **-r** options have to  
33718 come before the first sort key specified if they are intended to apply to all specified keys. The  
33719 way it is described in this volume of IEEE Std. 1003.1-200x matches historical practice, not  
33720 historical documentation. The results are unspecified if these options are specified after a **-k**  
33721 option.

33722 The **-f** option might not work as expected in locales where there is not a one-to-one mapping  
33723 between an uppercase and a lowercase letter.

### 33724 EXAMPLES

33725 1. The following command sorts the contents of **infile** with the second field as the sort key:

```
33726 sort -k 2,2 infile
```

33727 2. The following command sorts, in reverse order, the contents of **infile1** and **infile2**, placing  
33728 the output in **outfile** and using the second character of the second field as the sort key  
33729 (assuming that the first character of the second field is the field separator):

```
33730 sort -r -o outfile -k 2.2,2.2 infile1 infile2
```

33731 3. The following command sorts the contents of **infile1** and **infile2** using the second non-  
33732 <blank> character of the second field as the sort key:

```
33733 sort -k 2.2b,2.2b infile1 infile2
```

33734 4. The following command prints the System V password file (user database) sorted by the  
33735 numeric user ID (the third colon-separated field):

```
33736 sort -t : -k 3,3n /etc/passwd
```

33737 5. The following command prints the lines of the already sorted file **infile**, suppressing all  
33738 but one occurrence of lines having the same third field:

```
33739 sort -um -k 3.1,3.0 infile
```

### 33740 RATIONALE

33741 Examples in some historical documentation state that options **-um** with one input file keep the  
33742 first in each set of lines with equal keys. This behavior was deemed to be an implementation  
33743 artifact and was not standardized.

33744 The **-z** option was omitted; it is not standard practice on most systems and is inconsistent with  
33745 using *sort* to sort several files individually and then merge them together. The text concerning **-z**  
33746 in historical documentation appeared to require implementations to determine the proper buffer  
33747 length during the sort phase of operation, but not during the merge.

33748 The **-y** option was omitted because of non-portability. The **-M** option, present in System V, was  
33749 omitted because of non-portability in international usage.

33750 An undocumented **-T** option exists in some implementations. It is used to specify a directory for  
33751 intermediate files. Implementations are encouraged to support the use of the *TMPDIR*  
33752 environment variable instead of adding an option to support this functionality.

- 33753 The **-k** option was added to satisfy two objections. First, the zero-based counting used by *sort* is  
33754 not consistent with other utility conventions. Second, it did not meet syntax guideline  
33755 requirements.
- 33756 Historical documentation indicates that “setting **-n** implies **-b**”. The description of **-n** already  
33757 states that optional leading <blank>s are tolerated in doing the comparison. If **-b** is enabled,  
33758 rather than implied, by **-n**, this has unusual side effects. When a character offset is used in a  
33759 column of numbers (for example, to sort modulo 100), that offset is measured relative to the  
33760 most significant digit, not to the column. Based upon a recommendation from the author of the  
33761 original *sort* utility, the **-b** implication has been omitted from this volume of  
33762 IEEE Std. 1003.1-200x, and an application wishing to achieve the previously mentioned side  
33763 effects has to code the **-b** flag explicitly.
- 33764 **FUTURE DIRECTIONS**
- 33765 None.
- 33766 **SEE ALSO**
- 33767 *comm, join, uniq*, the System Interfaces volume of IEEE Std. 1003.1-200x, *toupper()*
- 33768 **CHANGE HISTORY**
- 33769 First released in Issue 2.
- 33770 **Issue 4**
- 33771 Aligned with the ISO/IEC 9945-2:1993 standard.
- 33772 **Issue 6**
- 33773 IEEE PASC Interpretation 1003.2 #174 is applied, updating the DESCRIPTION of comparisons.
- 33774 IEEE PASC Interpretation 1003.2 #168 is applied.

## 33775 NAME

33776 split — split files into pieces

## 33777 SYNOPSIS

33778 UP `split [-l line_count][-a suffix_length][file[name]]`33779 `split -b n[k|m][-a suffix_length][file[name]]`

33780

## 33781 DESCRIPTION

33782 The *split* utility shall read an input file and write one or more output files. The default size of  
 33783 each output file shall be 1 000 lines. The size of the output files can be modified by specification  
 33784 of the **-b** or **-l** options. Each output file shall be created with a unique suffix. The suffix shall  
 33785 consist of exactly *suffix\_length* lowercase letters from the POSIX locale. The letters of the suffix  
 33786 shall be used as if they were a base-26 digit system, with the first suffix to be created consisting  
 33787 of all 'a' characters, the second with a 'b' replacing the last 'a', and so on, until a name of all  
 33788 'z' characters is created. By default, the names of the output files shall be 'x', followed by a  
 33789 two-character suffix from the character set as described above, starting with "aa", "ab", "ac",  
 33790 and so on, and continuing until the suffix "zz", for a maximum of 676 files.

33791 If the number of files required exceeds the maximum allowed by the suffix length provided,  
 33792 such that the last allowable file would be larger than the requested size, the *split* utility shall fail  
 33793 after creating the last file with a valid suffix; *split* shall not delete the files it created with valid  
 33794 suffixes. If the file limit is not exceeded, the last file created shall contain the remainder of the  
 33795 input file, and may be smaller than the requested size.

## 33796 OPTIONS

33797 The *split* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 33798 12.2, Utility Syntax Guidelines.

33799 The following options shall be supported:

33800 **-a *suffix\_length***

33801 Use *suffix\_length* letters to form the suffix portion of the file names of the split file.  
 33802 If **-a** is not specified, the default suffix length shall be two. If the sum of the *name*  
 33803 operand and the *suffix\_length* option-argument would create a file name exceeding  
 33804 {NAME\_MAX} bytes, an error shall result; *split* shall exit with a diagnostic  
 33805 message and no files shall be created.

33806 **-b *n*** Split a file into pieces *n* bytes in size.

33807 **-b *nk*** Split a file into pieces *n*\*1024 bytes in size.

33808 **-b *nm*** Split a file into pieces *n*\*1 048 576 bytes in size.

33809 **-l *line\_count*** Specify the number of lines in each resulting file piece. The *line\_count* argument is  
 33810 an unsigned decimal integer. The default is 1 000. If the input does not end with a  
 33811 <newline> character, the partial line shall be included in the last output file.

## 33812 OPERANDS

33813 The following operands shall be supported:

33814 *file* The path name of the ordinary file to be split. If no input file is given or *file* is '-',  
 33815 the standard input shall be used.

33816 *name* The prefix to be used for each of the files resulting from the split operation. If no  
 33817 *name* argument is given, 'x' shall be used as the prefix of the output files. The  
 33818 combined length of the basename of *prefix* and *suffix\_length* cannot exceed  
 33819 {NAME\_MAX} bytes. See the OPTIONS section.

33820 **STDIN**

33821 See the INPUT FILES section.

33822 **INPUT FILES**

33823 Any file can be used as input.

33824 **ENVIRONMENT VARIABLES**33825 The following environment variables shall affect the execution of *split*:

33826 *LANG* Provide a default value for the internationalization variables that are unset or null.  
33827 If *LANG* is unset or null, the corresponding value from the implementation-  
33828 defined default locale shall be used. If any of the internationalization variables  
33829 contains an invalid setting, the utility shall behave as if none of the variables had  
33830 been defined.

33831 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
33832 internationalization variables.

33833 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
33834 characters (for example, single-byte as opposed to multi-byte characters in  
33835 arguments and input files).

33836 *LC\_MESSAGES*

33837 Determine the locale that should be used to affect the format and contents of  
33838 diagnostic messages written to standard error.

33839 *NSLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

33840 **ASYNCHRONOUS EVENTS**

33841 Default.

33842 **STDOUT**

33843 Not used.

33844 **STDERR**

33845 Used only for diagnostic messages.

33846 **OUTPUT FILES**

33847 The output files contain portions of the original input file; otherwise, unchanged.

33848 **EXTENDED DESCRIPTION**

33849 None.

33850 **EXIT STATUS**

33851 The following exit values shall be returned:

33852 0 Successful completion.

33853 &gt;0 An error occurred.

33854 **CONSEQUENCES OF ERRORS**

33855 Default.

33856 **APPLICATION USAGE**

33857 None.

33858 **EXAMPLES**33859 In the following examples **foo** is a text file that contains 5 000 lines.33860 1. Create five files, **xaa**, **xab**, **xac**, **xad**, and **xae**:33861 `split foo`33862 2. Create five files, but the suffixed portion of the created files consists of three letters, **xaaa**,  
33863 **xaab**, **xaac**, **xaad**, and **xaae**:33864 `split -a 3 foo`33865 3. Create three files with four-letter suffixes and a supplied prefix, **bar\_aaaa**, **bar\_aaab**, and  
33866 **bar\_aaac**:33867 `split -a 4 -l 2000 foo bar_`33868 4. Create as many files as are necessary to contain at most 20\*1 024 bytes, each with the  
33869 default prefix of **x** and a five-letter suffix:33870 `split -a 5 -b 20k foo`33871 **RATIONALE**33872 The **-b** option was added to provide a mechanism for splitting files other than by lines. While  
33873 most uses of the **-b** option are for transmitting files over networks, some believed it would have  
33874 additional uses.33875 The **-a** option was added to overcome the limitation of being able to create only 676 files.33876 Consideration was given to deleting this utility, using the rationale that the function provided  
33877 by this utility is available via the *csplit* utility (see *csplit* (on page 2492)). Upon reconsideration of  
33878 the purpose of the User Portability Extension, it was decided to retain both this utility and the  
33879 *csplit* utility because users use both utilities and have historical expectations of their behavior.  
33880 Furthermore, the splitting on byte boundaries in *split* cannot be duplicated with the historical  
33881 *csplit*.33882 The text “*split* shall not delete the files it created with valid suffixes” would normally be  
33883 assumed, but since the related utility, *csplit*, does delete files under some circumstances, the  
33884 historical behavior of *split* is made explicit to avoid misinterpretation.33885 **FUTURE DIRECTIONS**

33886 None.

33887 **SEE ALSO**33888 *csplit*33889 **CHANGE HISTORY**

33890 First released in Issue 2.

33891 **Issue 4**

33892 Aligned with the ISO/IEC 9945-2: 1993 standard.

33893 **Issue 6**

33894 This utility is now marked as part of the User Portability Utilities option.

33895 The APPLICATION USAGE section is added.

33896 The obsolescent SYNOPSIS is removed.

33897 **NAME**

33898 strings — find printable strings in files

33899 **SYNOPSIS**33900 UP strings [-a][-t *format*][-n *number*][*file...*]

33901

33902 **DESCRIPTION**

33903 The *strings* utility shall look for printable strings in regular files and shall write those strings to  
 33904 standard output. A printable string is any sequence of four (by default) or more printable  
 33905 characters terminated by a <newline> or NUL character. Additional implementation-defined  
 33906 strings may be written; see *localedef*.

33907 **OPTIONS**

33908 The *strings* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
 33909 12.2, Utility Syntax Guidelines.

33910 The following options shall be supported:

33911 **-a** Scan files in their entirety. If **-a** is not specified, it is implementation-defined what  
 33912 portion of each file is scanned for strings.

33913 **-n *number*** Specify the minimum string length, where the *number* argument is a positive  
 33914 decimal integer. The default shall be 4.

33915 **-t *format*** Write each string preceded by its byte offset from the start of the file. The format  
 33916 shall be dependent on the single character used as the *format* option-argument:

33917     d The offset shall be written in decimal.

33918     o The offset shall be written in octal.

33919     x The offset shall be written in hexadecimal.

33920 **OPERANDS**

33921 The following operand shall be supported:

33922 ***file*** A path name of a regular file to be used as input. If no *file* operand is specified, the  
 33923 *strings* utility shall read from the standard input.

33924 **STDIN**

33925 See the INPUT FILES section.

33926 **INPUT FILES**

33927 The input files named by the utility arguments or the standard input shall be regular files of any  
 33928 format.

33929 **ENVIRONMENT VARIABLES**33930 The following environment variables shall affect the execution of *strings*:

33931 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 33932 If **LANG** is unset or null, the corresponding value from the implementation-  
 33933 defined default locale shall be used. If any of the internationalization variables  
 33934 contains an invalid setting, the utility shall behave as if none of the variables had  
 33935 been defined.

33936 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 33937 internationalization variables.

33938 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 33939 characters (for example, single-byte as opposed to multi-byte characters in

- 33940 arguments and input files) and to identify printable strings.
- 33941 **LC\_MESSAGES**
- 33942 Determine the locale that should be used to affect the format and contents of
- 33943 diagnostic messages written to standard error.
- 33944 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 33945 **ASYNCHRONOUS EVENTS**
- 33946 Default.
- 33947 **STDOUT**
- 33948 Strings found shall be written to the standard output, one per line.
- 33949 When the **-t** option is not specified, the format of the output shall be:
- 33950 "%s", <string>
- 33951 With the **-t o** option, the format of the output shall be:
- 33952 "%o %s", <byte offset>, <string>
- 33953 With the **-t x** option, the format of the output shall be:
- 33954 "%x %s", <byte offset>, <string>
- 33955 With the **-t d** option, the format of the output shall be:
- 33956 "%d %s", <byte offset>, <string>
- 33957 **STDERR**
- 33958 Used only for diagnostic messages.
- 33959 **OUTPUT FILES**
- 33960 None.
- 33961 **EXTENDED DESCRIPTION**
- 33962 None.
- 33963 **EXIT STATUS**
- 33964 The following exit values shall be returned:
- 33965 0 Successful completion.
- 33966 >0 An error occurred.
- 33967 **CONSEQUENCES OF ERRORS**
- 33968 Default.
- 33969 **APPLICATION USAGE**
- 33970 By default the data area (as opposed to the text, "bss" or header areas) of a binary executable file
- 33971 is scanned. Implementations document which areas are scanned.
- 33972 Some historical implementations do not require NUL or <newline> character terminators for
- 33973 strings to permit those languages that do not use NUL as a string terminator to have their strings
- 33974 written.
- 33975 **EXAMPLES**
- 33976 None.



33977 **RATIONALE**

33978        Apart from rationalizing the option syntax and slight difficulties with object and executable  
33979        binary files, *strings* is specified to match historical practice closely. The **-a** and **-n** options were  
33980        introduced to replace the non-conforming **-** and *-number* options.

33981        The **-o** option historically means different things on different implementations. Some use it to  
33982        mean “*offset in decimal*”, while others use it as “*offset in octal*”. Instead of trying to decide which  
33983        way would be least objectionable, the **-t** option was added. It was originally named **-O** to mean  
33984        “*offset*”, but was changed to **-t** to be consistent with *od*.

33985        The ISO C standard function *isprint()* is restricted to a domain of **unsigned char**. This volume of  
33986        IEEE Std. 1003.1-200x requires implementations to write strings as defined by the current locale.

33987 **FUTURE DIRECTIONS**

33988        None.

33989 **SEE ALSO**

33990        *nm*

33991 **CHANGE HISTORY**

33992        First released in Issue 4.

33993 **Issue 6**

33994        This utility is now marked as part of the User Portability Utilities option.

33995        The obsolescent SYNOPSIS is removed.

33996        The normative text is reworded to avoid use of the term “*must*” for application requirements.

## 33997 NAME

33998 strip — remove unnecessary information from executable files (DEVELOPMENT)

## 33999 SYNOPSIS

34000 SD strip file...

34001

## 34002 DESCRIPTION

34003 The *strip* utility shall remove from executable files named by the *file* operands any information  
34004 the implementor deems unnecessary for execution of those files. The nature of that information  
34005 is unspecified. The effect of *strip* shall be similar to the use of the *-s* option to *cc*, *c99*, or *fort77*.

## 34006 OPTIONS

34007 None.

## 34008 OPERANDS

34009 The following operand shall be supported:

34010 *file* A path name referring to an executable file.

## 34011 STDIN

34012 Not used.

## 34013 INPUT FILES

34014 The input files shall be in the form of executable files successfully produced by any compiler  
34015 defined by this volume of IEEE Std. 1003.1-200x.

## 34016 ENVIRONMENT VARIABLES

34017 The following environment variables shall affect the execution of *strip*:

34018 *LANG* Provide a default value for the internationalization variables that are unset or null.  
34019 If *LANG* is unset or null, the corresponding value from the implementation-  
34020 defined default locale shall be used. If any of the internationalization variables  
34021 contains an invalid setting, the utility shall behave as if none of the variables had  
34022 been defined.

34023 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
34024 internationalization variables.

34025 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
34026 characters (for example, single-byte as opposed to multi-byte characters in  
34027 arguments).

34028 *LC\_MESSAGES*

34029 Determine the locale that should be used to affect the format and contents of  
34030 diagnostic messages written to standard error.

34031 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 34032 ASYNCHRONOUS EVENTS

34033 Default.

## 34034 STDOUT

34035 Not used.

## 34036 STDERR

34037 Used only for diagnostic messages.

**34038 OUTPUT FILES**

34039 The *strip* utility shall produce executable files of unspecified format.

**34040 EXTENDED DESCRIPTION**

34041 None.

**34042 EXIT STATUS**

34043 The following exit values shall be returned:

34044 0 Successful completion.

34045 >0 An error occurred.

**34046 CONSEQUENCES OF ERRORS**

34047 Default.

**34048 APPLICATION USAGE**

34049 None.

**34050 EXAMPLES**

34051 None.

**34052 RATIONALE**

34053 Historically, this utility has been used to remove the symbol table from an executable file. It was  
34054 included since it is known that the amount of symbolic information can amount to several  
34055 megabytes; the ability to remove it in a portable manner was deemed important, especially for  
34056 smaller systems.

34057 The behavior of *strip* is said to be the same as the `-s` option to a compiler. While the end result is  
34058 essentially the same, it is not required to be identical. The same effect can be achieved with  
34059 either `-s` during a compile or a *strip* on the final object file.

**34060 FUTURE DIRECTIONS**

34061 None.

**34062 SEE ALSO**

34063 *ar*, *c99*, *fort77*

**34064 CHANGE HISTORY**

34065 First released in Issue 2.

**34066 Issue 4**

34067 Aligned with the ISO/IEC 9945-2:1993 standard.

**34068 Issue 6**

34069 This utility is now marked as part of the Software Development Utilities option.

## 34070 NAME

34071 stty — set the options for a terminal

## 34072 SYNOPSIS

34073 stty [ -a | -g ]

34074 stty *operands*

## 34075 DESCRIPTION

34076 The *stty* utility shall set or report on terminal I/O characteristics for the device that is its  
34077 standard input. Without options or operands specified, it shall report the settings of certain  
34078 characteristics, usually those that differ from implementation-defined defaults. Otherwise, it  
34079 shall modify the terminal state according to the specified operands. Detailed information about  
34080 the modes listed in the first five groups below are described in the Base Definitions volume of  
34081 IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface. Operands in the Combination  
34082 Modes group (see **Combination Modes** (on page 3099)) are implemented using operands in the  
34083 previous groups. Some combinations of operands are mutually-exclusive on some terminal  
34084 types; the results of using such combinations are unspecified.

34085 Typical implementations of this utility require a communications line configured to use the  
34086 **termios** interface defined in the System Interfaces volume of IEEE Std. 1003.1-200x. On systems  
34087 where none of these lines are available, and on lines not currently configured to support the  
34088 **termios** interface, some of the operands need not affect terminal characteristics.

## 34089 OPTIONS

34090 The *stty* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section  
34091 12.2, Utility Syntax Guidelines.

34092 The following options shall be supported:

34093 **-a** Write to standard output all the current settings for the terminal.

34094 **-g** Write to standard output all the current settings in an unspecified form that can be  
34095 used as arguments to another invocation of the *stty* utility on the same system. The  
34096 form used shall not contain any characters that would require quoting to avoid  
34097 word expansion by the shell; see Section 2.6 (on page 2244).

## 34098 OPERANDS

34099 The following operands shall be supported to set the terminal characteristics.

## 34100 Control Modes

34101 **parenb** (**-parenb**) Enable (disable) parity generation and detection. This has the effect of setting  
34102 (not setting) PARENB in the **termios** *c\_flag* field, as defined in the Base  
34103 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
34104 Interface.

34105 **parodd** (**-parodd**) Select odd (even) parity. This shall have the effect of setting (not setting)  
34106 PARODD in the **termios** *c\_flag* field, as defined in the Base Definitions  
34107 volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.

34108 **cs5 cs6 cs7 cs8** Select character size, if possible. This shall have the effect of setting CS5, CS6,  
34109 CS7, and CS8, respectively, in the **termios** *c\_flag* field, as defined in the Base  
34110 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
34111 Interface.

34112 *number* Set terminal baud rate to the number given, if possible. If the baud rate is set  
34113 to zero, the modem control lines shall not be longer asserted. This shall have  
34114 the effect of setting the input and output **termios** baud rate values as defined

|       |                         |                                                                                                        |
|-------|-------------------------|--------------------------------------------------------------------------------------------------------|
| 34115 |                         | in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General                           |
| 34116 |                         | Terminal Interface.                                                                                    |
| 34117 | <b>ispeed number</b>    | Set terminal input baud rate to the number given, if possible. If the input baud                       |
| 34118 |                         | rate is set to zero, the input baud rate shall be specified by the value of the                        |
| 34119 |                         | output baud rate. This shall have the effect of setting the input <b>termios</b> baud                  |
| 34120 |                         | rate values as defined in the Base Definitions volume of IEEE Std. 1003.1-200x,                        |
| 34121 |                         | Chapter 11, General Terminal Interface.                                                                |
| 34122 | <b>ospeed number</b>    | Set terminal output baud rate to the number given, if possible. If the output                          |
| 34123 |                         | baud rate is set to zero, the modem control lines shall no longer be asserted.                         |
| 34124 |                         | This shall have the effect of setting the output <b>termios</b> baud rate values as                    |
| 34125 |                         | defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11,                           |
| 34126 |                         | General Terminal Interface.                                                                            |
| 34127 | <b>hupcl (-hupcl)</b>   | Stop asserting modem control lines (do not stop asserting modem control                                |
| 34128 |                         | lines) on last close. This shall have the effect of setting (not setting) HUPCL in                     |
| 34129 |                         | the <b>termios</b> <i>c_cflag</i> field, as defined in the Base Definitions volume of                  |
| 34130 |                         | IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                         |
| 34131 | <b>hup (-hup)</b>       | Same as <b>hupcl(-hupcl)</b> .                                                                         |
| 34132 | <b>cstopb (-cstopb)</b> | Use two (one) stop bits per character. This shall have the effect of setting (not                      |
| 34133 |                         | setting) CSTOPB in the <b>termios</b> <i>c_cflag</i> field, as defined in the Base Definitions         |
| 34134 |                         | volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                               |
| 34135 | <b>cread (-cread)</b>   | Enable (disable) the receiver. This shall have the effect of setting (not setting)                     |
| 34136 |                         | CREAD in the <b>termios</b> <i>c_cflag</i> field, as defined in the Base Definitions volume            |
| 34137 |                         | of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                      |
| 34138 | <b>local (-local)</b>   | Assume a line without (with) modem control. This shall have the effect of                              |
| 34139 |                         | setting (not setting) CLOCAL in the <b>termios</b> <i>c_cflag</i> field, as defined in the             |
| 34140 |                         | Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General                                  |
| 34141 |                         | Terminal Interface.                                                                                    |
| 34142 |                         | It is unspecified whether <i>stty</i> shall report an error if an attempt to set a Control Mode fails. |
| 34143 | <b>Input Modes</b>      |                                                                                                        |
| 34144 | <b>ignbrk (-ignbrk)</b> | Ignore (do not ignore) break on input. This shall have the effect of setting (not                      |
| 34145 |                         | setting) IGNBRK in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions         |
| 34146 |                         | volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                               |
| 34147 | <b>brkint (-brkint)</b> | Signal (do not signal) INTR on break. This shall have the effect of setting (not                       |
| 34148 |                         | setting) BRKINT in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions         |
| 34149 |                         | volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                               |
| 34150 | <b>ignpar (-ignpar)</b> | Ignore (do not ignore) bytes with parity errors. This shall have the effect of                         |
| 34151 |                         | setting (not setting) IGNPAR in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base        |
| 34152 |                         | Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal                              |
| 34153 |                         | Interface.                                                                                             |
| 34154 | <b>parmrk (-parmrk)</b> |                                                                                                        |
| 34155 |                         | Mark (do not mark) parity errors. This shall have the effect of setting (not                           |
| 34156 |                         | setting) PARMRK in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base                     |
| 34157 |                         | Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal                              |
| 34158 |                         | Interface.                                                                                             |

|           |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
|-----------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 34159     | <b>inpck</b> ( <b>-inpck</b> )   | Enable (disable) input parity checking. This shall have the effect of setting (not setting) INPCK in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                |
| 34160     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34161     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34162     | <b>istrip</b> ( <b>-istrip</b> ) | Strip (do not strip) input characters to seven bits. This shall have the effect of setting (not setting) ISTRIP in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                  |
| 34163     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34164     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34165     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34166     | <b>inlcr</b> ( <b>-inlcr</b> )   | Map (do not map) NL to CR on input. This shall have the effect of setting (not setting) INLCR in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                    |
| 34167     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34168     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34169     | <b>igncr</b> ( <b>-igncr</b> )   | Ignore (do not ignore) CR on input. This shall have the effect of setting (not setting) IGNCR in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                    |
| 34170     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34171     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34172     | <b>icrnl</b> ( <b>-icrnl</b> )   | Map (do not map) CR to NL on input. This shall have the effect of setting (not setting) ICRNL in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                    |
| 34173     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34174     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34175     | <b>ixon</b> ( <b>-ixon</b> )     | Enable (disable) START/STOP output control. Output from the system is stopped when the system receives STOP and started when the system receives START. This shall have the effect of setting (not setting) IXON in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface. |
| 34176     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34177     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34178     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34179     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34180 XSI | <b>ixany</b> ( <b>-ixany</b> )   | Allow any character to restart output. This shall have the effect of setting (not setting) IXANY in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                 |
| 34181     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34182     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34183     | <b>ixoff</b> ( <b>-ixoff</b> )   | Request that the system send (not send) STOP characters when the input queue is nearly full and START characters to resume data transmission. This shall have the effect of setting (not setting) IXOFF in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.          |
| 34184     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34185     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34186     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34187     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34188     | <b>Output Modes</b>              |                                                                                                                                                                                                                                                                                                                                                                          |
| 34189     | <b>opost</b> ( <b>-opost</b> )   | Post-process output (do not post-process output; ignore all other output modes). This shall have the effect of setting (not setting) OPOST in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                       |
| 34190     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34191     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34192     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34193 XSI | <b>ocrnl</b> ( <b>-ocrnl</b> )   | Map (do not map) CR to NL on output. This shall have the effect of setting (not setting) OCRNL in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                   |
| 34194     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34195     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34196     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34197     | <b>onocr</b> ( <b>-onocr</b> )   | Do not (do) output CR at column zero. This shall have the effect of setting (not setting) ONOCR in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                  |
| 34198     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34199     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34200     | <b>onlret</b> ( <b>-onlret</b> ) | The terminal newline key performs (does not perform) the CR function. This shall have the effect of setting (not setting) ONLRET in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                 |
| 34201     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34202     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |
| 34203     |                                  |                                                                                                                                                                                                                                                                                                                                                                          |

|       |                            |                                                                                                                                                                                                                                                                                                                                                                              |
|-------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 34204 | <b>ofill (-ofill)</b>      | Use fill characters (use timing) for delays. This shall have the effect of setting (not setting) OFILL in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                               |
| 34205 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34206 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34207 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34208 | <b>ofdel (-ofdel)</b>      | Fill characters are DELs (NULs). This shall have the effect of setting (not setting) OFDEL in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                                           |
| 34209 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34210 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34211 | <b>cr0 cr1 cr2 cr3</b>     | Select the style of delay for CRs. This shall have the effect of setting (not setting) CRDLY to CR1, CR2, CR3, or CR4, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                 |
| 34212 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34213 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34214 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34215 | <b>nl0 nl1</b>             | Select the style of delay for NL. This has the effect of setting (not setting) NLDLY to NL0 or NL1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                                    |
| 34216 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34217 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34218 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34219 | <b>tab0 tab1 tab2 tab3</b> | Select the style of delay for horizontal tabs. This shall have the effect of setting (not setting) TABDLY to TAB0, TAB1, TAB2, or TAB3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface. Note that TAB3 has the effect of expanding <tab>s to <space>s. |
| 34220 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34221 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34222 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34223 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34224 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34225 | <b>tabs (-tabs)</b>        |                                                                                                                                                                                                                                                                                                                                                                              |
| 34226 |                            | Synonym for <b>tab0 (tab3)</b> .                                                                                                                                                                                                                                                                                                                                             |
| 34227 | <b>bs0 bs1</b>             | Select the style of delay for backspaces. This shall have the effect of setting (not setting) BSDLY to BS0 or BS1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                     |
| 34228 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34229 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34230 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34231 | <b>ff0 ff1</b>             | Select the style of delay for form-feeds. This shall have the effect of setting (not setting) FFDLY to FF0 or FF1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                     |
| 34232 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34233 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34234 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34235 | <b>vt0 vt1</b>             | Select the style of delay for vertical-tabs. This shall have the effect of setting (not setting) VTDLY to VT0 or VT1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                  |
| 34236 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34237 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34238 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34239 | <b>Local Modes</b>         |                                                                                                                                                                                                                                                                                                                                                                              |
| 34240 | <b>isig (-isig)</b>        | Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SUSP. This shall have the effect of setting (not setting) ISIG in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                    |
| 34241 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34242 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34243 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34244 | <b>icanon (-icanon)</b>    | Enable (disable) canonical input (ERASE and KILL processing). This shall have the effect of setting (not setting) ICANON in the <b>termios</b> <i>c_iflag</i> field, as defined in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface.                                                                                             |
| 34245 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34246 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34247 |                            |                                                                                                                                                                                                                                                                                                                                                                              |
| 34248 | <b>ixten (-ixten)</b>      |                                                                                                                                                                                                                                                                                                                                                                              |
| 34249 |                            | Enable (disable) any implementation-defined special control characters not                                                                                                                                                                                                                                                                                                   |

- 34250 currently controlled by **icanon**, **isig**, **ixon**, or **ixoff**. This shall have the effect of  
 34251 setting (not setting) IEXTEN in the **termios** *c\_lflag* field, as defined in the Base  
 34252 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34253 Interface.
- 34254 **echo** (**-echo**) Echo back (do not echo back) every character typed. This shall have the effect  
 34255 of setting (not setting) ECHO in the **termios** *c\_lflag* field, as defined in the Base  
 34256 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34257 Interface.
- 34258 **echoe** (**-echoe**) The ERASE character visually erases (does not erase) the last character in the  
 34259 current line from the display, if possible. This shall have the effect of setting  
 34260 (not setting) ECHOE in the **termios** *c\_lflag* field, as defined in the Base  
 34261 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34262 Interface.
- 34263 **echok** (**-echok**) Echo (do not echo) NL after KILL character. This shall have the effect of  
 34264 setting (not setting) ECHOK in the **termios** *c\_lflag* field, as defined in the Base  
 34265 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34266 Interface.
- 34267 **echonl** (**-echonl**) Echo (do not echo) NL, even if **echo** is disabled. This shall have the effect of  
 34268 setting (not setting) ECHONL in the **termios** *c\_lflag* field, as defined in the  
 34269 Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General  
 34270 Terminal Interface.
- 34271 **noflsh** (**-noflsh**) Disable (enable) flush after INTR, QUIT, SUSP. This shall have the effect of  
 34272 setting (not setting) NOFLSH in the **termios** *c\_lflag* field, as defined in the Base  
 34273 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34274 Interface.
- 34275 **tostop** (**-tostop**) Send SIGTTOU for background output. This shall have the effect of setting  
 34276 (not setting) TOSTOP in the **termios** *c\_lflag* field, as defined in the Base  
 34277 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal  
 34278 Interface.

### 34279 **Special Control Character Assignments**

- 34280 *<control>-character string*
- 34281 Set *<control>-character* to *string*. If *<control>-character* is one of the character sequences in  
 34282 the first column of the following table, the corresponding Base Definitions volume of  
 34283 IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface control character from the  
 34284 second column shall be recognized. This has the effect of setting the corresponding element  
 34285 of the **termios** *c\_cc* array (see the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter  
 34286 13, Headers, **<termios.h>**).



34287

Table 4-19 Control Character Names in *stty*

34288

34289

34290

34291

34292

34293

34294

34295

34296

34297

| Control Character | c_cc Subscript | Description     |
|-------------------|----------------|-----------------|
| <b>eof</b>        | VEOF           | EOF character   |
| <b>eol</b>        | VEOL           | EOL character   |
| <b>erase</b>      | VERASE         | ERASE character |
| <b>intr</b>       | VINTR          | INTR character  |
| <b>kill</b>       | VKILL          | KILL character  |
| <b>quit</b>       | VQUIT          | QUIT character  |
| <b>susp</b>       | VSUSP          | SUSP character  |
| <b>start</b>      | VSTART         | START character |
| <b>stop</b>       | VSTOP          | STOP character  |

34298

34299

34300

34301

34302

34303

34304

If *string* is a single character, the control character shall be set to that character. If *string* is the two-character sequence "^\_" or the string *undef*, the control character shall be set to `_POSIX_VDISABLE`, if it is in effect for the device; if `_POSIX_VDISABLE` is not in effect for the device, it shall be treated as an error. In the POSIX locale, if *string* is a two-character sequence beginning with circumflex ('^'), and the second character is one of those listed in the "^c" column of the following table, the control character shall be set to the corresponding character value in the Value column of the table.

34305

Table 4-20 Circumflex Control Characters in *stty*

34306

34307

34308

34309

34310

34311

34312

34313

34314

34315

34316

34317

| ^c   | Value | ^c   | Value | ^c   | Value |
|------|-------|------|-------|------|-------|
| a, A | <SOH> | l, L | <FF>  | w, W | <ETB> |
| b, B | <STX> | m, M | <CR>  | x, X | <CAN> |
| c, C | <ETX> | n, N | <SO>  | y, Y | <EM>  |
| d, D | <EOT> | o, O | <SI>  | z, Z | <SUB> |
| e, E | <ENQ> | p, P | <DLE> | [    | <ESC> |
| f, F | <ACK> | q, Q | <DC1> | \    | <FS>  |
| g, G | <BEL> | r, R | <DC2> | ]    | <GS>  |
| h, H | <BS>  | s, S | <DC3> | ^    | <RS>  |
| i, I | <HT>  | t, T | <DC4> | _    | <US>  |
| j, J | <LF>  | u, U | <NAK> | ?    | <DEL> |
| k, K | <VT>  | v, V | <SYN> |      |       |

34318

**min** *number*

34319

**time** *number*

34320

Set the value of **min** or **time** to *number*. MIN and TIME are used in non-canonical mode input processing (**icanon**).

34321

34322

**Combination Modes**

34323

*saved settings*

34324

Set the current terminal characteristics to the saved settings produced by the **-g** option.

34325

**evenp** or **parity**

34326

Enable **parenb** and **cs7**; disable **parodd**.

34327

**oddp**

34328

Enable **parenb**, **cs7**, and **parodd**.

34329

**-parity**, **-evenp**, or **-oddp**

34330

Disable **parenb**, and set **cs8**.

- 34331 XSI **raw** (**-raw** or **cooked**)  
 34332 Enable (disable) raw input and output. Raw mode shall be equivalent to setting:
- ```
34333 stty cs8 erase ^- kill ^- intr ^- \  

  34334 quit ^- eof ^- eol ^- -post -inpck
```
- 34335 **nl** (**-nl**)
 34336 Enable (disable) **icrnl**. In addition, **-nl** unsets **inlcr** and **igncr**.
- 34337 **ek** Reset ERASE and KILL characters back to system defaults.
- 34338 **sane** Reset all modes to some reasonable, unspecified, values.
- 34339 **STDIN**
 34340 Although no input is read from standard input, standard input is used to get the current
 34341 terminal I/O characteristics and to set new terminal I/O characteristics.
- 34342 **INPUT FILES**
 34343 None.
- 34344 **ENVIRONMENT VARIABLES**
 34345 The following environment variables shall affect the execution of *stty*:
- 34346 *LANG* Provide a default value for the internationalization variables that are unset or null.
 34347 If *LANG* is unset or null, the corresponding value from the implementation-
 34348 defined default locale shall be used. If any of the internationalization variables
 34349 contains an invalid setting, the utility shall behave as if none of the variables had
 34350 been defined.
- 34351 *LC_ALL* If set to a non-empty string value, override the values of all the other
 34352 internationalization variables.
- 34353 *LC_CTYPE* This variable determines the locale for the interpretation of sequences of bytes of
 34354 text data as characters (for example, single-byte as opposed to multi-byte
 34355 characters in arguments) and which characters are in the class **print**.
- 34356 *LC_MESSAGES*
 34357 Determine the locale that should be used to affect the format and contents of
 34358 diagnostic messages written to standard error.
- 34359 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 34360 **ASYNCHRONOUS EVENTS**
 34361 Default.
- 34362 **STDOUT**
 34363 If operands are specified, no output shall be produced.
- 34364 If the **-g** option is specified, *stty* shall write to standard output the current settings in a form that
 34365 can be used as arguments to another instance of *stty* on the same system.
- 34366 If the **-a** option is specified, all of the information as described in the OPERANDS section shall
 34367 be written to standard output. Unless otherwise specified, this information shall be written as
 34368 <space>-separated tokens in an unspecified format, on one or more lines, with an unspecified
 34369 number of tokens per line. Additional information may be written.
- 34370 If no options or operands are specified, an unspecified subset of the information written for the
 34371 **-a** option shall be written.
- 34372 If speed information is written as part of the default output, or if the **-a** option is specified and if
 34373 the terminal input speed and output speed are the same, the speed information shall be written

34374 as follows:

34375 "speed %d baud;", <speed>

34376 Otherwise, speeds shall be written as:

34377 "ispeed %d baud; ospeed %d baud;", <ispeed>, <ospeed>

34378 In locales other than the POSIX locale, the word **baud** may be changed to something more
34379 appropriate in those locales.

34380 If control characters are written as part of the default output, or if the **-a** option is specified,
34381 control characters shall be written as:

34382 "%s = %s;", <control-character name>, <value>

34383 where <value> is either the character, or some visual representation of the character if it is non-
34384 printable, or the string *undef* if the character is disabled.

34385 **STDERR**

34386 Used only for diagnostic messages.

34387 **OUTPUT FILES**

34388 None.

34389 **EXTENDED DESCRIPTION**

34390 None.

34391 **EXIT STATUS**

34392 The following exit values shall be returned:

34393 0 The terminal options were read or set successfully.

34394 >0 An error occurred.

34395 **CONSEQUENCES OF ERRORS**

34396 Default.

34397 **APPLICATION USAGE**

34398 The **-g** flag is designed to facilitate the saving and restoring of terminal state from the shell level.
34399 For example, a program may:

```
34400 saveterm="$(stty -g)"           # save terminal state
34401 stty (new settings)           # set new state
34402 ...                           # ...
34403 stty $saveterm                # restore terminal state
```

34404 Since the format is unspecified, the saved value is not portable across systems.

34405 Since the **-a** format is so loosely specified, scripts that save and restore terminal settings should
34406 use the **-g** option.

34407 **EXAMPLES**

34408 None.

34409 **RATIONALE**

34410 The original *stty* description was taken directly from System V and reflected the System V
34411 terminal driver **termio**. It has been modified to correspond to the terminal driver **termios**.

34412 Output modes are specified only for XSI-conformant systems. All implementations are expected
34413 to provide *stty* operands corresponding to all of the output modes they support.

- 34414 The *stty* utility is primarily used to tailor the user interface of the terminal, such as selecting the
34415 preferred ERASE and KILL characters. As an application programming utility, *stty* can be used
34416 within shell scripts to alter the terminal settings for the duration of the script.
- 34417 The **termios** section states that individual disabling of control characters is possible through the
34418 option `_POSIX_VDISABLE`. If enabled, two conventions currently exist for specifying this:
34419 System V uses "`^-`", and BSD uses *undef*. Both are accepted by *stty* in this volume of
34420 IEEE Std. 1003.1-200x. The other BSD convention of using the letter '`u`' was rejected because it
34421 conflicts with the actual letter '`u`', which is an acceptable value for a control character.
- 34422 Early proposals did not specify the mapping of "`^c`" to control characters because the control
34423 characters were not specified in the POSIX locale character set description file requirements. The
34424 control character set is now specified in the Base Definitions volume of IEEE Std. 1003.1-200x,
34425 Chapter 3, Definitions so the historical mapping is specified. Note that although the mapping
34426 corresponds to control-character key assignments on many terminals that use the
34427 ISO/IEC 646:1991 standard (or ASCII) character encodings, the mapping specified here is to the
34428 control characters, not their keyboard encodings.
- 34429 Since **termios** supports separate speeds for input and output, two new options were added to
34430 specify each distinctly.
- 34431 Some historical implementations use standard input to get and set terminal characteristics;
34432 others use standard output. Since input from a login TTY is usually restricted to the owner while
34433 output to a TTY is frequently open to anyone, using standard input provides fewer chances of
34434 accidentally (or maliciously) altering the terminal settings of other users. Using standard input
34435 also allows *stty -a* and *stty -g* output to be redirected for later use. Therefore, usage of standard
34436 input is required by this volume of IEEE Std. 1003.1-200x.
- 34437 **FUTURE DIRECTIONS**
- 34438 None.
- 34439 **SEE ALSO**
- 34440 The Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface
- 34441 **CHANGE HISTORY**
- 34442 First released in Issue 2.
- 34443 **Issue 4**
- 34444 Aligned with the ISO/IEC 9945-2:1993 standard.
- 34445 **Issue 5**
- 34446 The description of **tabs** is clarified.
- 34447 **FUTURE DIRECTIONS** section added.
- 34448 **Issue 6**
- 34449 The legacy items **iucl(-iucl)**, **xcase**, **olcuc(-olcuc)**, **lcase(-lcase)**, and **LCASE(-LCASE)**, are
34450 removed.

34451 NAME

34452 tabs — set terminal tabs

34453 SYNOPSIS

34454 UP XSI tabs [-n | -a | -a2 | -c | -c2 | -c3 | -f | -p | -s | -u][+m[n]] [-T type]

34455 tabs [-T type][+[n]] n1[,n2,...]

34456

34457 DESCRIPTION

34458 The *tabs* utility shall display a series of characters that first clears the hardware terminal tab
 34459 XSI settings and then initializes the tab stops at the specified positions and optionally adjusts the
 34460 margin.

34461 The phrase “tab-stop position *N*” shall be taken to mean that, from the start of a line of output,
 34462 tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position
 34463 on that line. The maximum number of tab stops allowed is terminal-dependent.

34464 It need not be possible to implement *tabs* on certain terminals. If the terminal type obtained from
 34465 the *TERM* environment variable or *-T* option represents such a terminal, an appropriate
 34466 diagnostic message shall be written to standard error and *tabs* shall exit with a status greater
 34467 than zero.

34468 OPTIONS

34469 The *tabs* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 34470 XSI 12.2, Utility Syntax Guidelines, except for various extensions: the options *-a2*, *-c2*, and *-c3* are
 34471 multi-character.

34472 The following options shall be supported:

34473 *-n* Specify repetitive tab stops separated by a uniform number of column positions, *n*,
 34474 where *n* is a single-digit decimal number. The default usage of *tabs* with no
 34475 arguments shall be equivalent to *tabs-8*. When *-0* is used, the tab stops shall be
 34476 cleared and no new ones set.

34477 XSI *-a* 1,10,16,36,72
 34478 Assembler, applicable to some mainframes.

34479 XSI *-a2* 1,10,16,40,72
 34480 Assembler, applicable to some mainframes.

34481 XSI *-c* 1,8,12,16,20,55
 34482 COBOL, normal format.

34483 XSI *-c2* 1,6,10,14,49
 34484 COBOL, compact format (columns 1-6 omitted).

34485 XSI *-c3* 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
 34486 COBOL compact format (columns 1-6 omitted), with more tabs than *-c2*.

34487 XSI *-f* 1,7,11,15,19,23
 34488 FORTRAN

34489 XSI *-p* 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
 34490 PL/1

34491 XSI *-s* 1,10,55
 34492 SNOBOL

34493 XSI *-u* 1,12,20,44
 34494 Assembler, applicable to some mainframes.

34495 -T *type* Indicate the type of terminal. If this option is not supplied and the *TERM* variable
 34496 is unset or null, an unspecified default terminal type shall be used. The setting of
 34497 *type* shall take precedence over the value in *TERM*.

34498 **OPERANDS**

34499 The following operand shall be supported:

34500 *n1*[,*n2*,...] A single command line argument that consists of tab-stop values separated using
 34501 either commas or <blank> characters. The application shall ensure that the tab-
 34502 stop values are positive decimal integers in strictly ascending order. If any number
 34503 (except the first one) is preceded by a plus sign, it is taken as an increment to be
 34504 added to the previous value. For example, the tab lists 1,10,20,30 and 1,10,+10,+10
 34505 are considered to be identical.

34506 **STDIN**

34507 Not used.

34508 **INPUT FILES**

34509 None.

34510 **ENVIRONMENT VARIABLES**

34511 The following environment variables shall affect the execution of *tabs*:

34512 *LANG* Provide a default value for the internationalization variables that are unset or null.
 34513 If *LANG* is unset or null, the corresponding value from the implementation-
 34514 defined default locale shall be used. If any of the internationalization variables
 34515 contains an invalid setting, the utility shall behave as if none of the variables had
 34516 been defined.

34517 *LC_ALL* If set to a non-empty string value, override the values of all the other
 34518 internationalization variables.

34519 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 34520 characters (for example, single-byte as opposed to multi-byte characters in
 34521 arguments).

34522 *LC_MESSAGES*

34523 Determine the locale that should be used to affect the format and contents of
 34524 diagnostic messages written to standard error.

34525 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

34526 *TERM* Determine the terminal type. If this variable is unset or null, and if the -T option is
 34527 not specified, an unspecified default terminal type shall be used.

34528 **ASYNCHRONOUS EVENTS**

34529 Default.

34530 **STDOUT**

34531 If standard output is a terminal, the appropriate sequence to clear and set the tab stops may be
 34532 written to standard output in an unspecified format. If standard output is not a terminal,
 34533 undefined results occur.

34534 **STDERR**

34535 Used only for diagnostic messages.

34536 **OUTPUT FILES**

34537 None.

34538 **EXTENDED DESCRIPTION**

34539 None.

34540 **EXIT STATUS**

34541 The following exit values shall be returned:

34542 0 Successful completion.

34543 >0 An error occurred.

34544 **CONSEQUENCES OF ERRORS**

34545 Default.

34546 **APPLICATION USAGE**34547 This utility makes use of the terminal's hardware tabs and the *stty tabs* option.

34548 This utility is not recommended for application use.

34549 Some integrated display units might not have escape sequences to set tab stops, but may be set
 34550 by internal system calls. On these terminals, *tabs* works if standard output is directed to the
 34551 terminal; if output is directed to another file, however, *tabs* fails.

34552 **EXAMPLES**

34553 None.

34554 **RATIONALE**

34555 Consideration was given to having the *tput* utility handle all of the functions described in *tabs*.
 34556 However, the separate *tabs* utility was retained because it seems more intuitive to use a
 34557 command named *tabs* than *tput* with a new option. The POSIX Shell and Utilities *tput* does not
 34558 support setting or clearing tabs, and no known historical version of *tabs* supports the capability
 34559 of setting arbitrary tab stops.

34560 The System V *tabs* interface is very complex; the version in this volume of IEEE Std. 1003.1-200x
 34561 has a reduced feature list. There was considerable sentiment for specifying only a means of
 34562 resetting the tabs back to a known state—presumably the “standard” of tabs every eight
 34563 positions. The following features were omitted:

- 34564 • Setting tab stops tailored for certain programming languages; the standard developers were
 34565 concerned that it would be difficult to decide which languages to include and where the tabs
 34566 should be.
- 34567 • Setting tab stops via the first line in a file, using *—file*. Since even the SVID has no complete
 34568 explanation of this feature, it is doubtful that it is in widespread use.
- 34569 • Setting the left margin using *+mn*. As this does not work with all terminal types, it was
 34570 omitted.

34571 In an early proposal, a *-t tablist* option was added for consistency with *expand*; this was later
 34572 removed when inconsistencies with the historical list of tabs were identified.

34573 Consideration was given to adding a *-p* option that would output the current tab settings so
 34574 that they could be saved and then later restored. This was not accepted because querying the tab
 34575 stops of the terminal is not a capability in historical *terminfo* or *termcap* facilities and might not be
 34576 supported on a wide range of terminals.

34577 **FUTURE DIRECTIONS**

34578 None.

34579 **SEE ALSO**34580 *expand, stty, unexpand*34581 **CHANGE HISTORY**

34582 First released in Issue 2.

34583 **Issue 4**

34584 Aligned with the ISO/IEC 9945-2:1993 standard.

34585 **Issue 6**

34586 This utility is now marked as part of the User Portability Utilities option.

34587 The normative text is reworded to avoid use of the term “must” for application requirements.

34588 **NAME**

34589 tail — copy the last part of a file

34590 **SYNOPSIS**34591 tail [-f][-c *number*| -n *number*][*file*]34592 **DESCRIPTION**34593 The *tail* utility shall copy its input file to the standard output beginning at a designated place.

34594 Copying shall begin at the point in the file indicated by the *-c number* or *-n number* options. The
 34595 option-argument *number* shall be counted in units of lines or bytes, according to the options *-n*
 34596 and *-c*. Both line and byte counts start from 1.

34597 Tails relative to the end of the file may be saved in an internal buffer, and thus may be limited in
 34598 length. Such a buffer, if any, is no smaller than {LINE_MAX}*10 bytes.

34599 **OPTIONS**

34600 The *tail* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 34601 12.2, Utility Syntax Guidelines.

34602 The following options shall be supported:

34603 *-c number* The application shall ensure that the *number* option-argument is a decimal integer
 34604 whose sign affects the location in the file, measured in bytes, to begin the copying:

34605

34606

34607

34608

Sign	Copying Starts
+	Relative to the beginning of the file.
-	Relative to the end of the file.
<i>none</i>	Relative to the end of the file.

34609 The origin for counting shall be 1; that is, *-c +1* represents the first byte of the file,
 34610 *-c -1* the last.

34611 *-f* If the input file is a regular file or if the *file* operand specifies a FIFO, do not
 34612 terminate after the last line of the input file has been copied, but read and copy
 34613 further bytes from the input file when they become available. If no *file* operand is
 34614 specified and standard input is a pipe, the *-f* option shall be ignored. If the input
 34615 file is not a FIFO, pipe, or regular file, it is unspecified whether or not the *-f* option
 34616 shall be ignored.

34617 *-n number* This option is equivalent to *-c number*, except the starting location in the file shall
 34618 be measured in lines instead of bytes. The origin for counting shall be 1; that is, *-n*
 34619 *+1* represents the first line of the file, *-n -1* the last.

34620 If neither *-c* nor *-n* is specified, *-n 10* shall be assumed.34621 **OPERANDS**

34622 The following operand shall be supported:

34623 *file* A path name of an input file. If no *file* operands are specified, the standard input
 34624 shall be used.

34625 **STDIN**

34626 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES
 34627 section.

34628 **INPUT FILES**

34629 If the `-c` option is specified, the input file can contain arbitrary data; otherwise, the input file
34630 shall be a text file.

34631 **ENVIRONMENT VARIABLES**

34632 The following environment variables shall affect the execution of *tail*:

34633 *LANG* Provide a default value for the internationalization variables that are unset or null.
34634 If *LANG* is unset or null, the corresponding value from the implementation-
34635 defined default locale shall be used. If any of the internationalization variables
34636 contains an invalid setting, the utility shall behave as if none of the variables had
34637 been defined.

34638 *LC_ALL* If set to a non-empty string value, override the values of all the other
34639 internationalization variables.

34640 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
34641 characters (for example, single-byte as opposed to multi-byte characters in
34642 arguments and input files).

34643 *LC_MESSAGES*

34644 Determine the locale that should be used to affect the format and contents of
34645 diagnostic messages written to standard error.

34646 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

34647 **ASYNCHRONOUS EVENTS**

34648 Default.

34649 **STDOUT**

34650 The designated portion of the input file shall be written to standard output.

34651 **STDERR**

34652 Used only for diagnostic messages.

34653 **OUTPUT FILES**

34654 None.

34655 **EXTENDED DESCRIPTION**

34656 None.

34657 **EXIT STATUS**

34658 The following exit values shall be returned:

34659 0 Successful completion.

34660 >0 An error occurred.

34661 **CONSEQUENCES OF ERRORS**

34662 Default.

34663 **APPLICATION USAGE**

34664 The `-c` option should be used with caution when the input is a text file containing multi-byte
34665 characters; it may produce output that does not start on a character boundary.

34666 Although the input file to *tail* can be any type, the results might not be what would be expected
34667 on some character special device files or on file types not described by the System Interfaces
34668 volume of IEEE Std. 1003.1-200x. Since this volume of IEEE Std. 1003.1-200x does not specify the
34669 block size used when doing input, *tail* need not read all of the data from devices that only
34670 perform block transfers.

34671 **EXAMPLES**

34672 The `-f` option can be used to monitor the growth of a file that is being written by some other
34673 process. For example, the command:

```
34674 tail -f fred
```

34675 prints the last ten lines of the file **fred**, followed by any lines that are appended to **fred** between
34676 the time *tail* is initiated and killed. As another example, the command:

```
34677 tail -f -c 15 fred
```

34678 prints the last 15 bytes of the file **fred**, followed by any bytes that are appended to **fred** between
34679 the time *tail* is initiated and killed.

34680 **RATIONALE**

34681 This version of *tail* was created to allow conformance to the Utility Syntax Guidelines. The
34682 historical `-b` option was omitted because of the general non-portability of block-sized units of
34683 text. The `-c` option historically meant “characters”, but this volume of IEEE Std. 1003.1-200x
34684 indicates that it means “bytes”. This was selected to allow reasonable implementations when
34685 multi-byte characters are possible; it was not named `-b` to avoid confusion with the historical
34686 `-b`.

34687 The origin of counting both lines and bytes is 1, matching all widespread historical
34688 implementations.

34689 The restriction on the internal buffer is a compromise between the historical System V
34690 implementation of 4 096 bytes and the BSD 32 768 bytes.

34691 The `-f` option has been implemented as a loop that sleeps for 1 second and copies any bytes that
34692 are available. This is sufficient, but if more efficient methods of determining when new data are
34693 available are developed, implementations are encouraged to use them.

34694 Historical documentation indicates that *tail* ignores the `-f` option if the input file is a pipe (pipe
34695 and FIFO on systems that support FIFOs). On BSD-based systems, this has been true; on System
34696 V-based systems, this was true when input was taken from standard input, but it did not ignore
34697 the `-f` flag if a FIFO was named as the *file* operand. Since the `-f` option is not useful on pipes and
34698 all historical implementations ignore `-f` if no *file* operand is specified and standard input is a
34699 pipe, this volume of IEEE Std. 1003.1-200x requires this behavior. However, since the `-f` option is
34700 useful on a FIFO, this volume of IEEE Std. 1003.1-200x also requires that if standard input is a
34701 FIFO or a FIFO is named, the `-f` option shall not be ignored. Although historical behavior does
34702 not ignore the `-f` option for other file types, this is unspecified so that implementations are
34703 allowed to ignore the `-f` option if it is known that the file cannot be extended.

34704 This was changed to the current form based on comments noting that `-c` was almost never used
34705 without specifying a number and that there was no need to specify `-l` if `-n number` was given.

34706 **FUTURE DIRECTIONS**

34707 None.

34708 **SEE ALSO**34709 *head*34710 **CHANGE HISTORY**

34711 First released in Issue 2.

34712 **Issue 4**

34713 Aligned with the ISO/IEC 9945-2: 1993 standard.

34714 **Issue 6**

34715 The obsolescent SYNOPSIS lines and associated text are removed.

34716 The normative text is reworded to avoid use of the term “must” for application requirements.

34717 NAME

34718 talk — talk to another user

34719 SYNOPSIS

34720 UP `talk address [terminal]`

34721

34722 DESCRIPTION

34723 The *talk* utility is a two-way, screen-oriented communication program.34724 When first invoked, *talk* shall send a message similar to:34725 Message from *<unspecified string>*34726 talk: connection requested by *your_address*34727 talk: respond with: talk *your_address*34728 to the specified *address*. At this point, the recipient of the message can reply by typing:34729 `talk your_address`34730 Once communication is established, the two parties can type simultaneously, with their output
34731 displayed in separate regions of the screen. Characters shall be processed as follows:

- 34732 • Typing the alert character shall alert the recipient's terminal.
- 34733 • Typing `<control>-L` shall cause the sender's screen regions to be refreshed.
- 34734 • Typing the erase and kill characters shall affect the sender's terminal in the manner described
34735 by the **termios** interface in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11,
34736 General Terminal Interface.
- 34737 • Typing the interrupt or end-of-file characters shall terminate the local *talk* utility. Once the
34738 *talk* session has been terminated on one side, the other side of the *talk* session shall be notified
34739 that the *talk* session has been terminated and shall be able to do nothing except exit.
- 34740 • Typing characters from *LC_CTYPE* classifications **print** or **space** shall cause those characters
34741 to be sent to the recipient's terminal.
- 34742 • When and only when the *stty ixten* local mode is enabled, the existence and processing of
34743 additional special control characters and multi-byte or single-byte functions shall be
34744 implementation-defined.
- 34745 • Typing other non-printable characters shall cause implementation-defined sequences of
34746 printable characters to be sent to the recipient's terminal.

34747 Permission to be a recipient of a *talk* message can be denied or granted by use of the *mesg* utility.
34748 However, a user's privilege may further constrain the domain of accessibility of other users'
34749 terminals. The *talk* utility shall fail when the user lacks the appropriate privileges to perform the
34750 requested action.

34751 Certain block-mode terminals do not have all the capabilities necessary to support the
34752 simultaneous exchange of messages required for *talk*. When this type of exchange cannot be
34753 supported on such terminals, the implementation may support an exchange with reduced levels
34754 of simultaneous interaction or it may report an error describing the terminal-related deficiency.

34755 OPTIONS

34756 None.

34757 **OPERANDS**

34758 The following operands shall be supported:

34759 *address* The recipient of the *talk* session. One form of *address* is the <*user name*>, as returned
34760 by the *who* utility. Other address formats and how they are handled are
34761 unspecified.

34762 *terminal* If the recipient is logged in more than once, the *terminal* argument can be used to
34763 indicate the appropriate terminal name. If *terminal* is not specified, the *talk* message
34764 shall be displayed on one or more accessible terminals in use by the recipient. The
34765 format of *terminal* shall be the same as that returned by the *who* utility.

34766 **STDIN**

34767 Characters read from standard input shall be copied to the recipient's terminal in an unspecified
34768 manner. If standard input is not a terminal, *talk* shall write a diagnostic message and exit with a
34769 non-zero status.

34770 **INPUT FILES**

34771 None.

34772 **ENVIRONMENT VARIABLES**

34773 The following environment variables shall affect the execution of *talk*:

34774 *LANG* Provide a default value for the internationalization variables that are unset or null.
34775 If *LANG* is unset or null, the corresponding value from the implementation-
34776 defined default locale shall be used. If any of the internationalization variables
34777 contains an invalid setting, the utility shall behave as if none of the variables had
34778 been defined.

34779 *LC_ALL* If set to a non-empty string value, override the values of all the other
34780 internationalization variables.

34781 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
34782 characters (for example, single-byte as opposed to multi-byte characters in
34783 arguments and input files). If the recipient's locale does not use an *LC_CTYPE*
34784 equivalent to the sender's, the results are undefined.

34785 *LC_MESSAGES*

34786 Determine the locale that should be used to affect the format and contents of
34787 diagnostic messages written to standard error and informative messages written to
34788 standard output.

34789 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

34790 *TERM* Determine the name of the invoker's terminal type. If this variable is unset or null,
34791 an unspecified default terminal type shall be used.

34792 **ASYNCHRONOUS EVENTS**

34793 When the *talk* utility receives a SIGINT signal, the utility shall terminate and exit with a zero
34794 status. It shall take the standard action for all other signals.

34795 **STDOUT**

34796 If standard output is a terminal, characters copied from the recipient's standard input may be
34797 written to standard output. Standard output also may be used for diagnostic messages. If
34798 standard output is not a terminal, *talk* shall exit with a non-zero status.

34799 **STDERR**

34800 None.

34801 **OUTPUT FILES**

34802 None.

34803 **EXTENDED DESCRIPTION**

34804 None.

34805 **EXIT STATUS**

34806 The following exit values shall be returned:

34807 0 Successful completion.

34808 >0 An error occurred or *talk* was invoked on a terminal incapable of supporting it.34809 **CONSEQUENCES OF ERRORS**

34810 Default.

34811 **APPLICATION USAGE**

34812 Because the handling of non-printable, non-`<space>` characters is tied to the *stty* description of

34813 **ixten**, implementation extensions within the terminal driver can be accessed. For example,

34814 some implementations provide line editing functions with certain control character sequences.

34815 **EXAMPLES**

34816 None.

34817 **RATIONALE**

34818 The *write* utility was included in this volume of IEEE Std. 1003.1-200x since it can be

34819 implemented on all terminal types. The *talk* utility, which cannot be implemented on certain

34820 terminals, was considered to be a “better” communications interface. Both of these programs are

34821 in widespread use on historical implementations. Therefore, both utilities have been specified.

34822 All references to networking abilities (*talking* to a user on another system) were removed as

34823 being outside the scope of this volume of IEEE Std. 1003.1-200x.

34824 Historical BSD and System V versions of *talk* terminate both of the conversations when either

34825 user breaks out of the session. This can lead to adverse consequences if a user unwittingly

34826 continues to enter text that is interpreted by the shell when the other terminates the session.

34827 Therefore, the version of *talk* specified by this volume of IEEE Std. 1003.1-200x requires both

34828 users to terminate their end of the session explicitly.

34829 Only messages sent to the terminal of the invoking user can be internationalized in any way:

- 34830 • The original “Message from *<unspecified string>* ...” message sent to the terminal of the
- 34831 recipient cannot be internationalized because the environment of the recipient is as yet
- 34832 inaccessible to the *talk* utility. The environment of the invoking party is irrelevant.

- 34833 • Subsequent communication between the two parties cannot be internationalized because the
- 34834 two parties may specify different languages in their environment (and non-portable
- 34835 characters cannot be mapped from one language to another).

- 34836 • Neither party can be required to communicate in a language other than C and/or the one
- 34837 specified by their environment because unavailable terminal hardware support (for example,
- 34838 fonts) may be required.

34839 The text in the STDOUT section reflects the usage of the verb “display” in this section; some *talk*

34840 implementations actually use standard output to write to the terminal, but this volume of

34841 IEEE Std. 1003.1-200x does not require that to be the case.

34842 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*
34843 require that they all use or accept the same format.

34844 The handling of non-printable characters is partially implementation-defined because the details
34845 of mapping them to printable sequences is not needed by the user. Historical implementations,
34846 for security reasons, disallow the transmission of non-printable characters that may send
34847 commands to the other terminal.

34848 **FUTURE DIRECTIONS**

34849 None.

34850 **SEE ALSO**

34851 *mesg*, *who*, *write*, the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General
34852 Terminal Interface

34853 **CHANGE HISTORY**

34854 First released in Issue 4.

34855 **Issue 6**

34856 This utility is now marked as part of the User Portability Utilities option.

34857 **NAME**

34858 tee — duplicate standard input

34859 **SYNOPSIS**34860 tee [-ai][*file...*]34861 **DESCRIPTION**34862 The *tee* utility shall copy standard input to standard output, making a copy in zero or more files.34863 The *tee* utility shall not buffer output.34864 If the **-a** option is not specified, output files shall be written (see Section 1.7.1.4 (on page 2209)).34865 **OPTIONS**34866 The *tee* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
34867 12.2, Utility Syntax Guidelines.

34868 The following options shall be supported:

34869 **-a** Append the output to the files.34870 **-i** Ignore the SIGINT signal.34871 **OPERANDS**

34872 The following operands shall be supported:

34873 *file* A path name of an output file. Processing of at least 13 *file* operands shall be
34874 supported.34875 **STDIN**

34876 The standard input can be of any type.

34877 **INPUT FILES**

34878 None.

34879 **ENVIRONMENT VARIABLES**34880 The following environment variables shall affect the execution of *tee*:34881 **LANG** Provide a default value for the internationalization variables that are unset or null.
34882 If *LANG* is unset or null, the corresponding value from the implementation-
34883 defined default locale shall be used. If any of the internationalization variables
34884 contains an invalid setting, the utility shall behave as if none of the variables had
34885 been defined.34886 **LC_ALL** If set to a non-empty string value, override the values of all the other
34887 internationalization variables.34888 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
34889 characters (for example, single-byte as opposed to multi-byte characters in
34890 arguments).34891 **LC_MESSAGES**34892 Determine the locale that should be used to affect the format and contents of
34893 diagnostic messages written to standard error.34894 **XSI** **NLS_PATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.34895 **ASYNCHRONOUS EVENTS**34896 Default, except that if the **-i** option was specified, SIGINT shall be ignored.

34897 **STDOUT**

34898 The standard output shall be a copy of the standard input.

34899 **STDERR**

34900 Used only for diagnostic messages.

34901 **OUTPUT FILES**

34902 If any *file* operands are specified, the standard input shall be copied to each named file.

34903 **EXTENDED DESCRIPTION**

34904 None.

34905 **EXIT STATUS**

34906 The following exit values shall be returned:

34907 0 The standard input was successfully copied to all output files.

34908 >0 An error occurred.

34909 **CONSEQUENCES OF ERRORS**

34910 If a write to any successfully opened *file* operand fails, writes to other successfully opened *file* operands and standard output shall continue, but the exit status shall be non-zero. Otherwise, the default actions specified in Section 1.11 (on page 2224) apply.

34913 **APPLICATION USAGE**

34914 The *tee* utility is usually used in a pipeline, to make a copy of the output of some utility.

34915 The *file* operand is technically optional, but *tee* is no more useful than *cat* when none is specified.

34916 **EXAMPLES**

34917 Save an unsorted intermediate form of the data in a pipeline:

34918 ... | tee unsorted | sort > sorted

34919 **RATIONALE**

34920 The buffering requirement means that *tee* is not allowed to use ISO C standard fully buffered or line-buffered writes. It does not mean that *tee* has to do 1-byte reads followed by 1-byte writes.

34922 It should be noted that early versions of BSD ignore any invalid options and accept a single '-' as an alternative to -i. They also print a message if unable to open a file:

34924 "tee: cannot access %s\n", <pathname>

34925 Historical implementations ignore write errors. This is explicitly not permitted by this volume of IEEE Std. 1003.1-200x.

34927 Some historical implementations use O_APPEND when providing append mode; others use the *lseek()* function to seek to the end of file after opening the file without O_APPEND. This volume of IEEE Std. 1003.1-200x requires functionality equivalent to using O_APPEND; see Section 1.7.1.4 (on page 2209).

34931 **FUTURE DIRECTIONS**

34932 None.

34933 **SEE ALSO**

34934 *cat*

34935 **CHANGE HISTORY**

34936 First released in Issue 2.

- 34937 **Issue 4**
- 34938 Aligned with the ISO/IEC 9945-2: 1993 standard.
- 34939 **Issue 6**
- 34940 IEEE PASC Interpretation 1003.2 #168 is applied.

34941 **NAME**

34942 test — evaluate expression

34943 **SYNOPSIS**34944 test [*expression*]34945 [[*expression*]]34946 **DESCRIPTION**

34947 The *test* utility shall evaluate the *expression* and indicates the result of the evaluation by its exit
 34948 status. An exit status of zero indicates that the expression evaluated as true and an exit status of
 34949 1 indicates that the expression evaluated as false.

34950 In the second form of the utility, which uses "[]" rather than *test*, the application shall ensure
 34951 that the square brackets are separate arguments.

34952 **OPTIONS**

34953 The *test* utility shall not recognize the "--" argument in the manner specified by guideline 10 in
 34954 the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2, Utility Syntax Guidelines.

34955 No options shall be supported.

34956 **OPERANDS**

34957 The application shall ensure that all operators and elements of primaries are presented as
 34958 separate arguments to the *test* utility.

34959 The following primaries can be used to construct *expression*:

34960 **-b file** True if *file* exists and is a block special file.

34961 **-c file** True if *file* exists and is a character special file.

34962 **-d file** True if *file* exists and is a directory.

34963 **-e file** True if *file* exists.

34964 **-f file** True if *file* exists and is a regular file.

34965 **-g file** True if *file* exists and its set group ID flag is set.

34966 **-h file** True if *file* exists and is a symbolic link.

34967 **-n string** True if the length of *string* is non-zero.

34968 **-p file** True if *file* is a named pipe (FIFO).

34969 **-r file** True if *file* exists and is readable. True shall indicate that permission to read from
 34970 *file* will be granted, as defined in Section 1.7.1.4 (on page 2209).

34971 **-s file** True if *file* exists and has a size greater than zero.

34972 **-t file_descriptor**

34973 True if the file whose file descriptor number is *file_descriptor* is open and is
 34974 associated with a terminal.

34975 **-u file** True if *file* exists and its set-user-ID flag is set.

34976 **-w file** True if *file* exists and is writable. True shall indicate that permission to write from
 34977 *file* will be granted, as defined in Section 1.7.1.4 (on page 2209).

34978 **-x file** True if *file* exists and is executable. True if *file* exists and is executable. True shall
 34979 indicate that permission to execute *file* will be granted, as defined in Section 1.7.1.4
 34980 (on page 2209). If *file* is a directory, true shall indicate that permission to search *file*
 34981 will be granted.

- 34982 **-z string** True if the length of string *string* is zero.
- 34983 **string** True if the string *string* is not the null string.
- 34984 **s1 = s2** True if the strings *s1* and *s2* are identical.
- 34985 **s1 != s2** True if the strings *s1* and *s2* are not identical.
- 34986 **n1 -eq n2** True if the integers *n1* and *n2* are algebraically equal.
- 34987 **n1 -ne n2** True if the integers *n1* and *n2* are not algebraically equal.
- 34988 **n1 -gt n2** True if the integer *n1* is algebraically greater than the integer *n2*.
- 34989 **n1 -ge n2** True if the integer *n1* is algebraically greater than or equal to the integer *n2*.
- 34990 **n1 -lt n2** True if the integer *n1* is algebraically less than the integer *n2*.
- 34991 **n1 -le n2** True if the integer *n1* is algebraically less than or equal to the integer *n2*.
- 34992 XSI **expression1 -a expression2**
 34993 True if both *expression1* and *expression2* are true. The **-a** binary primary is left
 34994 associative. It has a higher precedence than **-o**.
- 34995 XSI **expression1 -o expression2**
 34996 True if either *expression1* or *expression2* is true. The **-o** binary primary is left
 34997 associative.
- 34998 With the exception of the **-h file** primary, if a *file* argument is a symbolic link, *test* shall evaluate
 34999 the expression by resolving the symbolic link and using the file referenced by the link.
- 35000 These primaries can be combined with the following operators:
- 35001 **! expression** True if *expression* is false.
- 35002 XSI **(expression)** True if *expression* is true. The parentheses can be used to alter the normal
 35003 precedence and associativity.
- 35004 The primaries with two elements of the form:
- 35005 *-primary_operator primary_operand*
- 35006 are known as *unary primaries*. The primaries with three elements in either of the two forms:
- 35007 *primary_operand -primary_operator primary_operand*
- 35008 *primary_operand primary_operator primary_operand*
- 35009 are known as *binary primaries*. Additional implementation-defined operators and
 35010 *primary_operators* may be provided by implementations. They shall be of the form *-operator*
 35011 where the first character of *operator* is not a digit.
- 35012 The algorithm for determining the precedence of the operators and the return value that shall be
 35013 generated is based on the number of arguments presented to *test*. (However, when using the
 35014 "[...]" form, the right-bracket final argument shall not be counted in this algorithm.)
- 35015 In the following list, \$1, \$2, \$3, and \$4 represent the arguments presented to *test*:
- 35016 0 arguments: Exit false (1).
- 35017 1 argument: Exit true (0) if \$1 is not null; otherwise, exit false.
- 35018 2 arguments: • If \$1 is '!', exit true if \$2 is null, false if \$2 is not null.
- 35019 • If \$1 is a unary primary, exit true if the unary test is true, false if the unary
 35020 test is false.

35060 **OUTPUT FILES**

35061 None.

35062 **EXTENDED DESCRIPTION**

35063 None.

35064 **EXIT STATUS**

35065 The following exit values shall be returned:

35066 0 *expression* evaluated to true.35067 1 *expression* evaluated to false or *expression* was missing.

35068 >1 An error occurred.

35069 **CONSEQUENCES OF ERRORS**

35070 Default.

35071 **APPLICATION USAGE**

35072 Scripts should be careful when dealing with user-supplied input that could be confused with
 35073 primaries and operators. Unless the application writer knows all the cases that produce input to
 35074 the script, invocations like:

35075 `test "$1" -a "$2"`

35076 should be written as:

35077 `test "$1" && test "$2"`

35078 to avoid problems if a user supplied values such as \$1 set to '!' and \$2 set to the null string.
 35079 That is, in cases where maximal portability is of concern, replace:

35080 `test expr1 -a expr2`

35081 with:

35082 `test expr1 && test expr2`

35083 and replace:

35084 `test expr1 -o expr2`

35085 with:

35086 `test expr1 || test expr2`

35087 but note that, in *test*, `-a` has higher precedence than `-o` while `&&` and `||` have equal
 35088 precedence in the shell.

35089 Parentheses or braces can be used in the shell command language to effect grouping.

35090 Parentheses must be escaped when using *sh*; for example:35091 `test \(expr1 -a expr2 \) -o expr3`

35092 This command is not always portable outside XSI-conformant systems. The following form can
 35093 be used instead:

35094 `(test expr1 && test expr2) || test expr3`

35095 The two commands:

35096 `test "$1"`35097 `test ! "$1"`

35098 could not be used reliably on some historical systems. Unexpected results would occur if such a
 35099 *string* expression were used and \$1 expanded to '!', '(', or a known unary primary. Better
 35100 constructs are:

```
35101 test -n "$1"
35102 test -z "$1"
```

35103 respectively.

35104 Historical systems have also been unreliable given the common construct:

```
35105 test "$response" = "expected string"
```

35106 One of the following is a more reliable form:

```
35107 test "X$response" = "Xexpected string"
35108 test "expected string" = "$response"
```

35109 Note that the second form assumes that *expected string* could not be confused with any unary
 35110 primary. If *expected string* starts with '- ', '(', '! ', or even '= ', the first form should be used
 35111 instead. Using the preceding rules without the XSI marked extensions, any of the three
 35112 comparison forms is reliable, given any input. (However, note that the strings are quoted in all
 35113 cases.)

35114 Because the string comparison binary primaries, '= ' and '! =', have a higher precedence than
 35115 any unary primary in the greater than 4 argument case, unexpected results can occur if
 35116 arguments are not properly prepared. For example, in:

```
35117 test -d $1 -o -d $2
```

35118 If \$1 evaluates to a possible directory name of '= ', the first three arguments are considered a
 35119 string comparison, which shall cause a syntax error when the second **-d** is encountered. One of
 35120 the following forms prevents this; the second is preferred:

```
35121 test \( -d "$1" \) -o \( -d "$2" \)
35122 test -d "$1" || test -d "$2"
```

35123 Also in the greater than 4 argument case:

```
35124 test "$1" = "bat" -a "$2" = "ball"
```

35125 Syntax errors occur if \$1 evaluates to '(' or '! '. One of the following forms prevents this; the
 35126 third is preferred:

```
35127 test "X$1" = "Xbat" -a "X$2" = "Xball"
35128 test "$1" = "bat" && test "$2" = "ball"
35129 test "X$1" = "Xbat" && test "X$2" = "Xball"
```

35130 EXAMPLES

35131 1. Exit if there are not two or three arguments (two variations):

```
35132 if [ $# -ne 2 -a $# -ne 3 ]; then exit 1; fi
35133 if [ $# -lt 2 -o $# -gt 3 ]; then exit 1; fi
```

35134 2. Perform a *mkdir* if a directory does not exist:

```
35135 test ! -d tempdir && mkdir tempdir
```

35136 3. Wait for a file to become non-readable:

```
35137 while test -r thefile
35138 do
```



```

35139         sleep 30
35140     done
35141     echo '"thefile" is no longer readable'
35142
35143     4. Perform a command if the argument is one of three strings (two variations):
35144
35145     if [ "$1" = "pear" ] || [ "$1" = "grape" ] || [ "$1" = "apple" ]
35146     then
35147         command
35148     fi
35149
35150     case "$1" in
35151         pear|grape|apple) command ;;
35152     esac

```

35150 RATIONALE

35151 The KornShell-derived conditional command (double bracket [[]]) was removed from the shell
 35152 command language description in an early proposal. Objections were raised that the real
 35153 problem is misuse of the *test* command (D), and putting it into the shell is the wrong way to fix
 35154 the problem. Instead, proper documentation and a new shell reserved word (!) are sufficient.

35155 Tests that require multiple *test* operations can be done at the shell level using individual
 35156 invocations of the *test* command and shell logicals, rather than using the error-prone *-o* flag of
 35157 *test*.

35158 XSI-conformant systems support more than four arguments.

35159 XSI-conformant systems support the combining of primaries with the following constructs:

35160 *expression1 -a expression2*

35161 True if both *expression1* and *expression2* are true.

35162 *expression1 -o expression2*

35163 True if at least one of *expression1* and *expression2* are true.

35164 (*expression*)

35165 True if *expression* is true.

35166 In evaluating these more complex combined expressions, the following precedence rules are
 35167 used:

- 35168 • The unary primaries have higher precedence than the algebraic binary primaries.
- 35169 • The unary primaries have lower precedence than the string binary primaries.
- 35170 • The unary and binary primaries have higher precedence than the unary *string* primary.
- 35171 • The ! operator has higher precedence than the *-a* operator, and the *-a* operator has higher
 35172 precedence than the *-o* operator.
- 35173 • The *-a* and *-o* operators are left associative.
- 35174 • The parentheses can be used to alter the normal precedence and associativity.

35175 The BSD and System V versions of *-f* are not the same. The BSD definition was:

35176 *-f file* True if *file* exists and is not a directory.

35177 The SVID version (true if the file exists and is a regular file) was chosen for this volume of
 35178 IEEE Std. 1003.1-200x because its use is consistent with the *-b*, *-c*, *-d*, and *-p* operands (*file*
 35179 exists and is a specific file type).

35180 The `-e` primary, possessing similar functionality to that provided by the C shell, was added
 35181 because it provides the only way for a shell script to find out if a file exists without trying to
 35182 open the file. Since implementations are allowed to add additional file types, a portable script
 35183 cannot use:

```
35184 test -b foo -o -c foo -o -d foo -o -f foo -o -p foo
```

35185 to find out if `foo` is an existing file.) On historical BSD systems, the existence of a file could be
 35186 determined by:

```
35187 test -f foo -o -d foo
```

35188 but there was no easy way to determine that an existing file was a regular file. An early proposal
 35189 used the KornShell `-a` primary (with the same meaning), but this was changed to `-e` because
 35190 there were concerns about the high probability of humans confusing the `-a` primary with the `-a`
 35191 binary operator.

35192 The following option was not included because it was undocumented in most implementations,
 35193 has been removed from some implementations (including System V), and the functionality is
 35194 provided by the shell (see Section 2.6.2 (on page 2245).

35195 `-l string` The length of the string *string*.

35196 The `-b`, `-c`, `-g`, `-p`, `-u`, and `-x` operands are derived from the SVID; historical BSD does not
 35197 provide them. The `-k` operand is derived from System V; historical BSD does not provide it.

35198 On historical BSD systems, `test -w directory` always returned false because `test` tried to open the
 35199 directory for writing, which always fails.

35200 Some additional primaries newly invented or from the KornShell appeared in an early proposal
 35201 as part of the conditional command (`[[]]`): `s1 > s2`, `s1 < s2`, `str = pattern`, `str != pattern`, `f1 -nt f2`, `f1`
 35202 `-ot f2`, and `f1 -ef f2`. They were not carried forward into the `test` utility when the conditional
 35203 command was removed from the shell because they have not been included in the `test` utility
 35204 built into historical implementations of the `sh` utility.

35205 The `-t file_descriptor` primary is shown with a mandatory argument because the grammar is
 35206 ambiguous if it can be omitted. Historical implementations have allowed it to be omitted,
 35207 providing a default of 1.

35208 FUTURE DIRECTIONS

35209 None.

35210 SEE ALSO

35211 *find*

35212 CHANGE HISTORY

35213 First released in Issue 2.

35214 Issue 4

35215 Aligned with the ISO/IEC 9945-2:1993 standard.

35216 Issue 5

35217 FUTURE DIRECTIONS section added.

35218 Issue 6

35219 The `-h` operand is added for symbolic links, and access permission requirements are clarified for
 35220 the `-r`, `-w`, and `-x` operands to align with the IEEE P1003.2b draft standard.

35221 The normative text is reworded to avoid use of the term “must” for application requirements.

35222 **NAME**

35223 time — time a simple command

35224 **SYNOPSIS**35225 UP time [-p] *utility* [*argument...*]

35226

35227 **DESCRIPTION**

35228 The *time* utility shall invoke the utility named by the *utility* operand with arguments supplied as
 35229 the *argument* operands and write a message to standard error that lists timing statistics for the
 35230 utility. The message shall include the following information:

- 35231 • The elapsed (real) time between invocation of *utility* and its termination.
- 35232 • The User CPU time, equivalent to the sum of the *tms_utime* and *tms_cutime* fields returned by
 35233 the *times()* function defined in the System Interfaces volume of IEEE Std. 1003.1-200x for the
 35234 process in which *utility* is executed.
- 35235 • The System CPU time, equivalent to the sum of the *tms_stime* and *tms_cstime* fields returned
 35236 by the *times()* function for the process in which *utility* is executed.

35237 The precision of the timing shall be no less than the granularity defined for the size of the clock
 35238 tick unit on the system, but the results shall be reported in terms of standard time units (for
 35239 example, 0.02 seconds, 00:00:00.02, 1m33.75s, 365.21 seconds), not numbers of clock ticks.

35240 When *time* is used as part of a pipeline, the times reported are unspecified, except when it is the
 35241 sole command within a grouping command (see Section 2.9.4.1 (on page 2261)) in that pipeline.
 35242 For example, the commands on the left are unspecified; those on the right report on utilities **a**
 35243 and **c**, respectively:

```
35244           time a | b | c       { time a } | b | c
35245           a | b | time c       a | b | (time c)
```

35246 **OPTIONS**

35247 The *time* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 35248 12.2, Utility Syntax Guidelines.

35249 The following option shall be supported:

- 35250 **-p** Write the timing output to standard error in the format shown in the **STDERR**
 35251 section.

35252 **OPERANDS**

35253 The following operands shall be supported:

- 35254 *utility* The name of a utility that is to be invoked. If the *utility* operand names any of the
 35255 special built-in utilities in Section 2.15 (on page 2276), the results are undefined.
- 35256 *argument* Any string to be supplied as an argument when invoking the utility named by the
 35257 *utility* operand.

35258 **STDIN**

35259 Not used.

35260 **INPUT FILES**

35261 None.

35262 **ENVIRONMENT VARIABLES**

35263 The following environment variables shall affect the execution of *time*:

35264 *LANG* Provide a default value for the internationalization variables that are unset or null.
 35265 If *LANG* is unset or null, the corresponding value from the implementation-
 35266 defined default locale shall be used. If any of the internationalization variables
 35267 contains an invalid setting, the utility shall behave as if none of the variables had
 35268 been defined.

35269 *LC_ALL* If set to a non-empty string value, override the values of all the other
 35270 internationalization variables.

35271 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 35272 characters (for example, single-byte as opposed to multi-byte characters in
 35273 arguments).

35274 *LC_MESSAGES*
 35275 Determine the locale that should be used to affect the format and contents of
 35276 diagnostic and informative messages written to standard error.

35277 *LC_NUMERIC*
 35278 Determine the locale for numeric formatting.

35279 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

35280 *PATH* Determine the search path that shall be used to locate the utility to be invoked; see
 35281 the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment
 35282 Variables.

35283 **ASYNCHRONOUS EVENTS**

35284 Default.

35285 **STDOUT**

35286 Not used.

35287 **STDERR**

35288 The standard error shall be used to write the timing statistics. If *-p* is specified, the following
 35289 format shall be used in the POSIX locale:

35290 "real %f\nuser %f\nsys %f\n", <real seconds>, <user seconds>,
 35291 <system seconds>

35292 where each floating-point number shall be expressed in seconds. The precision used may be less
 35293 than the default six digits of %f, but shall be sufficiently precise to accommodate the size of the
 35294 clock tick on the system (for example, if there were 60 clock ticks per second, at least two digits
 35295 shall follow the radix character). The number of digits following the radix character shall be no
 35296 less than one, even if this always results in a trailing zero. The implementation may append
 35297 white space and additional information following the format shown here.

35298 **OUTPUT FILES**

35299 None.

35300 **EXTENDED DESCRIPTION**

35301 None.

35302 **EXIT STATUS**

35303 If the *utility* utility is invoked, the exit status of *time* shall be the exit status of *utility*; otherwise,
 35304 the *time* utility shall exit with one of the following values:

- 35305 1-125 An error occurred in the *time* utility.
- 35306 126 The utility specified by *utility* was found but could not be invoked.
- 35307 127 The utility specified by *utility* could not be found.

35308 CONSEQUENCES OF ERRORS

35309 Default.

35310 APPLICATION USAGE

35311 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if
 35312 an error occurs so that applications can distinguish “failure to find a utility” from “invoked
 35313 utility exited with an error indication”. The value 127 was chosen because it is not commonly
 35314 used for other meanings; most utilities use small values for “normal error conditions” and the
 35315 values above 128 can be confused with termination due to receipt of a signal. The value 126 was
 35316 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some
 35317 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction
 35318 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to
 35319 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for
 35320 any other reason.

35321 EXAMPLES

35322 It is frequently desirable to apply *time* to pipelines or lists of commands. This can be done by
 35323 placing pipelines and command lists in a single file; this file can then be invoked as a utility, and
 35324 the *time* applies to everything in the file.

35325 Alternatively, the following command can be used to apply *time* to a complex command:

```
35326 time sh -c 'complex-command-line'
```

35327 RATIONALE

35328 The *time* utility when originally proposed for this volume of IEEE Std. 1003.1-200x, was rejected
 35329 because it was not useful for portable applications:

- 35330 • The underlying CPU definitions from the System Interfaces volume of IEEE Std. 1003.1-200x
 35331 are vague, so the numeric output could not be compared accurately between systems or even
 35332 between invocations.
- 35333 • The creation of portable benchmark programs was outside the scope this volume of
 35334 IEEE Std. 1003.1-200x.

35335 However, *time* does fit in the scope of user portability. Human judgement can be applied to the
 35336 analysis of the output, and it could be very useful in hands-on debugging of applications or in
 35337 providing subjective measures of system performance. Hence it has been included in this
 35338 volume of IEEE Std. 1003.1-200x.

35339 The default output format has been left unspecified because historical implementations differ
 35340 greatly in their style of depicting this numeric output. The **-p** option was invented to provide
 35341 scripts a common means of obtaining this information.

35342 In the KornShell, *time* is a shell reserved word that can be used to time an entire pipeline, rather
 35343 than just a simple command. The POSIX definition has been worded to allow this
 35344 implementation. Consideration was given to invalidating this approach because of the historical
 35345 model from the C shell and System V shell. However, since the System V *time* utility historically
 35346 has not produced accurate results in pipeline timing (because the constituent processes are not
 35347 all owned by the same parent process, as allowed by POSIX), it did not seem worthwhile to
 35348 break historical KornShell usage.

35349 The term *utility* is used, rather than *command*, to highlight the fact that shell compound
35350 commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility*
35351 includes user application programs and shell scripts, not just the standard utilities.

35352 **FUTURE DIRECTIONS**

35353 None.

35354 **SEE ALSO**

35355 *sh*, the System Interfaces volume of IEEE Std. 1003.1-200x, *times()*

35356 **CHANGE HISTORY**

35357 First released in Issue 2.

35358 **Issue 4**

35359 Aligned with the ISO/IEC 9945-2:1993 standard.

35360 **Issue 6**

35361 This utility is now marked as part of the User Portability Utilities option.

35362 **NAME**

35363 touch — change file access and modification times

35364 **SYNOPSIS**35365 touch [-acm][-r *ref_file*| -t *time*] *file*...35366 **DESCRIPTION**

35367 The *touch* utility shall change the modification times, access times, or both of files. The
 35368 modification time shall be equivalent to the value of the *st_mtime* member of the **stat** structure
 35369 for a file, as described in the System Interfaces volume of IEEE Std. 1003.1-200x; the access time
 35370 shall be equivalent to the value of *st_atime*.

35371 The time used can be specified by the **-t** *time* option-argument, the corresponding time fields of
 35372 the file referenced by the **-r** *ref_file* option-argument, or the *date_time* operand, as specified in the
 35373 following sections. If none of these are specified, *touch* shall use the current time (the value
 35374 returned by the equivalent of the *time()* function defined in the System Interfaces volume of
 35375 IEEE Std. 1003.1-200x).

35376 For each *file* operand, *touch* shall perform actions equivalent to the following functions defined
 35377 in the System Interfaces volume of IEEE Std. 1003.1-200x:

- 35378 1. If *file* does not exist, a *creat()* function call is made with the *file* operand used as the *path*
 35379 argument and the value of the bitwise-inclusive OR of S_IRUSR, S_IWUSR, S_IRGRP,
 35380 S_IWGRP, S_IROTH, and S_IWOTH used as the *mode* argument.
- 35381 2. The *utime()* function is called with the following arguments:
 - 35382 a. The *file* operand is used as the *path* argument.
 - 35383 b. The **utimbuf** structure members *actime* and *modtime* are determined as described in
 35384 the OPTIONS section.

35385 **OPTIONS**

35386 The *touch* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 35387 12.2, Utility Syntax Guidelines.

35388 The following options shall be supported:

- | | | |
|----------------|---------------------------|---|
| 35389
35390 | -a | Change the access time of <i>file</i> . Do not change the modification time unless -m is also specified. |
| 35391
35392 | -c | Do not create a specified <i>file</i> if it does not exist. Do not write any diagnostic messages concerning this condition. |
| 35393
35394 | -m | Change the modification time of <i>file</i> . Do not change the access time unless -a is also specified. |
| 35395
35396 | -r <i>ref_file</i> | Use the corresponding time of the file named by the path name <i>ref_file</i> instead of the current time. |
| 35397
35398 | -t <i>time</i> | Use the specified <i>time</i> instead of the current time. The option-argument shall be a decimal number of the form: |
| 35399 | | [[<i>CC</i>] <i>YY</i>] <i>MMDDhhmm</i> [. <i>SS</i>] |
| 35400 | | where each two digits represents the following: |
| 35401 | <i>MM</i> | The month of the year [01-12]. |
| 35402 | <i>DD</i> | The day of the month [01-31]. |

35403 *hh* The hour of the day [00-23].
 35404 *mm* The minute of the hour [00-59].
 35405 *CC* The first two digits of the year (the century).
 35406 *YY* The second two digits of the year.
 35407 *SS* The second of the minute [00-61].
 35408 Both *CC* and *YY* shall be optional. If neither is given, the current year shall be
 35409 assumed. If *YY* is specified, but *CC* is not, *CC* shall be derived as follows:

35410
 35411
 35412

If <i>YY</i> is:	<i>CC</i> becomes:
69-99	19
00-68	20

35413 The resulting time shall be affected by the value of the *TZ* environment variable. If
 35414 the resulting time value precedes the Epoch, *touch* shall exit immediately with an
 35415 error status. The range of valid times past the Epoch is implementation-defined,
 35416 but it shall extend to at least the time 0 hours, 0 minutes, 0 seconds, January 1,
 35417 2038, Coordinated Universal Time. Some systems may not be able to represent
 35418 dates beyond the January 18, 2038, because they use **signed int** as a time holder.

35419 The range for *SS* is (00-61) rather than (00-59) because of leap seconds. If *SS* is 60 or
 35420 61, and the resulting time, as affected by the *TZ* environment variable, does not
 35421 refer to a leap second, the resulting time shall be one or two seconds after a time
 35422 where *SS* is 59. If *SS* is not given a value, it is assumed to be zero.

35423 If neither the **-a** nor **-m** options were specified, *touch* shall behave as if both the **-a** and **-m**
 35424 options were specified.

35425 OPERANDS

35426 The following operands shall be supported:

35427 *file* A path name of a file whose times shall be modified.

35428 STDIN

35429 Not used.

35430 INPUT FILES

35431 None.

35432 ENVIRONMENT VARIABLES

35433 The following environment variables shall affect the execution of *touch*:

35434 *LANG* Provide a default value for the internationalization variables that are unset or null.
 35435 If *LANG* is unset or null, the corresponding value from the implementation-
 35436 defined default locale shall be used. If any of the internationalization variables
 35437 contains an invalid setting, the utility shall behave as if none of the variables had
 35438 been defined.

35439 *LC_ALL* If set to a non-empty string value, override the values of all the other
 35440 internationalization variables.

35441 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 35442 characters (for example, single-byte as opposed to multi-byte characters in
 35443 arguments).

35444 *LC_MESSAGES*

35445 Determine the locale that should be used to affect the format and contents of

- 35446 diagnostic messages written to standard error.
- 35447 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 35448 **TZ** Determine the timezone to be used for interpreting the *time* option-argument.
- 35449 **ASYNCHRONOUS EVENTS**
- 35450 Default.
- 35451 **STDOUT**
- 35452 Not used.
- 35453 **STDERR**
- 35454 Used only for diagnostic messages.
- 35455 **OUTPUT FILES**
- 35456 None.
- 35457 **EXTENDED DESCRIPTION**
- 35458 None.
- 35459 **EXIT STATUS**
- 35460 The following exit values shall be returned:
- 35461 0 The utility executed successfully and all requested changes were made.
- 35462 >0 An error occurred.
- 35463 **CONSEQUENCES OF ERRORS**
- 35464 Default.
- 35465 **APPLICATION USAGE**
- 35466 The interpretation of time is taken to be *seconds since the Epoch* (see the Base Definitions volume of IEEE Std. 1003.1-200x, Section 4.12, Seconds Since the Epoch). It should be noted that implementations conforming to the System Interfaces volume of IEEE Std. 1003.1-200x do not take leap seconds into account when computing seconds since the Epoch. When *SS=60* is used, the resulting time always refers to 1 plus *seconds since the Epoch* for a time when *SS=59*.
- 35471 Although the *-t time* option-argument specifies values in 1969, the access time and modification time fields are defined in terms of seconds since the Epoch (midnight on 1 January 1970 UTC). Therefore, depending on the value of *TZ* when *touch* is run, there is never more than a few valid hours in 1969 and there need not be any valid times in 1969.
- 35475 One ambiguous situation occurs if *-t time* is not specified, *-r ref_file* is not specified, and the first operand is an eight or ten-digit decimal number. A portable script can avoid this problem by using:
- 35476 `touch -- file`
- 35477
- 35478 or:
- 35479 `touch ./file`
- 35480
- 35481 in this case.
- 35482 **EXAMPLES**
- 35483 None.
- 35484 **RATIONALE**
- 35485 The functionality of *touch* is described almost entirely through references to functions in the System Interfaces volume of IEEE Std. 1003.1-200x. In this way, there is no duplication of effort required for describing such side effects as the relationship of user IDs to the user database,

- 35488 permissions, and so on.
- 35489 There are some significant differences between the *touch* utility in this volume of
35490 IEEE Std. 1003.1-200x and those in System V and BSD systems. They are upward-compatible for
35491 historical applications from both implementations:
- 35492 1. In System V, an ambiguity exists when a path name that is a decimal number leads the
35493 operands; it is treated as a time value. In BSD, no *time* value is allowed; files may only be
35494 *touched* to the current time. The `-t time` construct solves these problems for future portable
35495 applications (note that the `-t` option is not historical practice).
 - 35496 2. The inclusion of the century digits, *CC*, is also new. Note that a ten-digit *time* value is
35497 treated as if *YY*, and not *CC*, were specified. The caveat about the range of dates following
35498 the Epoch was included as recognition that some implementations are not able to
35499 represent dates beyond 18 January 2038 because they use **signed int** as a time holder.
- 35500 The `-r` option was added because several comments requested this capability. This option was
35501 named `-f` in an early proposal, but was changed because the `-f` option is used in the BSD version
35502 of *touch* with a different meaning.
- 35503 At least one historical implementation of *touch* incremented the exit code if `-c` was specified and
35504 the file did not exist. This volume of IEEE Std. 1003.1-200x requires exit status zero if no errors
35505 occur.
- 35506 **FUTURE DIRECTIONS**
- 35507 Applications should use the `-r` or `-t` options.
- 35508 **SEE ALSO**
- 35509 *date*, the System Interfaces volume of IEEE Std. 1003.1-200x, *creat()*, *time()*, `<sys/stat.h>`
- 35510 **CHANGE HISTORY**
- 35511 First released in Issue 2.
- 35512 **Issue 4**
- 35513 Aligned with the ISO/IEC 9945-2:1993 standard.
- 35514 **Issue 6**
- 35515 The obsolescent *date_time* operand is removed.
- 35516 The Open Group corrigenda item U027/1 has been applied. This extends the range of valid time
35517 past the Epoch to at least the time 0 hours, 0 minutes, 0 seconds, January 1, 2038, Coordinated
35518 Universal Time. This is a new requirement on POSIX implementations.
- 35519 **Notes to Reviewers**
- 35520 *This section with side shading will not appear in the final copy. - Ed.*
- 35521 Should leap seconds be 00-61? c9x infers that it is only 00-60, and astronomers confirm that
35522 double leap seconds do not occur.

35523 **NAME**

35524 tput — change terminal characteristics

35525 **SYNOPSIS**35526 UP tput [-T *type*] *operand...*

35527

35528 **DESCRIPTION**

35529 The *tput* utility shall display terminal-dependent information. The manner in which this
 35530 information is retrieved is unspecified. The information displayed shall clear the terminal screen,
 35531 initialize the user's terminal, or reset the user's terminal, depending on the operand given. The
 35532 exact consequences of displaying this information are unspecified.

35533 **OPTIONS**

35534 The *tput* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 35535 12.2, Utility Syntax Guidelines.

35536 The following option shall be supported:

35537 **-T *type*** Indicate the type of terminal. If this option is not supplied and the *TERM* variable
 35538 is unset or null, an unspecified default terminal type shall be used. The setting of
 35539 *type* shall take precedence over the value in *TERM*.

35540 **OPERANDS**

35541 The following strings shall be supported as operands by the implementation in the POSIX locale:

35542 **clear** Display the clear-screen sequence.

35543 **init** Display the sequence that initializes the user's terminal in an implementation-
 35544 defined manner.

35545 **reset** Display the sequence that resets the user's terminal in an implementation-defined
 35546 manner.

35547 If a terminal does not support any of the operations described by these operands, this shall not
 35548 be considered an error condition.

35549 **STDIN**

35550 Not used.

35551 **INPUT FILES**

35552 None.

35553 **ENVIRONMENT VARIABLES**

35554 The following environment variables shall affect the execution of *tput*:

35555 **LANG** Provide a default value for the internationalization variables that are unset or null.
 35556 If *LANG* is unset or null, the corresponding value from the implementation-
 35557 defined default locale shall be used. If any of the internationalization variables
 35558 contains an invalid setting, the utility shall behave as if none of the variables had
 35559 been defined.

35560 **LC_ALL** If set to a non-empty string value, override the values of all the other
 35561 internationalization variables.

35562 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 35563 characters (for example, single-byte as opposed to multi-byte characters in
 35564 arguments).

35565 **LC_MESSAGES**

35566 Determine the locale that should be used to affect the format and contents of

35567 diagnostic messages written to standard error.

35568 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.

35569 **TERM** Determine the terminal type. If this variable is unset or null, and if the **-T** option is
35570 not specified, an unspecified default terminal type shall be used.

35571 **ASYNCHRONOUS EVENTS**

35572 Default.

35573 **STDOUT**

35574 If standard output is a terminal device, it may be used for writing the appropriate sequence to
35575 clear the screen or reset or initialize the terminal. If standard output is not a terminal device,
35576 undefined results occur.

35577 **STDERR**

35578 Used only for diagnostic messages.

35579 **OUTPUT FILES**

35580 None.

35581 **EXTENDED DESCRIPTION**

35582 None.

35583 **EXIT STATUS**

35584 The following exit values shall be returned:

35585 0 The requested string was written successfully.

35586 1 Unspecified.

35587 2 Usage error.

35588 3 No information is available about the specified terminal type.

35589 4 The specified operand is invalid.

35590 >4 An error occurred.

35591 **CONSEQUENCES OF ERRORS**

35592 If one of the operands is not available for the terminal, *tput* continues processing the remaining
35593 operands.

35594 **APPLICATION USAGE**

35595 The difference between resetting and initializing a terminal is left unspecified, as they vary
35596 greatly based on hardware types. In general, resetting is a more severe action.

35597 Some terminals use control characters to perform the stated functions, and on such terminals it
35598 might make sense to use *tput* to store the initialization strings in a file or environment variable
35599 for later use. However, because other terminals might rely on system calls to do this work, the
35600 standard output cannot be used in a portable manner, such as the following non-portable
35601 constructs:

35602 ClearVar='tput clear'
35603 tput reset | mailx -s "Wake Up" ddg

35604 **EXAMPLES**

35605 1. Initialize the terminal according to the type of terminal in the environmental variable
35606 *TERM*. This command can be included in a **.profile** file.

35607 tput init

35608 2. Reset a 450 terminal.

35609 `tput -T 450 reset`

35610 **RATIONALE**

35611 The list of operands was reduced to a minimum for the following reasons:

35612 • The only features chosen were those that were likely to be used by human users interacting
35613 with a terminal.

35614 • Specifying the full *terminfo* set was not considered desirable, but the standard developers did
35615 not want to select among operands.

35616 • This volume of IEEE Std. 1003.1-200x does not attempt to provide applications with
35617 sophisticated terminal handling capabilities, as that falls outside of its assigned scope and
35618 intersects with the responsibilities of other standards bodies.

35619 The difference between resetting and initializing a terminal is left unspecified as this varies
35620 greatly based on hardware types. In general, resetting is a more severe action.

35621 The exit status of 1 is historically reserved for finding out if a Boolean operand is not set.
35622 Although the operands were reduced to a minimum, the exit status of 1 should still be reserved
35623 for the Boolean operands, for those sites that wish to support them.

35624 **FUTURE DIRECTIONS**

35625 None.

35626 **SEE ALSO**

35627 *stty*, *tabs*

35628 **CHANGE HISTORY**

35629 First released in Issue 4.

35630 **Issue 6**

35631 This utility is now marked as part of the User Portability Utilities option.

35632 **NAME**

35633 tr — translate characters

35634 **SYNOPSIS**35635 tr [-c | -C][-s] *string1 string2*35636 tr -s [-c | -C] *string1*35637 tr -d [-c | -C] *string1*35638 tr -ds [-c | -C] *string1 string2*35639 **DESCRIPTION**

35640 The *tr* utility shall copy the standard input to the standard output with substitution or deletion
 35641 of selected characters. The options specified and the *string1* and *string2* operands shall control
 35642 translations that occur while copying characters and single-character collating elements.

35643 **OPTIONS**

35644 The *tr* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,
 35645 Utility Syntax Guidelines.

35646 The following options shall be supported:

35647 **-c** Complement the set of values specified by *string1*. See the EXTENDED
 35648 DESCRIPTION section.

35649 **-C** Complement the set of characters specified by *string1*. See the EXTENDED
 35650 DESCRIPTION section.

35651 **-d** Delete all occurrences of input characters that are specified by *string1*.

35652 **-s** Replace instances of repeated characters with a single character, as described in the
 35653 EXTENDED DESCRIPTION section.

35654 **OPERANDS**

35655 The following operands shall be supported:

35656 *string1, string2*

35657 Translation control strings. Each string shall represent a set of characters to be
 35658 converted into an array of characters used for the translation. For a detailed
 35659 description of how the strings are interpreted, see the EXTENDED DESCRIPTION
 35660 section.

35661 **STDIN**

35662 The standard input can be any type of file.

35663 **INPUT FILES**

35664 None.

35665 **ENVIRONMENT VARIABLES**

35666 The following environment variables shall affect the execution of *tr*:

35667 **LANG** Provide a default value for the internationalization variables that are unset or null.
 35668 If *LANG* is unset or null, the corresponding value from the implementation-
 35669 defined default locale shall be used. If any of the internationalization variables
 35670 contains an invalid setting, the utility shall behave as if none of the variables had
 35671 been defined.

35672 **LC_ALL** If set to a non-empty string value, override the values of all the other
 35673 internationalization variables.

35674	<i>LC_COLLATE</i>	
35675		Determine the locale for the behavior of range expressions and equivalence classes.
35676	<i>LC_CTYPE</i>	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and the behavior of character classes.
35677		
35678		
35679	<i>LC_MESSAGES</i>	
35680		Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
35681		
35682	XSI <i>NLSPATH</i>	Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
35683	ASYNCHRONOUS EVENTS	
35684		Default.
35685	STDOUT	
35686		The <i>tr</i> output shall be identical to the input, with the exception of the specified transformations.
35687	STDERR	
35688		Used only for diagnostic messages.
35689	OUTPUT FILES	
35690		None.
35691	EXTENDED DESCRIPTION	
35692		The operands <i>string1</i> and <i>string2</i> (if specified) define two arrays of characters. The constructs in the following list can be used to specify characters or single-character collating elements. If any of the constructs result in multi-character collating elements, <i>tr</i> shall exclude, without a diagnostic, those multi-character elements from the resulting array.
35693		
35694		
35695		
35696	<i>character</i>	Any character not described by one of the conventions below represents itself.
35697	<i>\octal</i>	Octal sequences can be used to represent characters with specific coded values. An octal sequence shall consist of a backslash followed by the longest sequence of one, two, or three-octal-digit characters (01234567). The sequence shall cause the value whose encoding is represented by the one, two, or three-digit octal integer to be placed into the array. If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation-defined. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading ' <i>\</i> ' for each byte.
35698		
35699		
35700		
35701		
35702		
35703		
35704		
35705	<i>\character</i>	The backslash-escape sequences in the Base Definitions volume of IEEE Std. 1003.1-200x, Table 5-1, Escape Sequences and Associated Actions (' <i>\</i> <i>\</i> ', ' <i>\a</i> ', ' <i>\b</i> ', ' <i>\f</i> ', ' <i>\n</i> ', ' <i>\r</i> ', ' <i>\t</i> ', ' <i>\v</i> ') shall be supported. The results of using any other character, other than an octal digit, following the backslash are unspecified.
35706		
35707		
35708		
35709		
35710	<i>c-c</i>	Represents the range of collating elements between the range endpoints (as long as neither endpoint is an octal sequence of the form <i>\octal</i>), inclusive, as defined by the current setting of the <i>LC_COLLATE</i> locale category. The application shall ensure that the starting endpoint precedes the second endpoint in the current collation order. The characters or collating elements in the range shall be placed in the array in ascending collation sequence.
35711		
35712		
35713		
35714		
35715		
35716		If either or both of the range endpoints are octal sequences of the form <i>\octal</i> , this shall represent the range of specific coded values between the two range endpoints, inclusive.
35717		
35718		

35719	<code>[:class:]</code>	Represents all characters belonging to the defined character class, as defined by the current setting of the <code>LC_CTYPE</code> locale category. The following character class names shall be accepted when specified in <code>string1</code> :
35720		
35721		
35722		alnum blank digit lower punct upper
35723		alpha cntrl graph print space xdigit
35724 XSI		In addition, character class expressions of the form <code>[:name:]</code> shall be recognized in those locales where the <code>name</code> keyword has been given a charclass definition in the <code>LC_CTYPE</code> category.
35725		
35726		
35727		When both the <code>-d</code> and <code>-s</code> options are specified, any of the character class names shall be accepted in <code>string2</code> . Otherwise, only character class names lower or upper are valid in <code>string2</code> and then only if the corresponding character class (upper and lower , respectively) is specified in the same relative position in <code>string1</code> . Such a specification shall be interpreted as a request for case conversion. When <code>[:lower:]</code> appears in <code>string1</code> and <code>[:upper:]</code> appears in <code>string2</code> , the arrays shall contain the characters from the toupper mapping in the <code>LC_CTYPE</code> category of the current locale. When <code>[:upper:]</code> appears in <code>string1</code> and <code>[:lower:]</code> appears in <code>string2</code> , the arrays shall contain the characters from the tolower mapping in the <code>LC_CTYPE</code> category of the current locale. The first character from each mapping pair shall be in the array for <code>string1</code> and the second character from each mapping pair shall be in the array for <code>string2</code> in the same relative position.
35728		
35729		
35730		
35731		
35732		
35733		
35734		
35735		
35736		
35737		
35738		
35739		Except for case conversion, the characters specified by a character class expression shall be placed in the array in an unspecified order.
35740		
35741		If the name specified for <code>class</code> does not define a valid character class in the current locale, the behavior is undefined.
35742		
35743	<code>[=equiv=]</code>	Represents all characters or collating elements belonging to the same equivalence class as <code>equiv</code> , as defined by the current setting of the <code>LC_COLLATE</code> locale category. An equivalence class expression shall be allowed only in <code>string1</code> , or in <code>string2</code> when it is being used by the combined <code>-d</code> and <code>-s</code> options. The characters belonging to the equivalence class shall be placed in the array in an unspecified order.
35744		
35745		
35746		
35747		
35748		
35749	<code>[x*n]</code>	Represents <code>n</code> repeated occurrences of the character <code>x</code> . Because this expression is used to map multiple characters to one, it is only valid when it occurs in <code>string2</code> . If <code>n</code> is omitted or is zero, it shall be interpreted as large enough to extend the <code>string2</code> -based sequence to the length of the <code>string1</code> -based sequence. If <code>n</code> has a leading zero, it shall be interpreted as an octal value. Otherwise, it shall be interpreted as a decimal value.
35750		
35751		
35752		
35753		
35754		
35755		When the <code>-d</code> option is not specified:
35756		• Each input character found in the array specified by <code>string1</code> shall be replaced by the character in the same relative position in the array specified by <code>string2</code> . When the array specified by <code>string2</code> is shorter than the one specified by <code>string1</code> , the results are unspecified.
35757		
35758		
35759		• If the <code>-C</code> option is specified, the complements of the characters specified by <code>string1</code> (the set of all characters in the current character set, as defined by the current setting of <code>LC_CTYPE</code> , except for those actually specified in the <code>string1</code> operand) shall be placed in the array in ascending collation sequence, as defined by the current setting of <code>LC_COLLATE</code> .
35760		
35761		
35762		
35763		• If the <code>-c</code> option is specified, the complement of the values specified by <code>string1</code> shall be placed in the array in ascending order by binary value.
35764		

35765 • Because the order in which characters specified by character class expressions or equivalence
 35766 class expressions is undefined, such expressions should only be used if the intent is to map
 35767 several characters into one. An exception is case conversion, as described previously.

35768 When the `-d` option is specified:

- 35769 • Input characters found in the array specified by *string1* shall be deleted.
- 35770 • When the `-C` option is specified with `-d`, all characters except those specified by *string1* shall
 35771 be deleted. The contents of *string2* are ignored, unless the `-s` option is also specified.
- 35772 • When the `-c` option is specified with `-d`, all values except those specified by *string1* shall be
 35773 deleted. The contents of *string2* shall be ignored, unless the `-s` option is also specified.
- 35774 • The same string cannot be used for both the `-d` and the `-s` option; when both options are
 35775 specified, both *string1* (used for deletion) and *string2* (used for squeezing) shall be required.

35776 When the `-s` option is specified, after any deletions or translations have taken place, repeated
 35777 sequences of the same character shall be replaced by one occurrence of the same character, if the
 35778 character is found in the array specified by the last operand. If the last operand contains a
 35779 character class, such as the following example:

```
35780 tr -s '[:space:]'
```

35781 the last operand's array shall contain all of the characters in that character class. However, in a
 35782 case conversion, as described previously, such as:

```
35783 tr -s '[:upper:]' '[:lower:]'
```

35784 the last operand's array shall contain only those characters defined as the second characters in
 35785 each of the **toupper** or **tolower** character pairs, as appropriate.

35786 An empty string used for *string1* or *string2* produces undefined results.

35787 EXIT STATUS

35788 The following exit values shall be returned:

35789 0 All input was processed successfully.

35790 >0 An error occurred.

35791 CONSEQUENCES OF ERRORS

35792 Default.

35793 APPLICATION USAGE

35794 If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

35795 If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must
 35796 use the full three digits to avoid ambiguity.

35797 When *string2* is shorter than *string1*, a difference results between historical System V and BSD
 35798 systems. A BSD system pads *string2* with the last character found in *string2*. Thus, it is possible
 35799 to do the following:

```
35800 tr 0123456789 d
```

35801 which would translate all digits to the letter 'd'. Since this area is specifically unspecified in
 35802 this volume of IEEE Std. 1003.1-200x, both the BSD and System V behaviors are allowed, but a
 35803 portable application cannot rely on the BSD behavior. It would have to code the example in the
 35804 following way:

```
35805 tr 0123456789 '[d*]'
```

35806 It should be noted that, despite similarities in appearance, the string operands used by *tr* are not
35807 regular expressions.

35808 Unlike some historical implementations, this definition of the *tr* utility correctly processes NUL
35809 characters in its input stream. NUL characters can be stripped by using:

```
35810 tr -d '\000'
```

35811 EXAMPLES

35812 1. The following example creates a list of all words in **file1** one per line in **file2**, where a word
35813 is taken to be a maximal string of letters.

```
35814 tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```

35815 2. The next example translates all lowercase characters in **file1** to uppercase and writes the
35816 results to standard output.

```
35817 tr "[:lower:]" "[:upper:]" <file1
```

35818 3. This example uses an equivalence class to identify accented variants of the base character
35819 'e' in **file1**, which are stripped of diacritical marks and written to **file2**.

```
35820 tr "[=e=]" e <file1 >file2
```

35821 RATIONALE

35822 In some early proposals, an explicit option **-n** was added to disable the historical behavior of
35823 stripping NUL characters from the input. It was considered that automatically stripping NUL
35824 characters from the input was not correct functionality. However, the removal of **-n** in a later
35825 proposal does not remove the requirement that *tr* correctly process NUL characters in its input
35826 stream. NUL characters can be stripped by using *tr -d '\000'*.

35827 Historical implementations of *tr* differ widely in syntax and behavior. For example, the BSD
35828 version has not needed the bracket characters for the repetition sequence. The POSIX Shell and
35829 Utilities *tr* syntax is based more closely on the System V and XPG3 model while attempting to
35830 accommodate historical BSD implementations. In the case of the short *string2* padding, the
35831 decision was to unspecified the behavior and preserve System V and XPG3 scripts, which might
35832 find difficulty with the BSD method. The assumption was made that BSD users of *tr* have to
35833 make accommodations to meet the POSIX Shell and Utilities syntax. Since it is possible to use
35834 the repetition sequence to duplicate the desired behavior, whereas there is no simple way to
35835 achieve the System V method, this was the correct, if not desirable, approach.

35836 The use of octal values to specify control characters, while having historical precedents, is not
35837 portable. The introduction of escape sequences for control characters should provide the
35838 necessary portability. It is recognized that this may cause some historical scripts to break.

35839 An early proposal included support for multi-character collating elements. It was pointed out
35840 that, while *tr* does employ some syntactical elements from REs, the aim of *tr* is quite different;
35841 ranges, for example, do not have a similar meaning (“any of the chars in the range matches”,
35842 *versus* “translate each character in the range to the output counterpart”). As a result, the
35843 previously included support for multi-character collating elements has been removed. What
35844 remains are ranges in current collation order (to support, for example, accented characters),
35845 character classes, and equivalence classes.

35846 In XPG3 the *[:class:]* and *[=equiv=]* conventions are shown with double brackets, as in RE syntax.
35847 However, *tr* does not implement RE principles; it just borrows part of the syntax. Consequently,
35848 *[:class:]* and *[=equiv=]* should be regarded as syntactical elements on a par with *[x*n]*, which is
35849 not an RE bracket expression.

- 35850 The standard developers will consider changes to *tr* that allow it to translate characters between
35851 different character encodings, or they will consider providing a new utility to accomplish this.
- 35852 On historical System V systems, a range expression requires enclosing square-brackets, such as:
35853 `tr '[a-z]' '[A-Z]'`
- 35854 However, BSD-based systems did not require the brackets, and this convention is used by POSIX
35855 Shell and Utilities to avoid breaking large numbers of BSD scripts:
- 35856 `tr a-z A-Z`
- 35857 The preceding System V script will continue to work because the brackets, treated as regular
35858 characters, are translated to themselves. However, any System V script that relied on *a-z*
35859 representing the three characters '-', ',' and 'z' have to be rewritten as *az-*.
- 35860 A prior version of IEEE Std. 1003.1-200x had a *-c* option that behaved similarly to the *-C* option,
35861 but did not supply functionality equivalent to the *-c* option specified in IEEE Std. 1003.1-200x.
35862 This meant that historical practice of being able to specify `tr -d\200-\377` (which would delete
35863 all bytes with the top bit set) would have no effect because, in the C locale, bytes with the values
35864 octal 200 to octal 377 are not characters.
- 35865 The earlier version also said that octal sequences referred to collating elements and could be
35866 placed adjacent to each other to specify multi-byte characters. However, it was noted that this
35867 caused ambiguities because *tr* would not be able to tell whether adjacent octal sequences were
35868 intending to specify multi-byte characters or multiple single byte characters.
35869 IEEE Std. 1003.1-200x specifies that octal sequences always refer to single byte binary values.
- 35870 **FUTURE DIRECTIONS**
- 35871 None.
- 35872 **SEE ALSO**
- 35873 *sed*
- 35874 **CHANGE HISTORY**
- 35875 First released in Issue 2.
- 35876 **Issue 4**
- 35877 Aligned with the ISO/IEC 9945-2:1993 standard.
- 35878 **Issue 6**
- 35879 The *-C* operand is added, and the description of the *-c* operand is changed to align with the
35880 IEEE P1003.2b draft standard.
- 35881 The normative text is reworded to avoid use of the term “must” for application requirements.

35882 **NAME**

35883 true — return true value

35884 **SYNOPSIS**

35885 true

35886 **DESCRIPTION**35887 The *true* utility shall return with exit code zero.35888 **OPTIONS**

35889 None.

35890 **OPERANDS**

35891 None.

35892 **STDIN**

35893 Not used.

35894 **INPUT FILES**

35895 None.

35896 **ENVIRONMENT VARIABLES**

35897 None.

35898 **ASYNCHRONOUS EVENTS**

35899 Default.

35900 **STDOUT**

35901 Not used.

35902 **STDERR**

35903 None.

35904 **OUTPUT FILES**

35905 None.

35906 **EXTENDED DESCRIPTION**

35907 None.

35908 **EXIT STATUS**

35909 Default.

35910 **CONSEQUENCES OF ERRORS**

35911 None.

35912 **APPLICATION USAGE**35913 This utility is typically used in shell scripts, as shown in the **EXAMPLES** section. The special
35914 built-in utility `:` is sometimes more efficient than *true*.35915 **EXAMPLES**

35916 This command is executed forever:

35917 while true
35918 do
35919 command
35920 done

35921 **RATIONALE**

35922 The *true* utility has been retained in this volume of IEEE Std. 1003.1-200x, even though the shell
35923 special built-in : provides similar functionality, because *true* is widely used in historical scripts
35924 and is less cryptic to novice script readers.

35925 **FUTURE DIRECTIONS**

35926 None.

35927 **SEE ALSO**

35928 *false*, Section 2.9 (on page 2256)

35929 **CHANGE HISTORY**

35930 First released in Issue 2.

35931 **Issue 4**

35932 Aligned with the ISO/IEC 9945-2:1993 standard.

35933 **NAME**

35934 tsort — topological sort

35935 **SYNOPSIS**35936 XSI tsort [*file*]

35937

35938 **DESCRIPTION**35939 The *tsort* utility shall write to standard output a totally ordered list of items consistent with a
35940 partial ordering of items contained in the input.35941 The application shall ensure that the input consists of pairs of items (non-empty strings)
35942 separated by <blank>s. Pairs of different items indicate ordering. Pairs of identical items
35943 indicate presence, but not ordering.35944 **OPTIONS**

35945 None.

35946 **OPERANDS**

35947 The following operand shall be supported:

35948 *file* A path name of a text file to order. If no *file* operand is given, the standard input is
35949 used.35950 **STDIN**35951 The standard input shall be a text file that is used if no *file* operand is given.35952 **INPUT FILES**35953 The input file named by the *file* operand is a text file.35954 **ENVIRONMENT VARIABLES**35955 The following environment variables shall affect the execution of *tsort*:35956 *LANG* Provide a default value for the internationalization variables that are unset or null.
35957 If *LANG* is unset or null, the corresponding value from the implementation-
35958 defined default locale shall be used. If any of the internationalization variables
35959 contains an invalid setting, the utility shall behave as if none of the variables had
35960 been defined.35961 *LC_ALL* If set to a non-empty string value, override the values of all the other
35962 internationalization variables.35963 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
35964 characters (for example, single-byte as opposed to multi-byte characters in
35965 arguments and input files).35966 *LC_MESSAGES*35967 Determine the locale that should be used to affect the format and contents of
35968 diagnostic messages written to standard error.35969 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.35970 **ASYNCHRONOUS EVENTS**

35971 Default.

35972 **STDOUT**35973 The standard output shall be a text file consisting of the order list produced from the partially
35974 ordered input.

35975 **STDERR**

35976 Used only for diagnostic messages.

35977 **OUTPUT FILES**

35978 None.

35979 **EXTENDED DESCRIPTION**

35980 None.

35981 **EXIT STATUS**

35982 The following exit values shall be returned:

35983 0 Successful completion.

35984 >0 An error occurred.

35985 **CONSEQUENCES OF ERRORS**

35986 Default.

35987 **APPLICATION USAGE**

35988 The *LC_COLLATE* variable need not affect the actions of *tsort*. The output ordering is not
35989 lexicographic, but depends on the pairs of items given as input.

35990 **EXAMPLES**

35991 The command:

35992 `tsort <<EOF`

35993 `a b c c d e`

35994 `g g`

35995 `f g e f`

35996 `h h`

35997 `EOF`

35998 produces the output:

35999 **a**

36000 **b**

36001 **c**

36002 **d**

36003 **e**

36004 **f**

36005 **g**

36006 **h**

36007 **RATIONALE**

36008 None.

36009 **FUTURE DIRECTIONS**

36010 None.

36011 **SEE ALSO**

36012 None.

36013 **CHANGE HISTORY**

36014 First released in Issue 2.

36015 **Issue 4**

36016 Format reorganized.

36017 Internationalized environment variable support mandated.

36018 **Issue 6**
36019

The normative text is reworded to avoid use of the term “must” for application requirements.

36020 **NAME**

36021 tty — return user's terminal name

36022 **SYNOPSIS**

36023 tty

36024 **DESCRIPTION**

36025 The *tty* utility shall write to the standard output the name of the terminal that is open as
 36026 standard input. The name that is used shall be equivalent to the string that would be returned by
 36027 the *ttyname()* function defined in the System Interfaces volume of IEEE Std. 1003.1-200x.

36028 **OPTIONS**

36029 The *tty* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 36030 12.2, Utility Syntax Guidelines.

36031 **OPERANDS**

36032 None.

36033 **STDIN**

36034 While no input is read from standard input, standard input shall be examined to determine
 36035 whether or not it is a terminal, and, if so, to determine the name of the terminal.

36036 **INPUT FILES**

36037 None.

36038 **ENVIRONMENT VARIABLES**36039 The following environment variables shall affect the execution of *tty*:

36040 *LANG* Provide a default value for the internationalization variables that are unset or null.
 36041 If *LANG* is unset or null, the corresponding value from the implementation-
 36042 defined default locale shall be used. If any of the internationalization variables
 36043 contains an invalid setting, the utility shall behave as if none of the variables had
 36044 been defined.

36045 *LC_ALL* If set to a non-empty string value, override the values of all the other
 36046 internationalization variables.

36047 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 36048 characters (for example, single-byte as opposed to multi-byte characters in
 36049 arguments).

36050 *LC_MESSAGES*

36051 Determine the locale that should be used to affect the format and contents of
 36052 diagnostic messages written to standard error and informative messages written to
 36053 standard output.

36054 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

36055 **ASYNCHRONOUS EVENTS**

36056 Default.

36057 **STDOUT**

36058 If standard input is a terminal device, a path name of the terminal as specified by the *ttyname()*
 36059 function defined in the System Interfaces volume of IEEE Std. 1003.1-200x shall be written in the
 36060 following format:

36061 "%s\n", <terminal name>

36062 Otherwise, a message shall be written indicating that standard input is not connected to a
 36063 terminal. In the POSIX locale, the *tty* utility shall use the format:

36064 "not a tty\n"

36065 **STDERR**

36066 Used only for diagnostic messages.

36067 **OUTPUT FILES**

36068 None.

36069 **EXTENDED DESCRIPTION**

36070 None.

36071 **EXIT STATUS**

36072 The following exit values shall be returned:

36073 0 Standard input is a terminal.

36074 1 Standard input is not a terminal.

36075 >1 An error occurred.

36076 **CONSEQUENCES OF ERRORS**

36077 Default.

36078 **APPLICATION USAGE**

36079 This utility checks the status of the file open as standard input against that of a system-defined
36080 set of files. It is possible that no match can be found, or that the match found need not be the
36081 same file as that which was opened for standard input (although they are the same device).

36082 The `-s` option is useful only if the exit code is wanted. It does not rely on the ability to form a
36083 valid path name. Portable applications should use `test -t 0`.

36084 **EXAMPLES**

36085 None.

36086 **RATIONALE**

36087 None.

36088 **FUTURE DIRECTIONS**

36089 None.

36090 **SEE ALSO**

36091 The System Interfaces volume of IEEE Std. 1003.1-200x, `isatty()`, `ttyname()`

36092 **CHANGE HISTORY**

36093 First released in Issue 2.

36094 **Issue 4**

36095 Aligned with the ISO/IEC 9945-2:1993 standard.

36096 **Issue 5**

36097 The SYNOPSIS is changed to indicate two forms of the command, with the second form marked
36098 as obsolete. This is a clarification and does not change the functionality published in previous
36099 issues.

36100 **NAME**

36101 type — write a description of command type

36102 **SYNOPSIS**

36103 xsi type name...

36104

36105 **DESCRIPTION**36106 The *type* utility shall indicate how each argument would be interpreted if used as a command
36107 name.36108 **OPTIONS**

36109 None.

36110 **OPERANDS**

36111 The following operand shall be supported:

36112 *name* A name to be interpreted.36113 **STDIN**

36114 Not used.

36115 **INPUT FILES**

36116 None.

36117 **ENVIRONMENT VARIABLES**36118 The following environment variables shall affect the execution of *type*:36119 *LANG* Provide a default value for the internationalization variables that are unset or null.
36120 If *LANG* is unset or null, the corresponding value from the implementation-
36121 defined default locale shall be used. If any of the internationalization variables
36122 contains an invalid setting, the utility shall behave as if none of the variables had
36123 been defined.36124 *LC_ALL* If set to a non-empty string value, override the values of all the other
36125 internationalization variables.36126 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
36127 characters (for example, single-byte as opposed to multi-byte characters in
36128 arguments).36129 *LC_MESSAGES*36130 Determine the locale that should be used to affect the format and contents of
36131 diagnostic messages written to standard error.36132 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.36133 *PATH* Determine the location of *name*, as described in the Base Definitions volume of
36134 IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.36135 **ASYNCHRONOUS EVENTS**

36136 Default.

36137 **STDOUT**36138 The standard output of *type* contains information about each operand in an unspecified format.
36139 The information provided typically identifies the operand as a shell built-in, function, alias, or
36140 keyword, and where applicable, may display the operand's path name.

36141 **STDERR**

36142 Used only for diagnostic messages.

36143 **OUTPUT FILES**

36144 None.

36145 **EXTENDED DESCRIPTION**

36146 None.

36147 **EXIT STATUS**

36148 The following exit values shall be returned:

36149 0 Successful completion.

36150 >0 An error occurred.

36151 **CONSEQUENCES OF ERRORS**

36152 Default.

36153 **APPLICATION USAGE**

36154 Since *type* must be aware of the contents of the current shell execution environment (such as the
36155 lists of commands, functions, and built-ins processed by *hash*), it is always provided as a shell
36156 regular built-in. If it is called in a separate utility execution environment, such as one of the
36157 following:

36158 `nohup type writer`36159 `find . -type f | xargs type`

36160 it might not produce accurate results.

36161 **EXAMPLES**

36162 None.

36163 **RATIONALE**

36164 None.

36165 **FUTURE DIRECTIONS**

36166 None.

36167 **SEE ALSO**36168 *command*36169 **CHANGE HISTORY**

36170 First released in Issue 2.

36171 **Issue 4**36172 Relocated from the *sh* description to reflect its status as a regular built-in utility.

36173 **NAME**36174 `ulimit` — set or report file size limit36175 **SYNOPSIS**36176 XSI `ulimit [-f][blocks]`

36177

36178 **DESCRIPTION**

36179 The *ulimit* utility shall set or report the file-size writing limit imposed on files written by the
 36180 shell and its child processes (files of any size may be read). Only a process with appropriate
 36181 privileges can increase the limit.

36182 **OPTIONS**

36183 The *ulimit* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 36184 12.2, Utility Syntax Guidelines.

36185 The following option shall be supported:

36186 `-f` Set (or report, if no *blocks* operand is present), the file size limit in blocks. The `-f`
 36187 option shall also be the default case.

36188 **OPERANDS**

36189 The following operand shall be supported:

36190 *blocks* The number of 512-byte blocks to use as the new file size limit.

36191 **STDIN**

36192 Not used.

36193 **INPUT FILES**

36194 None.

36195 **ENVIRONMENT VARIABLES**

36196 The following environment variables shall affect the execution of *ulimit*:

36197 *LANG* Provide a default value for the internationalization variables that are unset or null.
 36198 If *LANG* is unset or null, the corresponding value from the implementation-
 36199 defined default locale shall be used. If any of the internationalization variables
 36200 contains an invalid setting, the utility shall behave as if none of the variables had
 36201 been defined.

36202 *LC_ALL* If set to a non-empty string value, override the values of all the other
 36203 internationalization variables.

36204 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 36205 characters (for example, single-byte as opposed to multi-byte characters in
 36206 arguments).

36207 *LC_MESSAGES*

36208 Determine the locale that should be used to affect the format and contents of
 36209 diagnostic messages written to standard error.

36210 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

36211 **ASYNCHRONOUS EVENTS**

36212 Default.

36213 STDOUT

36214 The standard output shall be used when no *blocks* operand is present. If the current number of
36215 blocks is limited, the number of blocks in the current limit shall be written in the following
36216 format:

36217 "%d\n", <number of 512-byte blocks>

36218 If there is no current limit on the number of blocks, in the POSIX locale the following format
36219 shall be used:

36220 "unlimited\n"

36221 STDERR

36222 Used only for diagnostic messages.

36223 OUTPUT FILES

36224 None.

36225 EXTENDED DESCRIPTION

36226 None.

36227 EXIT STATUS

36228 The following exit values shall be returned:

36229 0 Successful completion.

36230 >0 A request for a higher limit was rejected or an error occurred.

36231 CONSEQUENCES OF ERRORS

36232 Default.

36233 APPLICATION USAGE

36234 Since *ulimit* affects the current shell execution environment, it is always provided as a shell
36235 regular built-in. If it is called in separate utility execution environment, such as one of the
36236 following:

36237 nohup ulimit -f 10000

36238 env ulimit 10000

36239 it does not affect the file size limit of the caller's environment.

36240 Once a limit has been decreased by a process, it cannot be increased (unless appropriate
36241 privileges are involved), even back to the original system limit.

36242 EXAMPLES

36243 Set the file size limit to 51 200 bytes:

36244 ulimit -f 100

36245 RATIONALE

36246 None.

36247 FUTURE DIRECTIONS

36248 None.

36249 SEE ALSO

36250 The System Interfaces volume of IEEE Std. 1003.1-200x, *ulimit()*

36251 CHANGE HISTORY

36252 First released in Issue 2.

36253 **Issue 4**

36254 Relocated from the *sh* description to reflect its status as a regular built-in utility.

36255 **NAME**

36256 umask — get or set the file mode creation mask

36257 **SYNOPSIS**36258 umask [-S][*mask*]36259 **DESCRIPTION**

36260 The *umask* utility shall set the file mode creation mask of the current shell execution
 36261 environment (see Section 2.13 (on page 2273)) to the value specified by the *mask* operand. This
 36262 mask shall affect the initial value of the file permission bits of subsequently created files. If *umask*
 36263 is called in a subshell or separate utility execution environment, such as one of the following:

36264 (umask 002)

36265 nohup umask ...

36266 find . -exec umask ... \;

36267 it shall not affect the file mode creation mask of the caller's environment.

36268 If the *mask* operand is not specified, the *umask* utility shall write to standard output the value of
 36269 the invoking process's file mode creation mask.

36270 **OPTIONS**

36271 The *umask* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 36272 12.2, Utility Syntax Guidelines.

36273 The following option shall be supported:

36274 **-S** Produce symbolic output.

36275 The default output style is unspecified, but shall be recognized on a subsequent invocation of
 36276 *umask* on the same system as a *mask* operand to restore the previous file mode creation mask.

36277 **OPERANDS**

36278 The following operand shall be supported:

36279 ***mask*** A string specifying the new file mode creation mask. The string is treated in the
 36280 same way as the *mode* operand described in the the EXTENDED DESCRIPTION
 36281 section for *chmod*.

36282 For a *symbolic_mode* value, the new value of the file mode creation mask shall be
 36283 the logical complement of the file permission bits portion of the file mode specified
 36284 by the *symbolic_mode* string.

36285 In a *symbolic_mode* value, the permissions *op* characters '+' and '-' shall be
 36286 interpreted relative to the current file mode creation mask; '+' shall cause the bits
 36287 for the indicated permissions to be cleared in the mask; '-' shall cause the bits for
 36288 the indicated permissions to be set in the mask.

36289 The interpretation of *mode* values that specify file mode bits other than the file
 36290 permission bits is unspecified.

36291 In the octal integer form of *mode*, the specified bits are set in the file mode creation
 36292 mask.

36293 The file mode creation mask shall be set to the resulting numeric value.

36294 The default output of a prior invocation of *umask* on the same system with no
 36295 operand also shall be recognized as a *mask* operand.

36296 **STDIN**

36297 Not used.

36298 **INPUT FILES**

36299 None.

36300 **ENVIRONMENT VARIABLES**36301 The following environment variables shall affect the execution of *umask*:

36302 *LANG* Provide a default value for the internationalization variables that are unset or null.
 36303 If *LANG* is unset or null, the corresponding value from the implementation-
 36304 defined default locale shall be used. If any of the internationalization variables
 36305 contains an invalid setting, the utility shall behave as if none of the variables had
 36306 been defined.

36307 *LC_ALL* If set to a non-empty string value, override the values of all the other
 36308 internationalization variables.

36309 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 36310 characters (for example, single-byte as opposed to multi-byte characters in
 36311 arguments).

36312 *LC_MESSAGES*

36313 Determine the locale that should be used to affect the format and contents of
 36314 diagnostic messages written to standard error.

36315 *XS* *NLS_PATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

36316 **ASYNCHRONOUS EVENTS**

36317 Default.

36318 **STDOUT**

36319 When the *mask* operand is not specified, the *umask* utility shall write a message to standard
 36320 output that can later be used as a *umask mask* operand.

36321 If *-S* is specified, the message shall be in the following format:

36322 "u=%s,g=%s,o=%s\n", <owner permissions>, <group permissions>,
 36323 <other permissions>

36324 where the three values shall be combinations of letters from the set {r, w, x}; the presence of a
 36325 letter shall indicate that the corresponding bit is clear in the file mode creation mask.

36326 If a *mask* operand is specified, there shall be no output written to standard output.36327 **STDERR**

36328 Used only for diagnostic messages.

36329 **OUTPUT FILES**

36330 None.

36331 **EXTENDED DESCRIPTION**

36332 None.

36333 **EXIT STATUS**

36334 The following exit values shall be returned:

36335 0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

36336 >0 An error occurred.

36337 CONSEQUENCES OF ERRORS

36338 Default.

36339 APPLICATION USAGE

36340 Since *umask* affects the current shell execution environment, it is generally provided as a shell
 36341 regular built-in.

36342 In contrast to the negative permission logic provided by the file mode creation mask and the
 36343 octal number form of the *mask* argument, the symbolic form of the *mask* argument specifies those
 36344 permissions that are left alone.

36345 EXAMPLES

36346 Either of the commands:

36347 `umask a=rx,ug+w`36348 `umask 002`

36349 sets the mode mask so that subsequently created files have their S_IWOTH bit cleared.

36350 After setting the mode mask with either of the above commands, the *umask* command can be
 36351 used to write out the current value of the mode mask:

36352 `$ umask`36353 `0002`

36354 (The output format is unspecified, but historical implementations use the octal integer mode
 36355 format.)

36356 `$ umask -S`36357 `u=rwx,g=rwx,o=rx`

36358 Either of these outputs can be used as the mask operand to a subsequent invocation of the *umask*
 36359 utility.

36360 Assuming the mode mask is set as above, the command:

36361 `umask g-w`

36362 sets the mode mask so that subsequently created files have their S_IWGRP and S_IWOTH bits
 36363 cleared.

36364 The command:

36365 `umask -- -w`

36366 sets the mode mask so that subsequently created files have all their write bits cleared. Note that
 36367 *mask* operands `-r`, `-w`, `-x` or anything beginning with a hyphen, must be preceded by `--` to
 36368 keep it from being interpreted as an option.

36369 RATIONALE

36370 Since *umask* affects the current shell execution environment, it is generally provided as a shell
 36371 regular built-in. If it is called in a subshell or separate utility execution environment, such as one
 36372 of the following:

36373 `(umask 002)`36374 `nohup umask ...`36375 `find . -exec umask ... \;`

36376 it does not affect the file mode creation mask of the environment of the caller.

36377 The description of the historical utility was modified to allow it to use the symbolic modes of
 36378 *chmod*. The `-s` option used in early proposals was changed to `-S` because `-s` could be confused

36379 with a *symbolic_mode* form of mask referring to the S_ISUID and S_ISGID bits.

36380 **Notes to Reviewers**

36381 *This section with side shading will not appear in the final copy. - Ed.*

36382 D1, XCU, ERN 355 suggests we should specify the default output. Suggestions please.

36383 The default output style is implementation-defined to permit implementors to provide
36384 migration to the new symbolic style at the time most appropriate to their users. An `-o` flag to
36385 force octal mode output was omitted because the octal mode may not be sufficient to specify all
36386 of the information that may be present in the file mode creation mask when more secure file
36387 access permission checks are implemented.

36388 It has been suggested that trusted systems developers might appreciate ameliorating the
36389 requirement that the mode mask “affects” the file access permissions, since it seems access
36390 control lists might replace the mode mask to some degree. The wording has been changed to say
36391 that it affects the file permission bits, and it leaves the details of the behavior of how they affect
36392 the file access permissions to the description in the System Interfaces volume of
36393 IEEE Std. 1003.1-200x.

36394 **FUTURE DIRECTIONS**

36395 None.

36396 **SEE ALSO**

36397 *chmod*, the System Interfaces volume of IEEE Std. 1003.1-200x, *umask()*

36398 **CHANGE HISTORY**

36399 First released in Issue 2.

36400 **Issue 4**

36401 Aligned with the ISO/IEC 9945-2:1993 standard.

36402 **Issue 6**

36403 The following new requirements on POSIX implementations derive from alignment with the
36404 Single UNIX Specification:

- 36405
- The octal mode is supported.

36406 **NAME**

36407 unalias — remove alias definitions

36408 **SYNOPSIS**36409 UP unalias *alias-name*...

36410 unalias -a

36411

36412 **DESCRIPTION**

36413 The *unalias* utility shall remove the definition for each alias name specified. See Section 2.3.1 (on
 36414 page 2239). The aliases shall be removed from the current shell execution environment; see
 36415 Section 2.13 (on page 2273).

36416 **OPTIONS**

36417 The *unalias* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 36418 12.2, Utility Syntax Guidelines.

36419 The following option shall be supported:

36420 **-a** Remove all alias definitions from the current shell execution environment.

36421 **OPERANDS**

36422 The following operand shall be supported:

36423 *alias-name* The name of an alias to be removed.

36424 **STDIN**

36425 Not used.

36426 **INPUT FILES**

36427 None.

36428 **ENVIRONMENT VARIABLES**

36429 The following environment variables shall affect the execution of *unalias*:

36430 **LANG** Provide a default value for the internationalization variables that are unset or null.
 36431 If **LANG** is unset or null, the corresponding value from the implementation-
 36432 defined default locale shall be used. If any of the internationalization variables
 36433 contains an invalid setting, the utility shall behave as if none of the variables had
 36434 been defined.

36435 **LC_ALL** If set to a non-empty string value, override the values of all the other
 36436 internationalization variables.

36437 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 36438 characters (for example, single-byte as opposed to multi-byte characters in
 36439 arguments).

36440 **LC_MESSAGES**

36441 Determine the locale that should be used to affect the format and contents of
 36442 diagnostic messages written to standard error.

36443 **XSI NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.

36444 **ASYNCHRONOUS EVENTS**

36445 Default.

36446 **STDOUT**

36447 Not used.

36448 **STDERR**

36449 Used only for diagnostic messages.

36450 **OUTPUT FILES**

36451 None.

36452 **EXTENDED DESCRIPTION**

36453 None.

36454 **EXIT STATUS**

36455 The following exit values shall be returned:

36456 0 Successful completion.

36457 >0 One of the *alias-name* operands specified did not represent a valid alias definition, or an
36458 error occurred.36459 **CONSEQUENCES OF ERRORS**

36460 Default.

36461 **APPLICATION USAGE**36462 Since *unalias* affects the current shell execution environment, it is generally provided as a shell
36463 regular built-in.36464 **EXAMPLES**

36465 None.

36466 **RATIONALE**36467 The *unalias* description is based on that from historical KornShell implementations. Known
36468 differences exist between that and the C shell. The KornShell version was adopted to be
36469 consistent with all the other KornShell features in this volume of IEEE Std. 1003.1-200x, such as
36470 command line editing.36471 The *-a* option is the equivalent of the *unalias ** form of the C shell and is provided to address
36472 security concerns about unknown aliases entering the environment of a user (or application)
36473 through the allowable implementation-defined predefined alias route or as a result of an *ENV*
36474 file. (Although *unalias* could be used to simplify the “secure” shell script shown in the *command*
36475 rationale, it does not obviate the need to quote all command names. An initial call to *unalias -a*
36476 would have to be quoted in case there was an alias for *unalias*.)36477 **FUTURE DIRECTIONS**

36478 None.

36479 **SEE ALSO**36480 *alias*36481 **CHANGE HISTORY**

36482 First released in Issue 4.

36483 **Issue 6**

36484 This utility is now marked as part of the User Portability Utilities option.

36485 **NAME**36486 **uname** — return system name36487 **SYNOPSIS**36488 **uname** [**-snrvma**]36489 **DESCRIPTION**

36490 By default, the *uname* utility shall write the operating system name to standard output. When
 36491 options are specified, symbols representing one or more system characteristics shall be written
 36492 to the standard output. The format and contents of the symbols are implementation-defined. On
 36493 systems conforming to the System Interfaces volume of IEEE Std. 1003.1-200x, the symbols
 36494 written shall be those supported by the *uname()* function as defined in the System Interfaces
 36495 volume of IEEE Std. 1003.1-200x.

36496 **OPTIONS**

36497 The *uname* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 36498 12.2, Utility Syntax Guidelines.

36499 The following options shall be supported:

- 36500 **-a** Behave as though all of the options **-mnrsv** were specified.
- 36501 **-m** Write the name of the hardware type on which the system is running to standard
 36502 output.
- 36503 **-n** Write the name of this node within an implementation-defined communications
 36504 network.
- 36505 **-r** Write the current release level of the operating system implementation.
- 36506 **-s** Write the name of the implementation of the operating system.
- 36507 **-v** Write the current version level of this release of the operating system
 36508 implementation.

36509 If no options are specified, the *uname* utility shall write the operating system name, as if the **-s**
 36510 option had been specified.

36511 **OPERANDS**

36512 None.

36513 **STDIN**

36514 Not used.

36515 **INPUT FILES**

36516 None.

36517 **ENVIRONMENT VARIABLES**

36518 The following environment variables shall affect the execution of *uname*:

- 36519 **LANG** Provide a default value for the internationalization variables that are unset or null.
 36520 If *LANG* is unset or null, the corresponding value from the implementation-
 36521 defined default locale shall be used. If any of the internationalization variables
 36522 contains an invalid setting, the utility shall behave as if none of the variables had
 36523 been defined.
- 36524 **LC_ALL** If set to a non-empty string value, override the values of all the other
 36525 internationalization variables.
- 36526 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 36527 characters (for example, single-byte as opposed to multi-byte characters in

36528 arguments).

36529 **LC_MESSAGES**

36530 Determine the locale that should be used to affect the format and contents of

36531 diagnostic messages written to standard error.

36532 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.

36533 **ASYNCHRONOUS EVENTS**

36534 Default.

36535 **STDOUT**

36536 By default, the output shall be a single line of the following form:

36537 "%s\n", <sysname>

36538 If the **-a** option is specified, the output shall be a single line of the following form:

36539 "%s %s %s %s %s\n", <sysname>, <nodename>, <release>,
36540 <version>, <machine>

36541 Additional implementation-defined symbols may be written; all such symbols shall be written at
36542 the end of the line of output before the <newline> character.

36543 If options are specified to select different combinations of the symbols, only those symbols shall
36544 be written, in the order shown above for the **-a** option. If a symbol is not selected for writing, its
36545 corresponding trailing <blank> characters also shall not be written.

36546 **STDERR**

36547 Used only for diagnostic messages.

36548 **OUTPUT FILES**

36549 None.

36550 **EXTENDED DESCRIPTION**

36551 None.

36552 **EXIT STATUS**

36553 The following exit values shall be returned:

36554 0 The requested information was successfully written.

36555 >0 An error occurred.

36556 **CONSEQUENCES OF ERRORS**

36557 Default.

36558 **APPLICATION USAGE**

36559 Note that any of the symbols could include embedded <space> characters, which may affect
36560 parsing algorithms if multiple options are selected for output.

36561 The node name is typically a name that the system uses to identify itself for intersystem
36562 communication addressing.

36563 **EXAMPLES**

36564 The following command:

36565 `uname -sr`

36566 writes the operating system name and release level, separated by one or more <blank>
36567 characters.

36568 RATIONALE

36569 It was suggested that this utility cannot be used portably since the format of the symbols is
36570 implementation-defined. The POSIX.1 working group could not achieve consensus on defining
36571 these formats in the underlying *uname()* function, and there was no expectation that this volume
36572 of IEEE Std. 1003.1-200x would be any more successful. Some applications may still find this
36573 historical utility of value. For example, the symbols could be used for system log entries or for
36574 comparison with operator or user input.

36575 FUTURE DIRECTIONS

36576 None.

36577 SEE ALSO

36578 The System Interfaces volume of IEEE Std. 1003.1-200x, *uname()*

36579 CHANGE HISTORY

36580 First released in Issue 2.

36581 Issue 4

36582 Aligned with the ISO/IEC 9945-2:1993 standard.

36583 Issue 4, Version 2

36584 The SYNOPSIS section lists all the valid options.

36585 **NAME**

36586 uncompress — expand compressed data

36587 **SYNOPSIS**36588 xSI uncompress [-cfv][*file...*]

36589

36590 **DESCRIPTION**

36591 The *uncompress* utility shall restore files to their original state after they have been compressed
 36592 using the *compress* utility. If no files are specified, the standard input shall be uncompressed to
 36593 the standard output. If the invoking process has appropriate privileges, the ownership, modes,
 36594 access time, and modification time of the original file shall be preserved.

36595 This utility shall support the uncompressing of any files produced by the *compress* utility on the
 36596 same implementation. For files produced by *compress* on other systems, *uncompress* supports 9 to
 36597 14-bit compression (see *compress* (on page 2477), **-b**); it is implementation-defined whether
 36598 values of **-b** greater than 14 are supported.

36599 **OPTIONS**

36600 The *uncompress* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,
 36601 Section 12.2, Utility Syntax Guidelines.

36602 The following options shall be supported:

- 36603 **-c** Write to standard output; no files are changed.
- 36604 **-f** Do not prompt for overwriting files. Except when run in the background, if **-f** is
 36605 not given the user shall be prompted as to whether an existing file should be
 36606 overwritten. If the standard input is not a terminal and **-f** is not given, *uncompress*
 36607 shall write a diagnostic message to standard error and exit with a status greater
 36608 than zero.
- 36609 **-v** Write messages to standard error concerning the expansion of each file.

36610 **OPERANDS**

36611 The following operand shall be supported:

- 36612 *file* A path name of a file. If *file* already has the **.Z** suffix specified, it shall be used as
 36613 the input file and the output file shall be named **file** with the **.Z** suffix removed.
 36614 Otherwise, *file* shall be used as the name of the output file and **file** with the **.Z**
 36615 suffix appended shall be used as the input file.

36616 **STDIN**

36617 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is **'-'**.

36618 **INPUT FILES**

36619 Input files shall be in the format produced by the *compress* utility.

36620 **ENVIRONMENT VARIABLES**

36621 The following environment variables shall affect the execution of *uncompress*:

- 36622 **LANG** Provide a default value for the internationalization variables that are unset or null.
 36623 If **LANG** is unset or null, the corresponding value from the implementation-
 36624 defined default locale shall be used. If any of the internationalization variables
 36625 contains an invalid setting, the utility shall behave as if none of the variables had
 36626 been defined.
- 36627 **LC_ALL** If set to a non-empty string value, override the values of all the other
 36628 internationalization variables.

- 36629 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
36630 characters (for example, single-byte as opposed to multi-byte characters in
36631 arguments).
- 36632 *LC_MESSAGES*
36633 Determine the locale that should be used to affect the format and contents of
36634 diagnostic messages written to standard error.
- 36635 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 36636 **ASYNCHRONOUS EVENTS**
36637 Default.
- 36638 **STDOUT**
36639 When there are no *file* operands or the *-c* option is specified, the uncompressed output is written
36640 to standard output.
- 36641 **STDERR**
36642 Prompts shall be written to the standard error output under the conditions specified in the
36643 DESCRIPTION and OPTIONS sections. The prompts shall contain the *file* path name, but their
36644 format is otherwise unspecified. Otherwise, the standard error output shall be used only for
36645 diagnostic messages.
- 36646 **OUTPUT FILES**
36647 Output files are the same as the respective input files to *compress*.
- 36648 **EXTENDED DESCRIPTION**
36649 None.
- 36650 **EXIT STATUS**
36651 The following exit values shall be returned:
36652 0 Successful completion.
36653 >0 An error occurred.
- 36654 **CONSEQUENCES OF ERRORS**
36655 The input file remains unmodified.
- 36656 **APPLICATION USAGE**
36657 The limit of 14 on the *compress -b bits* argument is to achieve portability to all systems (within
36658 the restrictions imposed by the lack of an explicit published file format). Some systems based on
36659 16-bit architectures cannot support 15 or 16-bit uncompression.
- 36660 **EXAMPLES**
36661 None.
- 36662 **RATIONALE**
36663 None.
- 36664 **FUTURE DIRECTIONS**
36665 None.
- 36666 **SEE ALSO**
36667 *compress, zcat*
- 36668 **CHANGE HISTORY**
36669 First released in Issue 4.

36670 **Issue 4, Version 2**

36671 The DESCRIPTION is clarified to state that the ownership, modes, access time, and modification
36672 time of the original file are preserved if the invoking process has appropriate privileges.

36673 **Issue 6**

36674 The normative text is reworded to avoid use of the term “must” for application requirements.

36675 NAME

36676 unexpand — convert spaces to tabs

36677 SYNOPSIS

36678 UP unexpand [-a | -t *tablist*][*file...*]

36679

36680 DESCRIPTION

36681 The *unexpand* utility shall copy files or standard input to standard output, converting <blank>
 36682 characters at the beginning of each line into the maximum number of <tab> characters followed
 36683 by the minimum number of <space> characters needed to fill the same column positions
 36684 originally filled by the translated <blank> characters. By default, tabstops shall be set at every
 36685 eighth column position. Each <backspace> character shall be copied to the output, and shall
 36686 cause the column position count for tab calculations to be decremented; the count shall never be
 36687 decremented to a value less than one.

36688 OPTIONS

36689 The *unexpand* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,
 36690 Section 12.2, Utility Syntax Guidelines.

36691 The following options shall be supported:

36692 **-a** In addition to translating <blank> characters at the beginning of each line, translate
 36693 all sequences of two or more <blank> characters immediately preceding a tab stop
 36694 to the maximum number of <tab> characters followed by the minimum number of
 36695 <space> characters needed to fill the same column positions originally filled by the
 36696 translated <blank> characters.

36697 **-t *tablist*** Specify the tab stops. The application shall ensure that the *tablist* option-argument
 36698 is a single argument consisting of a single positive decimal integer or multiple
 36699 positive decimal integers, separated by <blank> characters or commas, in
 36700 ascending order. If a single number is given, tabs shall be set *tablist* column
 36701 positions apart instead of the default 8. If multiple numbers are given, the tabs
 36702 shall be set at those specific column positions.

36703 The application shall ensure that each tab-stop position *N* is an integer value
 36704 greater than zero, and the list shall be in strictly ascending order. This is taken to
 36705 mean that, from the start of a line of output, tabbing to position *N* shall cause the
 36706 next character output to be in the (*N*+1)th column position on that line. When the
 36707 **-t** option is not specified, the default shall be the equivalent of specifying **-t 8**
 36708 (except for the interaction with **-a**, described below).

36709 No <space>-to-<tab> character conversions shall occur for characters at positions
 36710 beyond the last of those specified in a multiple tab-stop list.

36711 When **-t** is specified, the presence or absence of the **-a** option shall be ignored;
 36712 conversion shall not be limited to the processing of leading <blank> characters.

36713 OPERANDS

36714 The following operand shall be supported:

36715 *file* A path name of a text file to be used as input.

36716 STDIN

36717 See the INPUT FILES section.

36718 **INPUT FILES**

36719 The input files shall be text files.

36720 **ENVIRONMENT VARIABLES**36721 The following environment variables shall affect the execution of *unexpand*:

36722 *LANG* Provide a default value for the internationalization variables that are unset or null.
 36723 If *LANG* is unset or null, the corresponding value from the implementation-
 36724 defined default locale shall be used. If any of the internationalization variables
 36725 contains an invalid setting, the utility shall behave as if none of the variables had
 36726 been defined.

36727 *LC_ALL* If set to a non-empty string value, override the values of all the other
 36728 internationalization variables.

36729 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 36730 characters (for example, single-byte as opposed to multi-byte characters in
 36731 arguments and input files), the processing of <tab> and <space> characters and for
 36732 the determination of the width in column positions each character would occupy
 36733 on an output device.

36734 *LC_MESSAGES*

36735 Determine the locale that should be used to affect the format and contents of
 36736 diagnostic messages written to standard error.

36737 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

36738 **ASYNCHRONOUS EVENTS**

36739 Default.

36740 **STDOUT**

36741 The standard output is equivalent to the input files with the specified <space>-to-<tab>
 36742 character conversions.

36743 **STDERR**

36744 Used only for diagnostic messages.

36745 **OUTPUT FILES**

36746 None.

36747 **EXTENDED DESCRIPTION**

36748 None.

36749 **EXIT STATUS**

36750 The following exit values shall be returned:

36751 0 Successful completion.

36752 >0 An error occurred.

36753 **CONSEQUENCES OF ERRORS**

36754 Default.

36755 APPLICATION USAGE

36756 One non-intuitive aspect of *unexpand* is its restriction to leading spaces when neither **-a** nor **-t** is
36757 specified. Users who desire to always convert all spaces in a file can easily alias *unexpand* to use
36758 the **-a** or **-t 8** option.

36759 EXAMPLES

36760 None.

36761 RATIONALE

36762 On several occasions, consideration was given to adding a **-t** option to the *unexpand* utility to
36763 complement the **-t** in *expand* (see *expand* (on page 2636)). The historical intent of *unexpand* was
36764 to translate multiple <blank>s into tab stops, where tab stops were a multiple of eight column
36765 positions on most UNIX systems. An early proposal omitted **-t** because it seemed outside the
36766 scope of the UPE; it was not described in any of the base documents. However, hard-coding tab
36767 stops every eight columns was not suitable for the international community and broke historical
36768 precedents for some vendors in the FORTRAN community, so **-t** was restored in conjunction
36769 with the list of valid extension categories considered by the standard developers. Thus, *unexpand*
36770 is now the logical converse of *expand*.

36771 FUTURE DIRECTIONS

36772 None.

36773 SEE ALSO

36774 *expand, tabs*

36775 CHANGE HISTORY

36776 First released in Issue 4.

36777 Issue 6

36778 This utility is now marked as part of the User Portability Utilities option.

36779 The definition of the *LC_CTYPE* environment variable is changed to align with the
36780 IEEE P1003.2b draft standard.

36781 The normative text is reworded to avoid use of the term “must” for application requirements.

36782 **NAME**36783 unget — undo a previous get of an SCCS file (**DEVELOPMENT**)36784 **SYNOPSIS**36785 xSI unget [-ns][-r *SID*] *file...*

36786

36787 **DESCRIPTION**36788 The *unget* utility shall reverse the effect of a *get -e* done prior to creating the intended new delta.36789 **OPTIONS**36790 The *unget* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
36791 12.2, Utility Syntax Guidelines.

36792 The following options shall be supported:

36793 **-r** *SID* Uniquely identify which delta is no longer intended. (This would have been
36794 specified by *get* as the new delta.) The use of this option is necessary only if two or
36795 more outstanding *get* commands for editing on the same SCCS file were done by
36796 the same person (login name).36797 **-s** Suppress the writing to standard output of the intended delta's SID.36798 **-n** Retain the file that was obtained by *get*, which would normally be removed from
36799 the current directory.36800 **OPERANDS**

36801 The following operands shall be supported:

36802 **file** A path name of an existing SCCS file or a directory. If *file* is a directory, the *unget*
36803 utility shall behave as though each file in the directory were specified as a named
36804 file, except that non-SCCS files (last component of the path name does not begin
36805 with **s.**) and unreadable files shall be silently ignored.36806 If a single instance *file* is specified as **'-'**, the standard input shall be read; each
36807 line of the standard input shall be taken to be the name of an SCCS file to be
36808 processed. Non-SCCS files and unreadable files shall be silently ignored.36809 **STDIN**36810 The standard input shall be a text file used only when the *file* operand is specified as **'-'**. Each
36811 line of the text file shall be interpreted as an SCCS path name.36812 **INPUT FILES**

36813 Any SCCS files processed are files of an unspecified format.

36814 **ENVIRONMENT VARIABLES**36815 The following environment variables shall affect the execution of *unget*:36816 **LANG** Provide a default value for the internationalization variables that are unset or null.
36817 If **LANG** is unset or null, the corresponding value from the implementation-
36818 defined default locale shall be used. If any of the internationalization variables
36819 contains an invalid setting, the utility shall behave as if none of the variables had
36820 been defined.36821 **LC_ALL** If set to a non-empty string value, override the values of all the other
36822 internationalization variables.36823 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
36824 characters (for example, single-byte as opposed to multi-byte characters in
36825 arguments and input files).

- 36826 **LC_MESSAGES**
36827 Determine the locale that should be used to affect the format and contents of
36828 diagnostic messages written to standard error.
- 36829 **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 36830 **ASYNCHRONOUS EVENTS**
36831 Default.
- 36832 **STDOUT**
36833 The standard output shall consist of a line for each file, in the following format:
36834 "%s\n", <*SID removed from file*>
36835 If there is more than one named file or if a directory or standard input is named, each path name
36836 shall be written before each of the preceding lines:
36837 "\n%s:\n", <*pathname*>
- 36838 **STDERR**
36839 Used only for diagnostic messages.
- 36840 **OUTPUT FILES**
36841 Any SCCS files updated are files of an unspecified format. During processing of a *file*, a locking
36842 *z-file*, as described in *get*, and a *q-file* (a working copy of the *p-file*), may be created and deleted.
36843 The *p-file* and *g-file*, as described in *get*, shall be deleted.
- 36844 **EXTENDED DESCRIPTION**
36845 None.
- 36846 **EXIT STATUS**
36847 The following exit values shall be returned:
36848 0 Successful completion.
36849 >0 An error occurred.
- 36850 **CONSEQUENCES OF ERRORS**
36851 Default.
- 36852 **APPLICATION USAGE**
36853 None.
- 36854 **EXAMPLES**
36855 None.
- 36856 **RATIONALE**
36857 None.
- 36858 **FUTURE DIRECTIONS**
36859 None.
- 36860 **SEE ALSO**
36861 *delta, get, sact*
- 36862 **CHANGE HISTORY**
36863 First released in Issue 2.
36864 **Issue 4**
36865 Format reorganized.
36866 Utility Syntax Guidelines support mandated.

36867 Internationalized environment variable support mandated.

36868 **Issue 6**

36869 The normative text is reworded to avoid use of the term “must” for application requirements. |

36870 The normative text is reworded to emphasise the term “shall” for implementation requirements. |

36871 **NAME**

36872 un~~iq~~ — report or filter out repeated lines in a file

36873 **SYNOPSIS**

36874 un~~iq~~ [-c|-d|-u][-f *fields*][-s *char*][*input_file* [*output_file*]]

36875 **DESCRIPTION**

36876 The *un~~iq~~* utility shall read an input file comparing adjacent lines, and writes one copy of each
36877 input line on the output. The second and succeeding copies of repeated adjacent input lines shall
36878 not be written.

36879 Repeated lines in the input shall not be detected if they are not adjacent.

36880 **OPTIONS**

36881 The *un~~iq~~* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
36882 12.2, Utility Syntax Guidelines.

36883 The following options shall be supported:

36884 -c Precede each output line with a count of the number of times the line occurred in
36885 the input.

36886 -d Suppress the writing of lines that are not repeated in the input.

36887 -f *fields* Ignore the first *fields* fields on each input line when doing comparisons, where
36888 *fields* is a positive decimal integer. A field is the maximal string matched by the
36889 basic regular expression:

36890 [[:blank:]]*[^[:blank:]]*

36891 If the *fields* option-argument specifies more fields than appear on an input line, a
36892 null string shall be used for comparison.

36893 -s *chars* Ignore the first *chars* characters when doing comparisons, where *chars* shall be a
36894 positive decimal integer. If specified in conjunction with the -f option, the first
36895 *chars* characters after the first *fields* fields shall be ignored. If the *chars* option-
36896 argument specifies more characters than remain on an input line, a null string shall
36897 be used for comparison.

36898 -u Suppress the writing of lines that are repeated in the input.

36899 **OPERANDS**

36900 The following operands shall be supported:

36901 *input_file* A path name of the input file. If the *input_file* operand is not specified, or if the
36902 *input_file* is '-', the standard input is used.

36903 *output_file* A path name of the output file. If the *output_file* operand is not specified, the
36904 standard output shall be used. The results are unspecified if the file named by
36905 *output_file* is the file named by *input_file*.

36906 **STDIN**

36907 The standard input shall be used only if no *input_file* operand is specified or if *input_file* is '-'.
36908 See the INPUT FILES section.

36909 **INPUT FILES**

36910 The input file shall be a text file.

36911 **ENVIRONMENT VARIABLES**

36912 The following environment variables shall affect the execution of *uniq*:

36913 *LANG* Provide a default value for the internationalization variables that are unset or null.
 36914 If *LANG* is unset or null, the corresponding value from the implementation-
 36915 defined default locale shall be used. If any of the internationalization variables
 36916 contains an invalid setting, the utility shall behave as if none of the variables had
 36917 been defined.

36918 *LC_ALL* If set to a non-empty string value, override the values of all the other
 36919 internationalization variables.

36920 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 36921 characters (for example, single-byte as opposed to multi-byte characters in
 36922 arguments and input files) which characters constitute a <blank> character in the
 36923 current locale.

36924 *LC_MESSAGES*
 36925 Determine the locale that should be used to affect the format and contents of
 36926 diagnostic messages written to standard error.

36927 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

36928 **ASYNCHRONOUS EVENTS**

36929 Default.

36930 **STDOUT**

36931 The standard output shall be used only if no *output_file* operand is specified. See the OUTPUT
 36932 FILES section.

36933 **STDERR**

36934 Used only for diagnostic messages.

36935 **OUTPUT FILES**

36936 If the *-c* option is specified, the application shall ensure that the output file is empty or each line
 36937 shall be of the form:

36938 "%d %s", <number of duplicates>, <line>

36939 otherwise, the application shall ensure that the output file is empty or each line shall be of the
 36940 form:

36941 "%s", <line>

36942 **EXTENDED DESCRIPTION**

36943 None.

36944 **EXIT STATUS**

36945 The following exit values shall be returned:

36946 0 The utility executed successfully.

36947 >0 An error occurred.

36948 **CONSEQUENCES OF ERRORS**

36949 Default.

36950 **APPLICATION USAGE**

36951 The *sort* utility can be used to cause repeated lines to be adjacent in the input file.

36952 **EXAMPLES**

36953 The following input file data (but flushed left) was used for a test series on *uniqu*:

```
36954 #01 foo0 bar0 fool bar1
36955 #02 bar0 fool bar1 fool
36956 #03 foo0 bar0 fool bar1
36957 #04
36958 #05 foo0 bar0 fool bar1
36959 #06 foo0 bar0 fool bar1
36960 #07 bar0 fool bar1 foo0
```

36961 What follows is a series of test invocations of the *uniqu* utility that use a mixture of *uniqu* options
36962 against the input file data. These tests verify the meaning of *adjacent*. The *uniqu* utility views the
36963 input data as a sequence of strings delimited by '\n'. Accordingly, for the *fieldsth* member of
36964 the sequence, *uniqu* interprets unique or repeated adjacent lines strictly relative to the *fields+1*th
36965 member.

36966 1. This first example tests the line counting option, comparing each line of the input file data
36967 starting from the second field:

```
36968      uniqu -c -f 1 uniqu_0I.t
36969          1 #01 foo0 bar0 fool bar1
36970          1 #02 bar0 fool bar1 foo0
36971          1 #03 foo0 bar0 fool bar1
36972          1 #04
36973          2 #05 foo0 bar0 fool bar1
36974          1 #07 bar0 fool bar1 foo0
```

36975 The number '2', prefixing the fifth line of output, signifies that the *uniqu* utility detected a
36976 pair of repeated lines. Given the input data, this can only be true when *uniqu* is run using
36977 the *-f 1* option (which shall cause *uniqu* to ignore the first field on each input line).

36978 2. The second example tests the option to suppress unique lines, comparing each line of the
36979 input file data starting from the second field:

```
36980      uniqu -d -f 1 uniqu_0I.t
36981      #05 foo0 bar0 fool bar1
```

36982 3. This test suppresses repeated lines, comparing each line of the input file data starting from
36983 the second field:

```
36984      uniqu -u -f 1 uniqu_0I.t
36985      #01 foo0 bar0 fool bar1
36986      #02 bar0 fool bar1 fool
36987      #03 foo0 bar0 fool bar1
36988      #04
36989      #07 bar0 fool bar1 foo0
```

36990 4. This suppresses unique lines, comparing each line of the input file data starting from the
36991 third character:

```
36992      uniqu -d -s 2 uniqu_0I.t
```

36993 In the last example, the *uniqu* utility found no input matching the above criteria.

36994 **RATIONALE**

36995 Some historical implementations have limited lines to be 1 080 bytes in length, which does not
36996 meet the implied {LINE_MAX} limit.

36997 **FUTURE DIRECTIONS**

36998 None.

36999 **SEE ALSO**

37000 *comm, sort*

37001 **CHANGE HISTORY**

37002 First released in Issue 2.

37003 **Issue 4**

37004 Aligned with the ISO/IEC 9945-2: 1993 standard.

37005 **Issue 6**

37006 The obsolescent SYNOPSIS and associated text are removed.

37007 The normative text is reworded to avoid use of the term “must” for application requirements.

37008 **NAME**

37009 unlink — call the *unlink()* function

37010 **SYNOPSIS**

37011 xSI unlink *file*

37012

37013 **DESCRIPTION**

37014 The *unlink* utility shall perform the function call:

37015 unlink(*file*);

37016 A user may need appropriate privilege to invoke the *unlink* utility.

37017 **OPTIONS**

37018 None.

37019 **OPERANDS**

37020 The following operands shall be supported:

37021 *file* The path name of an existing file.

37022 **STDIN**

37023 Not used.

37024 **INPUT FILES**

37025 Not used.

37026 **ENVIRONMENT VARIABLES**

37027 The following environment variables shall affect the execution of *unlink*:

37028 *LANG* Provide a default value for the internationalization variables that are unset or null.
 37029 If *LANG* is unset or null, the corresponding value from the implementation-
 37030 defined default locale shall be used. If any of the internationalization variables
 37031 contain an invalid setting, the utility behaves as if none of the variables had been
 37032 set.

37033 *LC_ALL* If set to a non-empty string value, override the values of all the other
 37034 internationalization variables.

37035 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 37036 characters (for example, single-byte as opposed to multi-byte characters in
 37037 arguments).

37038 *LC_MESSAGES*
 37039 Determine the locale that should be used to affect the format and contents of
 37040 diagnostic messages written to standard error.

37041 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

37042 **ASYNCHRONOUS EVENTS**

37043 Default.

37044 **STDOUT**

37045 None.

37046 **STDERR**

37047 Used only for diagnostic messages.

37048 **OUTPUT FILES**

37049 None.

37050 **EXTENDED DESCRIPTION**

37051 None.

37052 **EXIT STATUS**

37053 The following exit values shall be returned:

37054 0 Successful completion.

37055 >0 An error occurred.

37056 **CONSEQUENCES OF ERRORS**

37057 Default.

37058 **APPLICATION USAGE**

37059 None.

37060 **EXAMPLES**

37061 None.

37062 **RATIONALE**

37063 None.

37064 **FUTURE DIRECTIONS**

37065 None.

37066 **SEE ALSO**37067 *link*, *rm*, the System Interfaces volume of IEEE Std. 1003.1-200x, *unlink()*37068 **CHANGE HISTORY**

37069 First released in Issue 5.

37070 NAME

37071 uucp — system-to-system copy

37072 SYNOPSIS

37073 UN XSI uucp [-cCdfjmr][-n user] source-file... destination-file

37074

37075 DESCRIPTION

37076 The *uucp* utility shall copy files named by the *source-file* arguments to the *destination-file*
37077 argument. The files named can be on local or remote systems.37078 The *uucp* utility cannot guarantee support for all character encodings in all circumstances. For
37079 example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and
37080 file names need not be portable to non-internationalized systems, and so on. Under these
37081 circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991
37082 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used,
37083 and that only characters defined in the Portable File Name Character Set be used for naming
37084 files. The protocol for transfer of files is unspecified by IEEE Std. 1003.1-200x.

37085 OPTIONS

37086 The *uucp* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
37087 12.2, Utility Syntax Guidelines.

37088 The following options shall be supported:

37089 -c Do not copy local file to the spool directory for transfer to the remote machine
37090 (default).

37091 UN -C Force the copy of local files to the spool directory for transfer.

37092 -d Make all necessary directories for the file copy (default).

37093 UN -f Do not make intermediate directories for the file copy.

37094 UN -j Write the job identification string to standard output. This job identification can be
37095 used by *uustat* to obtain the status or terminate a job.

37096 -m Send mail to the requester when the copy is completed.

37097 UN -n user Notify *user* on the remote system that a file was sent.

37098 UN -r Do not start the file transfer; just queue the job.

37099 OPERANDS

37100 The following operands shall be supported:

37101 *destination-file*, *source-file*37102 A path name of a file to be copied to, or from, respectively. Either name can be a
37103 path name on the local machine, or can have the form:37104 *system-name!pathname*37105 where *system-name* is taken from a list of system names that *uucp* knows about.37106 The destination *system-name* can also be a list of names such as:37107 *system-name!system-name!...!system-name!pathname*37108 in which case, an attempt is made to send the file via the specified route to the
37109 destination. Care should be taken to ensure that intermediate nodes in the route
37110 are willing to forward information.

37111 The shell pattern matching notation characters '?', '*', and "[...]" appearing
 37112 in *pathname* are expanded on the appropriate system.

37113 Path names can be one of:

37114 1. An absolute path name.

37115 2. A path name preceded by *~user* where *user* is a login name on the specified
 37116 system and is replaced by that user's login directory. Note that if an invalid
 37117 login is specified, the default is to the public directory (called *PUBDIR*; the
 37118 actual location of *PUBDIR* is implementation-defined).

37119 3. A path name preceded by *~/destination* where *destination* is appended to
 37120 *PUBDIR*.

37121 **Note:** This destination is treated as a file name unless more than one file
 37122 is being transferred by this request or the destination is already a
 37123 directory. To ensure that it is a directory, follow the destination
 37124 with a '/'. For example, *~/dan/* as the destination makes the
 37125 directory **PUBDIR/dan** if it does not exist and put the requested
 37126 files in that directory.

37127 4. Anything else is prefixed by the current directory.

37128 If the result is an erroneous path name for the remote system, the copy fails. If the
 37129 *destination-file* is a directory, the last part of the *source-file* name is used.

37130 The read, write, and execute permissions given by *uucp* are implementation-
 37131 defined.

37132 **STDIN**

37133 Not used.

37134 **INPUT FILES**

37135 The files to be copied are regular files.

37136 **ENVIRONMENT VARIABLES**

37137 The following environment variables shall affect the execution of *uucp*:

37138 *LANG* Provide a default value for the internationalization variables that are unset or null.
 37139 If *LANG* is unset or null, the corresponding value from the implementation-
 37140 defined default locale shall be used. If any of the internationalization variables
 37141 contains an invalid setting, the utility shall behave as if none of the variables had
 37142 been defined.

37143 *LC_ALL* If set to a non-empty string value, override the values of all the other
 37144 internationalization variables.

37145 *LC_COLLATE*

37146 Determine the locale for the behavior of ranges, equivalence classes and multi-
 37147 character collating elements within bracketed file name patterns.

37148 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 37149 characters (for example, single-byte as opposed to multi-byte characters in
 37150 arguments and input files) and the behavior of character classes within bracketed
 37151 file name patterns (for example, "[[:lower:]]*").

37152 *LC_MESSAGES*

37153 Determine the locale that should be used to affect the format and contents of
 37154 diagnostic messages written to standard error, and informative messages written

- 37155 to standard output.
- 37156 *LC_TIME* Determine the format of date and time strings output by *uucp*.
- 37157 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 37158 *TZ* Determine the timezone used with date and time strings.
- 37159 **ASYNCHRONOUS EVENTS**
- 37160 Default.
- 37161 **STDOUT**
- 37162 Not used.
- 37163 **STDERR**
- 37164 Used only for diagnostic messages.
- 37165 **OUTPUT FILES**
- 37166 The output files (which may be on other systems) are copies of the input files.
- 37167 If the *-m* is used, mail files are modified.
- 37168 **EXTENDED DESCRIPTION**
- 37169 None.
- 37170 **EXIT STATUS**
- 37171 The following exit values shall be returned:
- 37172 0 Successful completion.
- 37173 >0 An error occurred.
- 37174 **CONSEQUENCES OF ERRORS**
- 37175 Default.
- 37176 **APPLICATION USAGE**
- 37177 The domain of remotely accessible files can (and for obvious security reasons usually should) be severely restricted.
- 37178
- 37179 Note that the *'!*' character in addresses has to be escaped when using *cs**h* as a command interpreter because of its history substitution syntax. For *ksh* and *sh* the escape is not necessary, but may be used.
- 37180
- 37181
- 37182 Typical implementations of this utility require a communications line configured to use the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface, but other communications means may be used. On systems where there are no available communications means (either temporarily or permanently), this utility shall write an error message describing the problem and exit with a non-zero exit status.
- 37183
- 37184
- 37185
- 37186
- 37187 As noted above, shell metacharacters appearing in path names are expanded on the appropriate system. On an internationalized system, this is done under the control of local settings of *LC_COLLATE* and *LC_CTYPE*. Thus, care should be taken when using bracketed file name patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes, and collating symbols) need not be supported on non-internationalized systems.
- 37188
- 37189
- 37190
- 37191
- 37192
- 37193 **EXAMPLES**
- 37194 None.

37195 **RATIONALE**

37196 None.

37197 **FUTURE DIRECTIONS**

37198 None.

37199 **SEE ALSO**37200 *mailx, uuencode, uustat, uux*37201 **CHANGE HISTORY**

37202 First released in Issue 2.

37203 **Issue 4**

37204 Format reorganized.

37205 Split into a separate description.

37206 Utility Syntax Guidelines support mandated.

37207 Internationalized environment variable support mandated.

37208 Presence of the utility mandated, even on systems where no communications are available.

37209 **NAME**

37210 uudecode — decode a binary file

37211 **SYNOPSIS**37212 UP uudecode [-o *outfile*][*file*]

37213

37214 **DESCRIPTION**

37215 The *uudecode* utility shall read a file, or standard input if no file is specified, that includes data
 37216 created by the *uuencode* utility. The *uudecode* utility shall scan the input file, searching for data
 37217 compatible with one of the formats specified in *uuencode* and attempt to create or overwrite the
 37218 file described by the data (or overridden by the **-o** option). The path name shall be contained in
 37219 the data or specified by the **-o** option. The file access permission bits and contents for the file to
 37220 be produced shall be contained in that data. The mode bits of the created file (other than
 37221 standard output) shall be set from the file access permission bits contained in the data; that is,
 37222 other attributes of the mode, including the file mode creation mask (see *umask*), shall not affect
 37223 the file being produced.

37224 If the path name of the file to be produced exists, and the user does not have write permission on
 37225 that file, *uudecode* shall terminate with an error. If the path name of the file to be produced exists,
 37226 and the user has write permission on that file, the existing file shall be overwritten.

37227 If the input data was produced by *uuencode* on a system with a different number of bits per byte
 37228 than on the target system, the results of *uudecode* are unspecified.

37229 **OPTIONS**

37230 The *uudecode* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,
 37231 Section 12.2, Utility Syntax Guidelines.

37232 The following option shall be supported by the implementation:

37233 **-o *outfile*** A path name of a file that shall be used instead of any path name contained in the
 37234 input data. Specifying an *outfile* option-argument of **/dev/stdout** shall indicate
 37235 standard output.

37236 **OPERANDS**

37237 The following operand shall be supported:

37238 ***file*** The path name of a file containing the output of *uuencode*.

37239 **STDIN**

37240 See the INPUT FILES section.

37241 **INPUT FILES**

37242 The input files shall be files containing the output of *uuencode*.

37243 **ENVIRONMENT VARIABLES**

37244 The following environment variables shall affect the execution of *uudecode*:

37245 ***LANG*** Provide a default value for the internationalization variables that are unset or null.
 37246 If *LANG* is unset or null, the corresponding value from the implementation-
 37247 defined default locale shall be used. If any of the internationalization variables
 37248 contains an invalid setting, the utility shall behave as if none of the variables had
 37249 been defined.

37250 ***LC_ALL*** If set to a non-empty string value, override the values of all the other
 37251 internationalization variables.

37252 ***LC_CTYPE*** Determine the locale for the interpretation of sequences of bytes of text data as
 37253 characters (for example, single-byte as opposed to multi-byte characters in

- 37254 arguments and input files).
- 37255 **LC_MESSAGES**
- 37256 Determine the locale that should be used to affect the format and contents of
- 37257 diagnostic messages written to standard error.
- 37258 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.
- 37259 **ASYNCHRONOUS EVENTS**
- 37260 Default.
- 37261 **STDOUT**
- 37262 If the file data header encoded by *uuencode* is `-` or `/dev/stdout`, or the `-o /dev/stdout` option
- 37263 overrides the file data, the standard output shall be in the same format as the file originally
- 37264 encoded by *uuencode*. Otherwise, the standard output shall not be used.
- 37265 **STDERR**
- 37266 Used only for diagnostic messages.
- 37267 **OUTPUT FILES**
- 37268 The output file shall be in the same format as the file originally encoded by *uuencode*.
- 37269 **EXTENDED DESCRIPTION**
- 37270 None.
- 37271 **EXIT STATUS**
- 37272 The following exit values shall be returned:
- 37273 0 Successful completion.
- 37274 >0 An error occurred.
- 37275 **CONSEQUENCES OF ERRORS**
- 37276 Default.
- 37277 **APPLICATION USAGE**
- 37278 The user who is invoking *uuencode* must have write permission on any file being created.
- 37279 The output of *uuencode* is essentially an encoded bit stream that is not cognizant of byte
- 37280 boundaries. It is possible that a 9-bit byte target machine can process input from an 8-bit source,
- 37281 if it is aware of the requirement, but the reverse is unlikely to be satisfying. Of course, the only
- 37282 data that is meaningful for such a transfer between architectures is generally character data.
- 37283 **EXAMPLES**
- 37284 None.
- 37285 **RATIONALE**
- 37286 Input files are not necessarily text files, as stated by an early proposal. Although the *uuencode*
- 37287 output is a text file, that output could have been wrapped within another file or mail message
- 37288 that is not a text file.
- 37289 The `-o` option is not historical practice, but was added at the request of WG15 so that the user
- 37290 could override the target path name without having to edit the input data itself.
- 37291 In early drafts, the `[-o outfile]` option-argument allowed the use of `-` to mean standard output.
- 37292 The symbol `-` has only been used previously in IEEE Std. 1003.1-200x as a standard input
- 37293 indicator. The developers of the standard did not wish to overload the meaning of `-` in this
- 37294 manner. The `/dev/stdout` concept exists on most modern systems. The `/dev/stdout` syntax does
- 37295 not refer to a new special file. It is just a magic cookie to specify standard output.

37296 **FUTURE DIRECTIONS**

37297 None.

37298 **SEE ALSO**37299 *uuencode*37300 **CHANGE HISTORY**

37301 First released in Issue 4.

37302 **Issue 6**

37303 This utility is now marked as part of the User Portability Utilities option.

37304 The **-o** *outfile* option is added, as specified in the IEEE P1003.2b draft standard.

37305 The normative text is reworded to avoid use of the term “must” for application requirements.

37306 **NAME**

37307 uuencode — encode a binary file

37308 **SYNOPSIS**37309 UP uuencode [-m][*file*] *decode_pathname*

37310

37311 **DESCRIPTION**

37312 The *uuencode* utility shall write an encoded version of the named input file, or standard input if
 37313 no *file* is specified, to standard output. The output shall be encoded using one of the algorithms
 37314 described in the STDOUT section and shall include the file access permission bits (in *chmod* octal
 37315 or symbolic notation) of the input file and the *decode_pathname*, for re-creation of the file on
 37316 another system that conforms to this volume of IEEE Std. 1003.1-200x.

37317 **OPTIONS**

37318 The *uuencode* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x,
 37319 Section 12.2, Utility Syntax Guidelines.

37320 The following option shall be supported by the implementation:

37321 **-m** Encode the output using the MIME Base64 algorithm described below. If **-m** is not
 37322 specified, the historical algorithm described in STDOUT shall be used.

37323 **OPERANDS**

37324 The following operands shall be supported:

37325 *decode_pathname*

37326 The path name of the file into which the *uudecode* utility shall place the decoded
 37327 file. Specifying a *decode_pathname* operand of **/dev/stdout** shall indicate that
 37328 *uudecode* is to use standard output. If there are characters in *decode_pathname* that
 37329 are not in the portable file name character set the results are unspecified.

37330 *file* A path name of the file to be encoded.

37331 **STDIN**

37332 See the INPUT FILES section.

37333 **INPUT FILES**

37334 Input files can be files of any type.

37335 **ENVIRONMENT VARIABLES**

37336 The following environment variables shall affect the execution of *uuencode*:

37337 **LANG** Provide a default value for the internationalization variables that are unset or null.
 37338 If **LANG** is unset or null, the corresponding value from the implementation-
 37339 defined default locale shall be used. If any of the internationalization variables
 37340 contains an invalid setting, the utility shall behave as if none of the variables had
 37341 been defined.

37342 **LC_ALL** If set to a non-empty string value, override the values of all the other
 37343 internationalization variables.

37344 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
 37345 characters (for example, single-byte as opposed to multi-byte characters in
 37346 arguments and input files).

37347 **LC_MESSAGES**

37348 Determine the locale that should be used to affect the format and contents of
 37349 diagnostic messages written to standard error.

37350 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.

37351 **ASYNCHRONOUS EVENTS**

37352 Default.

37353 **STDOUT**

37354 **uuencode Base64 Algorithm**

37355 The standard output shall be a text file (encoded in the character set of the current locale) that
37356 begins with the line:

37357 "begin-base64 Δ %s Δ %s\n", <mode>, *decode_pathname*

37358 and ends with the line:

37359 "====\n"

37360 In both cases, the lines shall have no preceding or trailing <blank>s.

37361 The encoding process represents 24-bit groups of input bits as output strings of four encoded
37362 characters. Preceding from left to right, a 24-bit input group shall be formed by concatenating
37363 three 8-bit input groups. These 24-bit then shall be treated as four concatenated 6-bit groups,
37364 each of which shall be translated into a single digit in the base64 alphabet. When encoding a bit
37365 stream via the base64 encoding, the bit stream shall be presumed to be ordered with the most-
37366 significant bit first. That is, the first bit in the stream shall be the high-order bit in the first byte,
37367 and the eighth bit shall be the low-order bit in the first byte, and so on. Each 6-bit group is used
37368 as an index into an array of 64 printable characters, as shown in Table 4-21.

Table 4-21 uuencode Base64 Values

37369

37370

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	(pad)	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

37388 The character referenced by the index shall be placed in the output string.

37389 The output stream (encoded bytes) shall be represented in lines of no more than 76 characters
37390 each. All line breaks or other characters not found in the table shall be ignored by decoding
37391 software (see *uudecode*).

37392 Special processing shall be performed if fewer than 24 bits are available at the end of a message
37393 or encapsulated part of a message. A full encoding quantum shall always be completed at the

37394 end of a message. When fewer than 24 input bits are available in an input group, zero bits shall
 37395 be added (on the right) to form an integral number of 6-bit groups. Output character positions
 37396 that are not required to represent actual input data shall be set to the character '='. Since all
 37397 base64 input is an integral number of octets, only the following cases can arise:

- 37398 1. The final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of
 37399 encoded output shall be an integral multiple of 4 characters with no '=' padding.
- 37400 2. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output
 37401 shall be two characters followed by two '=' padding characters.
- 37402 3. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded
 37403 output shall be three characters followed by one '=' padding character.
- 37404 4. The terminating "====" evaluates to nothing and denotes the end of the encoded data.

37405 uuencode Historical Algorithm

37406 The standard output shall be a text file (encoded in the character set of the current locale) that
 37407 begins with the line:

```
37408 "beginΔ%sΔ%s\n" <mode>, <decode_pathname>
```

37409 and ends with the line:

```
37410 end\n
```

37411 In both cases, the lines shall have no preceding or trailing <blank> characters.

37412 The algorithm that shall be used for lines in between **begin** and **end** takes three octets as input
 37413 and writes four characters of output by splitting the input at six-bit intervals into four octets,
 37414 containing data in the lower six bits only. These octets shall be converted to characters by adding
 37415 a value of 0x20 to each octet, so that each octet is in the range 0x20-0x5f, and then it shall be
 37416 assumed to represent a printable character in the ISO/IEC 646: 1991 standard encoded character
 37417 set. It then shall be translated into the corresponding character codes for the codeset in use in the
 37418 current locale. (For example, the octet 0x41, representing 'A', would be translated to 'A' in the
 37419 current codeset, such as 0xc1 if it were EBCDIC.)

37420 Where the bits of two octets are combined, the least significant bits of the first octet shall be
 37421 shifted left and combined with the most significant bits of the second octet shifted right. Thus
 37422 the three octets *A*, *B*, *C* shall be converted into the four octets:

```
37423 0x20 + (( A >> 2 ) & 0x3F)
37424 0x20 + ((( A << 4 ) | (( B >> 4 ) & 0xF ) ) & 0x3F)
37425 0x20 + ((( B << 2 ) | (( C >> 6 ) & 0x3 ) ) & 0x3F)
37426 0x20 + (( C ) & 0x3F)
```

37427 These octets then shall be translated into the local character set.

37428 Each encoded line contains a length character, equal to the number of characters to be decoded
 37429 plus 0x20 translated to the local character set as described above, followed by the encoded
 37430 characters. The maximum number of octets to be encoded on each line shall be 45.

37431 STDERR

37432 Used only for diagnostic messages.

37433 OUTPUT FILES

37434 None.

37435 **EXTENDED DESCRIPTION**

37436 None.

37437 **EXIT STATUS**

37438 The following exit values shall be returned:

37439 0 Successful completion.

37440 >0 An error occurred.

37441 **CONSEQUENCES OF ERRORS**

37442 Default.

37443 **APPLICATION USAGE**

37444 The file is expanded by 35 percent (each three octets become four, plus control information)
37445 causing it to take longer to transmit.

37446 Since this utility is intended to create files to be used for data interchange between systems with
37447 possibly different codesets, and to represent binary data as a text file, the ISO/IEC 646:1991
37448 standard was chosen for a midpoint in the algorithm as a known reference point. The output
37449 from *uuencode* is a text file on the local system. If the output were in the ISO/IEC 646:1991
37450 standard codeset, it might not be a text file (at least because the <newline> characters might not
37451 match), and the goal of creating a text file would be defeated. If this text file was then carried to
37452 another machine with the same codeset, it would be perfectly compatible with that system's
37453 *uudecode*. If it was transmitted over a mail system or sent to a machine with a different codeset,
37454 it is assumed that, as for every other text file, some translation mechanism would convert it (by
37455 the time it reached a user on the other system) into an appropriate codeset. This translation only
37456 makes sense from the local codeset, not if the file has been put into a ISO/IEC 646:1991 standard
37457 representation first. Similarly, files processed by *uuencode* can be placed in *pax* archives,
37458 intermixed with other text files in the same codeset.

37459 The algorithm is described in terms of 8-bit quantities, or octets. Since no byte alignment is
37460 implied, it encodes data from machines with any number of bits per byte. However, unless that
37461 encoded data is then decoded on a machine with the same number of bits per byte, the output
37462 might not be useful.

37463 **EXAMPLES**

37464 None.

37465 **RATIONALE**

37466 A new algorithm was added at the request of the international community to parallel work in
37467 RFC 2045 (MIME). As with the historical *uuencode* format, the Base64 Content-Transfer-Encoding
37468 is designed to represent arbitrary sequences of octets in a form that is not humanly readable. A
37469 65-character subset of the ISO/IEC 646:1991 standard is used, enabling 6 bits to be represented
37470 per printable character. (The extra 65th character, '=', is used to signify a special processing
37471 function.)

37472 This subset has the important property that it is represented identically in all versions of the
37473 ISO/IEC 646:1991 standard, including US ASCII, and all characters in the subset are also
37474 represented identically in all versions of EBCDIC. The historical *uuencode* algorithm does not
37475 share this property, which is the reason that a second algorithm was added to the ISO POSIX-2
37476 standard.

37477 The string "====" was used for the termination instead of the end used in the original format
37478 because the latter is a string that could be valid encoded input.

37479 In an early draft, the `-m` option was named `-b` (for Base64), but it was renamed to reflect its
37480 relationship to the RFC 2045. A `-u` was also present to invoke the default algorithm, but since

- 37481 this was not historical practice, it was omitted as being unnecessary.
- 37482 See the RATIONALE section in *uudecode* for the derivation of the `/dev/stdout` symbol.
- 37483 **FUTURE DIRECTIONS**
- 37484 None.
- 37485 **SEE ALSO**
- 37486 *mailx, uudecode*
- 37487 **CHANGE HISTORY**
- 37488 First released in Issue 4.
- 37489 **Issue 6**
- 37490 This utility is now marked as part of the User Portability Utilities option.
- 37491 The Base64 algorithm and the ability to output to `/dev/stdout` are added as specified in the
- 37492 IEEE P1003.2b draft standard.

37493 NAME

37494 uustat — uucp status inquiry and job control

37495 SYNOPSIS

37496 UN XSI uustat [-q | -k *jobid* | -r *jobid*]37497 XSI uustat [-s *system*][-u *user*]

37498 DESCRIPTION

37499 The *uustat* utility shall display the status of, or cancel, previously specified *uucp* requests, or
37500 provide general status on *uucp* connections to other systems.37501 When no options are given, *uustat* shall write to standard output the status of all *uucp* requests
37502 issued by the current user.37503 Typical implementations of this utility require a communications line configured to use the Base
37504 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface, but other
37505 communications means may be used. On systems where there are no available communications
37506 means (either temporarily or permanently), this utility shall write an error message describing
37507 the problem and exits with a non-zero exit status.

37508 OPTIONS

37509 The *uustat* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
37510 12.2, Utility Syntax Guidelines.

37511 The following options shall be supported:

37512 UN -q Write the jobs queued for each machine.

37513 -k *jobid* Kill the *uucp* request whose job identification is *jobid*. The application shall ensure
37514 that the killed *uucp* request belongs to the person invoking *uustat* unless that user
37515 has appropriate privileges.37516 -r *jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their
37517 modification time is set to the current time. This prevents the cleanup program
37518 from deleting the job until the jobs modification time reaches the limit imposed by
37519 the program.37520 -s *system* Write the status of all *uucp* requests for remote system *system*.37521 -u *user* Write the status of all *uucp* requests issued by *user*.

37522 OPERANDS

37523 None.

37524 STDIN

37525 Not used.

37526 INPUT FILES

37527 None.

37528 ENVIRONMENT VARIABLES

37529 The following environment variables shall affect the execution of *uustat*:37530 *LANG* Provide a default value for the internationalization variables that are unset or null.
37531 If *LANG* is unset or null, the corresponding value from the implementation-
37532 defined default locale shall be used. If any of the internationalization variables
37533 contains an invalid setting, the utility shall behave as if none of the variables had
37534 been defined.

- 37535 *LC_ALL* If set to a non-empty string value, override the values of all the other
37536 internationalization variables.
- 37537 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
37538 characters (for example, single-byte as opposed to multi-byte characters in
37539 arguments).
- 37540 *LC_MESSAGES*
37541 Determine the locale that should be used to affect the format and contents of
37542 diagnostic messages written to standard error, and informative messages written
37543 to standard output.
- 37544 *LC_TIME* Determine the format of date and time strings output by *uustat*.
- 37545 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.
- 37546 *TZ* Determine the timezone used with date and time strings.
- 37547 **ASYNCHRONOUS EVENTS**
37548 Default.
- 37549 **STDOUT**
37550 The standard output shall consist of information about each job selected, in an unspecified
37551 format. The information shall include at least the job ID, the user ID or name, and the remote
37552 system name.
- 37553 **STDERR**
37554 Used only for diagnostic messages.
- 37555 **OUTPUT FILES**
37556 None.
- 37557 **EXTENDED DESCRIPTION**
37558 None.
- 37559 **EXIT STATUS**
37560 The following exit values shall be returned:
37561 0 Successful completion.
37562 >0 An error occurred.
- 37563 **CONSEQUENCES OF ERRORS**
37564 Default.
- 37565 **APPLICATION USAGE**
37566 None.
- 37567 **EXAMPLES**
37568 None.
- 37569 **RATIONALE**
37570 None.
- 37571 **FUTURE DIRECTIONS**
37572 None.
- 37573 **SEE ALSO**
37574 *uucp*

37575 **CHANGE HISTORY**

37576 First released in Issue 2.

37577 **Issue 4**

37578 Format reorganized.

37579 Utility Syntax Guidelines support mandated.

37580 Internationalized environment variable support mandated.

37581 Presence of the utility mandated, even on systems where no communications are available.

37582 **Issue 6**

37583 The normative text is reworded to avoid use of the term “must” for application requirements.

37584 **NAME**

37585 uux — remote command execution

37586 **SYNOPSIS**37587 XSI uux [-np] *command-string*37588 UN XSI uux [-jnp] *command-string*37589 **DESCRIPTION**

37590 The *uux* utility shall gather zero or more files from various systems, execute a shell pipeline (see
 37591 Section 2.9 (on page 2256)) on a specified system, and then send the standard output of the
 37592 command to a file on a specified system. Only the first command of a pipeline can have a
 37593 *system-name!* prefix. All other commands in the pipeline shall be executed on the system of the
 37594 first command.

37595 The following restrictions are applicable to the shell pipeline processed by *uux*:

- 37596 • In gathering files from different systems, path name expansion is not performed by *uux*.
 37597 Thus, a request such as:

```
37598 uux "c99 remsys!~/*.c"
```

37599 would attempt to copy the file named literally *.c to the local system.

- 37600 • The redirection operators ">>", "<<", ">|", and ">&" shall not be accepted. Any use of
 37601 these redirection operators shall cause this utility to write an error message describing the
 37602 problem and exit with a non-zero exit status.
- 37603 • The reserved word ! cannot be used at the head of the pipeline to modify the exit status.
- 37604 • Alias substitution is not performed.

37605 A file name can be specified as for *uucp*; it can be an absolute path name, a path name preceded
 37606 by *~name* (which is replaced by the corresponding login directory), a path name specified as
 37607 *~/dest* (*dest* is prefixed by the public directory called *PUBDIR*; the actual location of *PUBDIR* is
 37608 implementation-defined), or a simple file name (which is prefixed by *uux* with the current
 37609 directory). See *uucp* (on page 3178) for the details.

37610 The execution of commands on remote systems shall take place in an execution directory known
 37611 to the *uucp* system. All files required for the execution shall be put into this directory unless they
 37612 already reside on that machine. Therefore, the application shall ensure that non-local file names
 37613 (without path or machine reference) are unique within the *uux* request.

37614 The *uux* utility shall attempt to get all files to the execution system. For files that are output files,
 37615 the application shall ensure that the file name is escaped using parentheses.

37616 The remote system shall notify the user by mail if the requested command on the remote system
 37617 was disallowed or the files were not accessible. This notification can be turned off by the *-n*
 37618 option.

37619 Typical implementations of this utility require a communications line configured to use the Base
 37620 Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General Terminal Interface, but other
 37621 communications means may be used. On systems where there are no available communications
 37622 means (either temporarily or permanently), this utility shall write an error message describing
 37623 the problem and exits with a non-zero exit status.

37624 The *uux* utility cannot guarantee support for all character encodings in all circumstances. For
 37625 example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and
 37626 file names need not be portable to non-internationalized systems, and so on. Under these
 37627 circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991

37628 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used
 37629 and that only characters defined in the Portable File Name Character Set be used for naming
 37630 files.

37631 OPTIONS

37632 The *uux* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 37633 12.2, Utility Syntax Guidelines.

37634 The following options shall be supported:

37635 **-p** Make the standard input to *uux* the standard input to the *command-string*.

37636 UN **-j** Write the job identification string to standard output. This job identification can be
 37637 used by *uustat* to obtain the status or terminate a job.

37638 **-n** Do not notify the user if the command fails.

37639 OPERANDS

37640 The following operand shall be supported:

37641 *command-string*

37642 A string made up of one or more arguments that are similar to normal command
 37643 arguments, except that the command and any file names can be prefixed by
 37644 *system-name!*. A null *system-name* shall be interpreted as the local system.

37645 STDIN

37646 The standard input shall not be used unless the *'-'* or **-p** option is specified; in those cases, the
 37647 standard input shall be made the standard input of the *command-string*.

37648 INPUT FILES

37649 Input files shall be selected according to the contents of *command-string*.

37650 ENVIRONMENT VARIABLES

37651 The following environment variables shall affect the execution of *uux*:

37652 *LANG* Provide a default value for the internationalization variables that are unset or null.
 37653 If *LANG* is unset or null, the corresponding value from the implementation-
 37654 defined default locale shall be used. If any of the internationalization variables
 37655 contains an invalid setting, the utility shall behave as if none of the variables had
 37656 been defined.

37657 *LC_ALL* If set to a non-empty string value, override the values of all the other
 37658 internationalization variables.

37659 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 37660 characters (for example, single-byte as opposed to multi-byte characters in
 37661 arguments).

37662 *LC_MESSAGES*

37663 Determine the locale that should be used to affect the format and contents of
 37664 diagnostic messages written to standard error.

37665 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

37666 ASYNCHRONOUS EVENTS

37667 Default.

37668 **STDOUT**

37669 The standard output shall be not used unless the **-j** option is specified; in that case, the job
37670 identification string shall be written to standard output in the following format:

37671 "%s\n", <jobid>

37672 **STDERR**

37673 Used only for diagnostic messages.

37674 **OUTPUT FILES**

37675 Output files shall be created or written, or both, according to the contents of *command-string*.

37676 If the **-n** is not used, mail files shall be modified following any command or file-access failures
37677 on the remote system.

37678 **EXTENDED DESCRIPTION**

37679 None.

37680 **EXIT STATUS**

37681 The following exit values shall be returned:

37682 0 Successful completion.

37683 >0 An error occurred.

37684 **CONSEQUENCES OF ERRORS**

37685 Default.

37686 **APPLICATION USAGE**

37687 Note that, for security reasons, many installations limit the list of commands executable on
37688 behalf of an incoming request from *uux*. Many sites permit little more than the receipt of mail
37689 via *uux*.

37690 Any characters special to the command interpreter should be quoted either by quoting the entire
37691 *command-string* or quoting the special characters as individual arguments.

37692 As noted in *uucp*, shell pattern matching notation characters appearing in path names are
37693 expanded on the appropriate local system. This is done under the control of local settings of
37694 *LC_COLLATE* and *LC_CTYPE*. Thus, care should be taken when using bracketed file name
37695 patterns, as collation and typing rules may vary from one system to another. Also be aware that
37696 certain types of expression (that is, equivalence classes, character classes, and collating symbols)
37697 need not be supported on non-internationalized systems.

37698 **EXAMPLES**

37699 1. The following command gets **file1** from system **a** and **file2** file from system **b**, executes *diff*
37700 on the local system, and puts the results in **file.diff** in the local *PUBDIR* directory.
37701 (*PUBDIR* is the *uucp* public directory on the local system.)

37702 uux "!diff a!/usr/file1 b!/a4/file2 >!~/file.diff"

37703 2. The following command fails because *uux* places all files copied to a system in the same
37704 working directory. Although the files **xyz** are from two different systems, their file names
37705 are the same and conflict.

37706 uux "!diff a!/usr1/xyz b!/usr2/xyz >!~/xyz.diff"

37707 3. The following command succeeds (assuming *diff* is permitted on system **a**) because the file
37708 local to system **a** is not copied to the working directory, and hence does not conflict the file
37709 from system **c**.

37710 uux "a!diff a!/usr/xyz c!/usr/xyz >!~/xyz.diff"

37711 **RATIONALE**

37712 None.

37713 **FUTURE DIRECTIONS**

37714 A version of *uux* that fully supports the Base Definitions volume of IEEE Std. 1003.1-200x,
37715 Section 12.2, Utility Syntax Guidelines may be introduced in a future issue.

37716 **SEE ALSO**

37717 *uucp, uuencode, uustat*

37718 **CHANGE HISTORY**

37719 First released in Issue 2.

37720 **Issue 4**

37721 Format reorganized.

37722 Exceptions to Utility Syntax Guidelines conformance noted.

37723 Internationalized environment variable support mandated.

37724 Presence of the utility mandated, even on systems where no communications are available.

37725 **Issue 6**

37726 The obsolescent SYNOPSIS is removed.

37727 The normative text is reworded to avoid use of the term “must” for application requirements.

37728 **NAME**37729 val — validate SCCS files (**DEVELOPMENT**)37730 **SYNOPSIS**

37731 xSI val -

37732 val [-s][-m name][-r SID][-y type] file...

37733

37734 **DESCRIPTION**37735 The *val* utility shall determine whether the specified *file* is an SCCS file meeting the
37736 characteristics specified by the options.37737 **OPTIONS**37738 The *val* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
37739 12.2, Utility Syntax Guidelines, except that the usage of the '-' operand is not strictly as
37740 intended by the guidelines (that is, reading options and operands from standard input).

37741 The following options shall be supported:

37742 **-m name** Specify a *name*, which is compared with the SCCS %M% keyword in *file*; see *get*
37743 (on page 2685).37744 **-r SID** Specify a *SID* (SCCS Identification String), an SCCS delta number. A check shall be
37745 made to determine whether the *SID* is ambiguous (for example, **-r 1** is ambiguous
37746 because it physically does not exist but implies 1.1, 1.2, and so on, which may
37747 exist) or invalid (for example, **-r 1.0** or **-r 1.1.0** are invalid because neither case can
37748 exist as a valid delta number). If the *SID* is valid and not ambiguous, a check shall
37749 be made to determine whether it actually exists.37750 **-s** Silence the diagnostic message normally written to standard output for any error
37751 that is detected while processing each named file on a given command line.37752 **-y type** Specify a *type*, which shall be compared with the SCCS %Y% keyword in *file*; see
37753 *get* (on page 2685).37754 **OPERANDS**

37755 The following operands shall be supported:

37756 *file* A path name of an existing SCCS file. If exactly one *file* operand appears, and it is
37757 '-', the standard input shall be read: each line is independently processed as if it
37758 were a command line argument list. (However, the line is not subjected to any of
37759 the shell word expansions, such as parameter expansion or quote removal.)37760 **STDIN**37761 The standard input shall be a text file used only when the *file* operand is specified as '-'.37762 **INPUT FILES**

37763 Any SCCS files processed are files of an unspecified format.

37764 **ENVIRONMENT VARIABLES**37765 The following environment variables shall affect the execution of *val*:37766 **LANG** Provide a default value for the internationalization variables that are unset or null.
37767 If *LANG* is unset or null, the corresponding value from the implementation-
37768 defined default locale shall be used. If any of the internationalization variables
37769 contains an invalid setting, the utility shall behave as if none of the variables had
37770 been defined.

37771 *LC_ALL* If set to a non-empty string value, override the values of all the other
37772 internationalization variables.

37773 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
37774 characters (for example, single-byte as opposed to multi-byte characters in
37775 arguments and input files).

37776 *LC_MESSAGES*
37777 Determine the locale that should be used to affect the format and contents of
37778 diagnostic messages written to standard error, and informative messages written
37779 to standard output.

37780 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

37781 **ASYNCHRONOUS EVENTS**
37782 Default.

37783 **STDOUT**
37784 The standard output shall consist of informative messages about either:
37785 1. Each file processed
37786 2. Each command line read from standard input

37787 If the standard input is not used, for each *file* operand yielding a discrepancy, the output line
37788 shall have the following format:
37789 "%s: %s\n", <pathname>, <unspecified string>

37790 If standard input is used, a line of input shall be written before each of the preceding lines for
37791 files containing discrepancies:
37792 "%s:\n", <input line>

37793 **STDERR**
37794 Not used.

37795 **OUTPUT FILES**
37796 None.

37797 **EXTENDED DESCRIPTION**
37798 None.

37799 **EXIT STATUS**
37800 The 8-bit code returned by *val* is a disjunction of the possible errors, that is, it can be interpreted
37801 as a bit string where set bits are interpreted as follows:

37802 0x80 = Missing file argument.
37803 0x40 = Unknown or duplicate option.
37804 0x20 = Corrupted SCCS file.
37805 0x10 = Cannot open file or file not SCCS.
37806 0x08 = *SID* is invalid or ambiguous.
37807 0x04 = *SID* does not exist.
37808 0x02 = %Y%, -y mismatch.
37809 0x01 = %M%, -m mismatch.

37810 Note that *val* can process two or more files on a given command line and can process multiple
37811 command lines (when reading the standard input). In these cases an aggregate code shall be
37812 returned: a logical OR of the codes generated for each command line and file processed.

37813 **CONSEQUENCES OF ERRORS**

37814 Default.

37815 **APPLICATION USAGE**

37816 Since the *val* exit status sets the 0x80 bit, shell applications checking "\$?" cannot tell if it
37817 terminated due to a missing file argument or receipt of a signal.

37818 **EXAMPLES**

37819 In a directory with three SCCS files, *s.x* (of *t* type "text"), *s.y*, and *s.z* (a corrupted file), the
37820 following command could produce the output shown:

37821 `val - <<EOF`37822 `-y source s.x`37823 `-m y s.y`37824 `s.z`37825 `EOF`37826 `-y source s.x`37827 `s.x: %Y%, -y mismatch`37828 `s.z`37829 `s.z: corrupted SCCS file`37830 **RATIONALE**

37831 None.

37832 **FUTURE DIRECTIONS**

37833 None.

37834 **SEE ALSO**37835 *admin, delta, get, prs*37836 **CHANGE HISTORY**

37837 First released in Issue 2.

37838 **Issue 4**

37839 Format reorganized.

37840 Exceptions to Utility Syntax Guidelines conformance noted.

37841 Internationalized environment variable support mandated.

37842 **Issue 6**37843 The Open Group corrigenda item U025/4 has been applied, correcting a typographical error in
37844 the EXIT STATUS.

37845 The normative text is reworded to emphasise the term "shall" for implementation requirements.

37846 **NAME**

37847 vi — screen-oriented (visual) display editor

37848 **SYNOPSIS**37849 UP `vi [-rR][-l][-c command][-t tagstring][-w size][file ...]`

37850

37851 **DESCRIPTION**37852 This utility shall be provided on systems that both support the User Portability Utilities option
37853 and define the POSIX2_CHAR_TERM symbol. On other systems it is optional.37854 The *vi* (visual) utility is a screen-oriented text editor. Only the open and visual modes of the
37855 editor are described in IEEE Std. 1003.1-200x; see the line editor *ex* for additional editing
37856 capabilities used in *vi*. The user can switch back and forth between *vi* and *ex* and execute *ex*
37857 commands from within *vi*.37858 This reference page uses the term *edit buffer* to describe the current working text. No specific
37859 implementation is implied by this term. All editing changes are performed on the edit buffer,
37860 and no changes to it shall affect any file until an editor command writes the file.37861 When using *vi*, the terminal screen acts as a window into the editing buffer. Changes made to
37862 the editing buffer shall be reflected in the screen display; the position of the cursor on the screen
37863 shall indicate the position within the editing buffer.37864 Certain terminals do not have all the capabilities necessary to support the complete *vi* definition.
37865 When these commands cannot be supported on such terminals, this condition shall not produce
37866 an error message such as “not an editor command” or report a syntax error. The implementation
37867 may either accept the commands and produce results on the screen that are the result of an
37868 unsuccessful attempt to meet the requirements of this volume of IEEE Std. 1003.1-200x or report
37869 an error describing the terminal-related deficiency.37870 **OPTIONS**37871 The *vi* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section 12.2,
37872 Utility Syntax Guidelines.

37873 The following options shall be supported:

37874 `-c command` See the *ex* command description of the `-c` option.37875 `-l` (The letter ell.) Set lisp mode; see **Edit Options in *ex*** (on page 2602).37876 `-r` See the *ex* command description of the `-r` option.37877 `-R` See the *ex* command description of the `-R` option.37878 `-t tagstring` See the *ex* command description of the `-t` option.37879 `-w size` See the *ex* command description of the `-w` option.37880 **OPERANDS**37881 See the OPERANDS section of the *ex* command for a description of the operands supported by
37882 the *vi* command.37883 **STDIN**37884 If standard input is not a terminal device, the results are undefined. The standard input consists
37885 of a series of commands and input text, as described in the EXTENDED DESCRIPTION section.37886 If a read from the standard input returns an error, or if the editor detects an end-of-file condition
37887 from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

37888 **INPUT FILES**

37889 See the INPUT FILES section of the *ex* command for a description of the input files supported by
37890 the *vi* command.

37891 **ENVIRONMENT VARIABLES**

37892 See the ENVIRONMENT VARIABLES section of the *ex* command for the environment variables
37893 that affect the execution of the *vi* command.

37894 **ASYNCHRONOUS EVENTS**

37895 See the ASYNCHRONOUS EVENTS section of the *ex* for the asynchronous events that affect the
37896 execution of the *vi* command.

37897 **STDOUT**

37898 If standard output is not a terminal device, undefined results occur.

37899 Standard output may be used for writing prompts to the user, for informational messages, and
37900 for writing lines from the file.

37901 **STDERR**

37902 If standard output is not a terminal device, undefined results occur.

37903 Used only for diagnostic messages.

37904 **OUTPUT FILES**

37905 See the OUTPUT FILES section of the *ex* command for a description of the output files
37906 supported by the *vi* command.

37907 **EXTENDED DESCRIPTION**

37908 If the terminal does not have the capabilities necessary to support an unspecified portion of the
37909 *vi* definition, implementations shall start initially in *ex* mode or open mode. Otherwise, after
37910 initialization, *vi* shall be in command mode; text input mode can be entered by one of several
37911 commands used to insert or change text. In text input mode, <ESC> can be used to return to
37912 command mode; other uses of <ESC> are described later in this section; see **Terminate**
37913 **Command or Input Mode** (on page 3209).

37914 **Initialization in *ex* and *vi***

37915 See **Initialization in *ex* and *vi*** (on page 2569) for a description of *ex* and *vi* initialization for the *vi*
37916 utility.

37917 **Command Descriptions in *vi***

37918 The following symbols are used in this reference page to represent arguments to commands.

37919 *buffer* See the description of *buffer* in the EXTENDED DESCRIPTION section of the *ex* utility;
37920 see **Command Descriptions in *ex*** (on page 2578).

37921 In open and visual mode, when a command synopsis shows both [*buffer*] and [*count*]
37922 preceding the command name, they can be specified in either order.

37923 *count* A positive integer used as an optional argument to most commands, either to give a
37924 repeat count or as a size. This argument is optional and shall default to 1 unless
37925 otherwise specified.

37926 The Synopsis lines for the *vi* commands <control>-G, <control>-L, <control>-R,
37927 <control>-], %, &, ^, D, m, M, Q, u, U, and ZZ do not have *count* as an optional
37928 argument. Regardless, it shall not be an error to specify a *count* to these commands, and
37929 any specified *count* shall be ignored.

37930 *motion* An optional trailing argument used by the **!**, **<**, **>**, **c**, **d**, and **y** commands, which is used
 37931 to indicate the region of text that shall be affected by the command. The motion can be
 37932 either one of the command characters repeated or one of several other *vi* commands
 37933 (listed in the following table). Each of the applicable commands specifies the region of
 37934 text matched by repeating the command; each command that can be used as a motion
 37935 command specifies the region of text it affects.

37936 Commands that take *motion* arguments operate on either lines or characters, depending
 37937 on the circumstances. When operating on lines, all lines that fall partially or wholly
 37938 within the text region specified for the command shall be affected. When operating on
 37939 characters, only the exact characters in the specified text region shall be affected. Each
 37940 motion command specifies this individually.

37941 When commands that may be motion commands are not used as motion commands,
 37942 they shall set the current position to the current line and column as specified.

37943 The following commands shall be valid cursor motion commands:

37944 <control>-H	;	'character
37945 <newline>	?	b
37946 <carriage-return>	B	e
37947 <control>-N	E	f
37948 <control>-P	F	h
37949 <space>	G	j
37950 \$	H	k
37951 %	L	l
37952 'character	M	n
37953 (N	t
37954)	T	w
37955 +	W	{
37956 ,	[[
37957 -]]	}
37958 /	^	0
37959 _		

37960 Any *count* that is specified to a command that has an associated motion command shall
 37961 be applied to the motion command. If a *count* is applied to both the command and its
 37962 associated motion command, the effect shall be multiplicative.

37963 The following symbol is used in this section to specify locations in the edit buffer:

37964 *current character*

37965 The character that is currently displayed by the cursor.

37966 The following symbols are used in this section to specify command actions:

37967 *bigword* In the POSIX locale, *vi* shall recognize four kinds of *bigwords*:

- 37968 1. A maximal sequence of non-<blank> characters preceded and followed by
 37969 <blank> characters or the beginning or end of a line or the edit buffer
- 37970 2. One or more sequential empty or <blank> character-filled lines
- 37971 3. The first character in the edit buffer
- 37972 4. The last character in the edit buffer

37973 *word* In the POSIX locale, *vi* shall recognize five kinds of words:

- 37974 1. A maximal sequence of letters, digits, and underscores, delimited at both ends by:
- 37975 — Characters other than letters, digits, or underscores
- 37976 — The beginning or end of a line
- 37977 — The beginning or end of the edit buffer
- 37978 2. A maximal sequence of characters other than letters, digits, underscores, or
- 37979 <blank> characters, delimited at both ends by:
- 37980 — A letter, digit, underscore
- 37981 — <blank> characters
- 37982 — The beginning or end of a line
- 37983 — The beginning or end of the edit buffer
- 37984 3. One or more sequential empty or <blank> character-filled lines
- 37985 4. The first character in the edit buffer
- 37986 5. The last character in the edit buffer

37987 *section boundary*

37988 A *section boundary* is one of the following:

- 37989 1. A line whose first character is a <form-feed> character
- 37990 2. A line whose first character is an open curly brace (' { ')
- 37991 3. A line whose first character is a period and whose second and third characters
- 37992 match a two-character pair in the **sections** edit option (see *ed*)
- 37993 4. A line whose first character is a period and whose only other character matches
- 37994 the first character of a two-character pair in the **sections** edit option, where the
- 37995 second character of the two-character pair is a <space> character
- 37996 5. The first line of the edit buffer
- 37997 6. The last line of the edit buffer if the last line of the edit buffer is empty or if it is a
- 37998 **]]** or **}** command; otherwise, the last character of the last line of the edit buffer

37999 *paragraph boundary*

38000 A *paragraph boundary* is one of the following:

- 38001 1. A section boundary
- 38002 2. A line whose first character is a period and whose second and third characters
- 38003 match a two-character pair in the **paragraphs** edit option (see *ed*)
- 38004 3. A line whose first character is a period and whose only other character matches
- 38005 the first character of a two-character pair in the *paragraphs* edit option, where the
- 38006 second character of the two-character pair is a <space> character
- 38007 4. One or more sequential empty or <blank> character-filled lines

38008 *remembered search direction*

38009 See the description of remembered search direction in *ed*.

38010 *sentence boundary*

38011 A *sentence boundary* is one of the following:

- 38012 1. A paragraph boundary
- 38013 2. The first non-<blank> character that occurs after a paragraph boundary
- 38014 3. The first non-<blank> character that occurs after a period ('.'), exclamation
38015 mark (' ! '), or question mark (' ? '), followed by two <space> characters or the
38016 end of a line; any number of closing parenthesis (') '), closing brackets ('] '),
38017 double quote (' " '), or single quote (' \ ' ') characters can appear between the
38018 punctuation mark and the two <space> characters or end-of-line

38019 Any lines displayed on the screen that logically represent lines after the last line in the edit buffer
38020 shall consist of a single tilde ('~ ') character.

38021 The last line of the screen shall be used to report errors or display informational messages. It
38022 shall also be used to display the input for “line-oriented commands” (/ , ? , : , and !). When a line-
38023 oriented command is executed, the editor shall enter text input mode on the last line on the
38024 screen, using the respective command characters as prompt characters. (In the case of the !
38025 command, the associated motion shall be entered by the user before the editor enters text input
38026 mode.) The line entered by the user shall be terminated by a character, a non-<control>-V-
38027 escaped <carriage-return> character, or unescaped <ESC>. It is unspecified if more characters
38028 than require a display width minus one column number of screen columns can be entered.

38029 If any command is executed that overwrites a portion of the screen other than the last line of the
38030 screen (for example, the *ex suspend*, or ! commands), other than the *ex shell* command, the user
38031 shall be prompted for a character before the screen is refreshed and the edit session continued.

38032 <tab> characters shall take up the number of columns on the screen set by the **tabstop** edit
38033 option (see *ed*), unless there are less than that number of columns before the display margin that
38034 will cause the displayed line to be folded; in this case, they shall only take up the number of
38035 columns up to that boundary.

38036 The cursor shall be placed on the current line and relative to the current column as specified by
38037 each command described in the following sections.

38038 In open mode, if the current line is not already displayed, then it shall be displayed.

38039 In the remainder of the description of the *vi* utility, the term “physical line” refers to a line in the
38040 edit buffer and the term “logical line” refers to the line or lines on the display screen used to
38041 display a physical line.

38042 In visual mode, if the current line is not displayed, then the lines that are displayed shall be
38043 expanded, scrolled, or redrawn to cause an unspecified portion of the current line to be
38044 displayed. If the screen is redrawn, no more than the number of logical lines specified by the
38045 value of the **window** edit option shall be displayed (unless the current line cannot be completely
38046 displayed in the number of logical lines specified by the **window** edit option) and the current
38047 line shall be positioned as close to the center of the displayed lines as possible (within the
38048 constraints imposed by the distance of the line from the beginning or end of the edit buffer). If
38049 the current line is before the first line in the display and the screen is scrolled, an unspecified
38050 portion of the current line shall be placed on the first line of the display. If the current line is after
38051 the last line in the display and the screen is scrolled, an unspecified portion of the current line
38052 shall be placed on the last line of the display.

38053 In visual mode, if a line from the edit buffer (other than the current line) does not entirely fit into
38054 the lines at the bottom of the display that are available for its presentation, the editor may
38055 choose not to display any portion of the line. The lines of the display that do not contain text
38056 from the edit buffer for this reason shall each consist of a single '@' character.

38057 In visual mode, the editor may choose for unspecified reasons to not update lines in the display
 38058 to correspond to the underlying edit buffer text. The lines of the display that do not correctly
 38059 correspond to text from the edit buffer for this reason shall consist of a single '@' character, and
 38060 the <control>-R command shall cause the editor to update the screen to correctly represent the
 38061 edit buffer.

38062 Open and visual mode commands that set the current column set it to a column position in the
 38063 display, and not a character position in the line. In this case, however, the column position in the
 38064 display shall be calculated for a infinite width display; for example, the column related to a
 38065 character that is part of a line that has been folded onto additional screen lines will be offset from
 38066 the screen column where the physical line begins, not from the beginning of a particular screen
 38067 line.

38068 The physical cursor column in the display is based on the value of the current column, as
 38069 follows, with each rule applied in turn:

- 38070 1. If the current column is after the last screen column used by the displayed line, the
 38071 physical cursor column shall be set to the last screen column occupied by the last character
 38072 in the current line; otherwise, the physical cursor column shall be set to the current
 38073 column.
- 38074 2. If the character of which some portion is displayed in the screen column specified by the
 38075 physical cursor column requires more than a single screen column:
 - 38076 a. If in text input mode, the physical cursor column shall be adjusted to the first screen
 38077 column in which any portion of that character is displayed.
 - 38078 b. Otherwise, the physical cursor column shall be adjusted to the last screen column in
 38079 which any portion of that character is displayed.

38080 The current column shall not be changed by these adjustments to the physical cursor column.

38081 If an error occurs during the parsing or execution of a *vi* command:

- 38082 • The terminal shall be alerted. Execution of the *vi* command shall stop, and the cursor (for
 38083 example, the current line and column) shall not be further modified.
- 38084 • Unless otherwise specified by the following command sections, it is unspecified whether an
 38085 informational message shall be displayed.
- 38086 • Any partially entered *vi* command shall be discarded.
- 38087 • If the *vi* command resulted from a **map** expansion, all characters from that **map** expansion
 38088 shall be discarded, except as otherwise specified by the **map** command (see *ed*).
- 38089 • If the *vi* command resulted from the execution of a buffer, no further commands caused by
 38090 the execution of the buffer shall be executed.

38091 **Page Backwards**

38092 *Synopsis:* [count] <control>-B

38093 If in open mode, the <control>-B command shall behave identically to the **z** command.
 38094 Otherwise, if the current line is the first line of the edit buffer, it shall be an error.

38095 If the **window** edit option is less than 3, display a screen where the last line of the display shall
 38096 be some portion of:

38097 (*current first line*) -1

38098 otherwise, display a screen where the first line of the display shall be some portion of:
 38099 $(\text{current first line}) - \text{count} \times ((\text{window edit option}) - 2)$
 38100 If this calculation would result in a line that is before the first line of the edit buffer, the first line
 38101 of the display shall display some portion of the first line of the edit buffer.
 38102 *Current line*: If no lines from the previous display remain on the screen, set to the last line of the
 38103 display; otherwise, set to $(\text{line} - \text{the number of new lines displayed on this screen})$.
 38104 *Current column*: Set to non-`<blank>`.

38105 **Scroll Forward**
 38106 *Synopsis*: `[count] <control>-D`
 38107 If the current line is the last line of the edit buffer, it shall be an error.
 38108 If no *count* is specified, *count* shall default to the *count* associated with the previous `<control>-D`
 38109 or `<control>-U` command. If there was no previous `<control>-D` or `<control>-U` command, *count*
 38110 shall default to the value of the **scroll** edit option.
 38111 If in open mode, write lines starting with the line after the current line, until *count* lines or the
 38112 last line of the file have been written.
 38113 *Current line*: If the current line + *count* is past the last line of the edit buffer, set to the last line of
 38114 the edit buffer; otherwise, set to the current line + *count*.
 38115 *Current column*: Set to non-`<blank>`.

38116 **Scroll Forward by Line**
 38117 *Synopsis*: `[count] <control>-E`
 38118 Display the line *count* lines after the last line currently displayed.
 38119 If the last line of the edit buffer is displayed, it shall be an error. If there is no line *count* lines
 38120 after the last line currently displayed, the last line of the display shall display some portion of
 38121 the last line of the edit buffer.
 38122 *Current line*: Unchanged if the previous current character is displayed; otherwise, set to the first
 38123 line displayed.
 38124 *Current column*: Unchanged.

38125 **Page Forward**
 38126 *Synopsis*: `[count] <control>-F`
 38127 If in open mode, the `<control>-F` command shall behave identically to the **z** command.
 38128 Otherwise, if the current line is the last line of the edit buffer, it shall be an error.
 38129 If the **window** edit option is less than 3, display a screen where the first line of the display shall
 38130 be some portion of:
 38131 $(\text{current last line}) + 1$
 38132 otherwise, display a screen where the first line of the display shall be some portion of:
 38133 $(\text{current first line}) + \text{count} \times ((\text{window edit option}) - 2)$
 38134 If this calculation would result in a line that is after the last line of the edit buffer, the last line of
 38135 the display shall display some portion of the last line of the edit buffer.

38136 *Current line*: If no lines from the previous display remain on the screen, set to the first line of the
 38137 display; otherwise, set to (*line* + the number of new lines displayed on this screen).

38138 *Current column*: Set to non-<blank>.

38139 **Display Information**

38140 *Synopsis*: <control>-G

38141 This command shall be equivalent to the *ex file* command .

38142 **Move Cursor Backwards**

38143 *Synopsis*: [*count*] <control>-H

38144 [*count*] h

38145 the current *erase* character (see *stty*)

38146 If there are no characters before the current character on the current line, it shall be an error. If
 38147 there are less than *count* previous characters on the current line, *count* shall be adjusted to the
 38148 number of previous characters on the line.

38149 If used as a motion command:

38150 1. The text region shall be from the character before the starting cursor up to and including
 38151 the *count*th character before the starting cursor.

38152 2. Any text copied to a buffer shall be in character mode.

38153 If not used as a motion command:

38154 *Current line*: Unchanged.

38155 *Current column*: Set to (*column* – the number of columns occupied by *count* characters ending
 38156 with the previous current column).

38157 **Move Down**

38158 *Synopsis*: [*count*] <newline>

38159 [*count*] <control>-J

38160 [*count*] <control>-M

38161 [*count*] <control>-N

38162 [*count*] j

38163 [*count*] <carriage-return>

38164 [*count*] +

38165 If there are less than *count* lines after the current line in the edit buffer, it shall be an error.

38166 If used as a motion command:

38167 1. The text region shall include the starting line and the next *count* – 1 lines.

38168 2. Any text copied to a buffer shall be in line mode.

38169 If not used as a motion command:

38170 *Current line*: Set to *current line*+ *count*.

38171 *Current column*: Set to non-<blank> for the <carriage-return> character, <control>-M, and +
 38172 commands; otherwise, unchanged.

38173 **Clear and Redisplay**38174 *Synopsis:* <control>-L38175 If in open mode, clear the screen and redisplay the current line. Otherwise, clear and redisplay
38176 the screen.38177 *Current line:* Unchanged.38178 *Current column:* Unchanged.38179 **Move Up**38180 *Synopsis:* [*count*] <control>-P38181 [*count*] k38182 [*count*] -38183 If there are less than *count* lines before the current line in the edit buffer, it shall be an error.

38184 If used as a motion command:

- 38185 1. The text region shall include the starting line and the previous *count* lines.
- 38186 2. Any text copied to a buffer shall be in line mode.

38187 If not used as a motion command:

38188 *Current line:* Set to *current line* - *count*.38189 *Current column:* Set to non-<blank> for the - command; otherwise, unchanged.38190 **Redraw Screen**38191 *Synopsis:* <control>-R38192 If any lines have been deleted from the logical screen and flagged as deleted on the terminal
38193 using the @ convention (see the beginning of the EXTENDED DESCRIPTION section), they shall
38194 be redisplayed to match the contents of the edit buffer.38195 It is unspecified whether lines flagged with @ because they do not fit on the terminal display
38196 shall be affected.38197 *Current line:* Unchanged.38198 *Current column:* Unchanged.38199 **Scroll Backward**38200 *Synopsis:* [*count*] <control>-U

38201 If the current line is the first line of the edit buffer, it shall be an error.

38202 If no *count* is specified, *count* shall default to the *count* associated with the previous <control>-D
38203 or <control>-U command. If there was no previous <control>-D or <control>-U command, *count*
38204 shall default to the value of the **scroll** edit option.38205 *Current line:* If *count* is greater than the current line, set to 1; otherwise, set to the current line -
38206 *count*.38207 *Current column:* Set to non-<blank>.

38208 **Scroll Backward by Line**38209 *Synopsis:* [count] <control>-Y38210 Display the line *count* lines before the first line currently displayed.38211 If the current line is the first line of the edit buffer, it shall be an error. If this calculation would
38212 result in a line that is before the first line of the edit buffer, the first line of the display shall
38213 display some portion of the first line of the edit buffer.38214 *Current line:* Unchanged if the previous current character is displayed; otherwise, set to the first
38215 line displayed.38216 *Current column:* Unchanged.38217 **Edit the Alternate File**38218 *Synopsis:* <control>-^38219 This command shall be equivalent to the *ex edit* command, with the alternate path name as its
38220 argument.38221 **Terminate Command or Input Mode**38222 *Synopsis:* <ESC>38223 If a partial *vi* command (as defined by at least one, non-*count* character) has been entered,
38224 discard the *count* and the command character(s).38225 Otherwise, if no command characters have been entered, and the <ESC> was the result of a map
38226 expansion, the terminal shall be alerted and the <ESC> character shall be discarded, but it shall
38227 not be an error.

38228 Otherwise, it shall be an error.

38229 *Current line:* Unchanged.38230 *Current column:* Unchanged.38231 **Search for tagstring**38232 *Synopsis:* <control>-]

38233 If the current character is not a word or <blank> character, it shall be an error.

38234 This command shall be equivalent to the *ex tag* command, with the argument to that command
38235 defined as follows.

38236 If the current character is a <blank> character:

- 38237
1. Skip all <blank> characters after the cursor up to the end of the line.
 2. If the end of the line is reached, it shall be an error.
- 38238

38239 Then, the argument to the *ex tag* command shall be the current character and all subsequent
38240 characters, up to the first non-word character or the end of the line.

38241 **Move Cursor Forward**

38242 *Synopsis:* [*count*] <space>
 38243 [*count*] 1 (ell)

38244 If there are less than *count* characters after the cursor on the current line, *count* shall be adjusted
 38245 to the number of characters after the cursor on the line.

38246 If used as a motion command:

- 38247 1. If the current or *count*th character after the cursor is the last character in the line, the text
 38248 region shall be comprised of the current character up to and including the last character in
 38249 the line. Otherwise, the text region shall be from the current character up to, but not
 38250 including, the *count*th character after the cursor.
- 38251 2. Any text copied to a buffer shall be in character mode.

38252 If not used as a motion command:

38253 If there are no characters after the current character on the current line, it shall be an error.

38254 *Current line:* Unchanged.

38255 *Current column:* Set to the last column that displays any portion of the *count*th character after the
 38256 current character.

38257 **Replace Text with Results from Shell Command**

38258 *Synopsis:* [*count*] ! *motion shell-commands* <newline>

38259 If the motion command is the ! command repeated:

- 38260 1. If the edit buffer is empty and no *count* was supplied, the command shall be the equivalent
 38261 of the *ex:read!* command, with the text input, and no text shall be copied to any buffer.
- 38262 2. Otherwise:
 - 38263 a. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be
 38264 an error.
 - 38265 b. The text region shall be from the current line up to and including the next *count* - 1
 38266 lines.

38267 Otherwise, the text region shall be the lines in which any character of the text region specified by
 38268 the motion command appear.

38269 Any text copied to a buffer shall be in line mode.

38270 This command shall be equivalent to the *ex!* command for the specified lines.

38271 **Move Cursor to End-of-line**

38272 *Synopsis:* [*count*] \$

38273 It shall be an error if there are less than (*count* - 1) lines after the current line in the edit buffer.

38274 If used as a motion command:

- 38275 1. If *count* is 1:
 - 38276 a. It shall be an error if the line is empty.
 - 38277 b. Otherwise, the text region shall consist of all characters from the starting cursor to
 38278 the last character in the line, inclusive, and any text copied to a buffer shall be in

- 38279 character mode.
- 38280 2. Otherwise, if the starting cursor position is at or before the first non-<blank> character in
38281 the line, the text region shall consist of the current and the next *count* - 1 lines, and any text
38282 saved to a buffer shall be in line mode.
- 38283 3. Otherwise, the text region shall consist of all characters from the starting cursor to the last
38284 character in the line that is *count* - 1 lines forward from the current line, and any text copied
38285 to a buffer shall be in character mode.
- 38286 If not used as a motion command:
- 38287 *Current line*: Set to the *current line* + *count* - 1.
- 38288 *Current column*: The current column is set to the last screen column of the last character in the
38289 line, or column position 1 if the line is empty.
- 38290 The current column shall be adjusted to be on the last screen column of the last character of the
38291 current line as subsequent commands change the current line, until a command changes the
38292 current column.
- 38293 **Move to Matching Character**
- 38294 *Synopsis*: %
- 38295 If the character at the current position is not a parenthesis, bracket, or curly brace, search
38296 forward in the line to the first one of those characters. If no such character is found, it shall be an
38297 error.
- 38298 The matching character shall be the parenthesis, bracket, or curly brace matching the
38299 parenthesis, bracket, or curly brace, respectively, that was at the current position or that was
38300 found on the current line.
- 38301 Matching shall be determined as follows, for an open parenthesis:
- 38302 1. Set a counter to 1.
- 38303 2. Search forwards until a parenthesis is found or the end of the edit buffer is reached.
- 38304 3. If the end of the edit buffer is reached, it shall be an error.
- 38305 4. If an open parenthesis is found, increment the counter by 1.
- 38306 5. If a close parenthesis is found, decrement the counter by 1.
- 38307 6. If the counter is zero, the current character is the matching character.
- 38308 Matching for a close parenthesis shall be equivalent, except that the search shall be backwards,
38309 from the starting character to the beginning of the buffer, a close parenthesis shall increment the
38310 counter by 1, and an open parenthesis shall decrement the counter by 1.
- 38311 Matching for brackets and curly braces shall be equivalent, except that searching shall be done
38312 for open and close brackets or open and close curly braces. It is implementation-defined whether
38313 other characters are searched for and matched as well.
- 38314 If used as a motion command:
- 38315 1. If the matching cursor was after the starting cursor in the edit buffer, and the starting
38316 cursor position was at or before the first non-<blank> character in the starting line, and the
38317 matching cursor position was at or after the last non-<blank> character in the matching
38318 line, the text region shall consist of the current line to the matching line, inclusive, and any
38319 text copied to a buffer shall be in line mode.

38320 2. If the matching cursor was before the starting cursor in the edit buffer, and the starting
 38321 cursor position was at or after the last non-<blank> character in the starting line, and the
 38322 matching cursor position was at or before the first non-<blank> character in the matching
 38323 line, the text region shall consist of the current line to the matching line, inclusive, and any
 38324 text copied to a buffer shall be in line mode.

38325 3. Otherwise, the text region shall consist of the starting character to the matching character,
 38326 inclusive, and any text copied to a buffer shall be in character mode.

38327 If not used as a motion command:

38328 *Current line*: Set to the line where the matching character is located.

38329 *Current column*: Set to the last column where any portion of the matching character is displayed.

38330 **Repeat Substitution**

38331 *Synopsis*: &

38332 Repeat the previous substitution command. This command shall be equivalent to the *ex &*
 38333 command with the current line as its addresses, and without *options*, *count*, or *flags*.

38334 **Return to Previous Context at Beginning of Line**

38335 *Synopsis*: ' *character*

38336 It shall be an error if there is no line in the edit buffer marked by *character*.

38337 If used as a motion command:

38338 1. If the starting cursor is after the marked cursor, then the locations of the starting cursor
 38339 and the marked cursor in the edit buffer shall be logically swapped.

38340 2. The text region shall consist of the starting line up to and including the marked line, and
 38341 any text copied to a buffer shall be in line mode.

38342 If not used as a motion command:

38343 *Current line*: Set to the line referenced by the mark.

38344 *Current column*: Set to non-<blank>.

38345 **Return to Previous Context**

38346 *Synopsis*: ` *character*

38347 It shall be an error if the marked line is no longer in the edit buffer. If the marked line no longer
 38348 contains a character in the saved numbered character position, it shall be as if the marked
 38349 position is the first non-<blank> character.

38350 If used as a motion command:

38351 1. It shall be an error if the marked cursor references the same character in the edit buffer as
 38352 the starting cursor.

38353 2. If the starting cursor is after the marked cursor, then the locations of the starting cursor
 38354 and the marked cursor in the edit buffer shall be logically swapped.

38355 3. If the starting line is empty or the starting cursor is at or before the first non-<blank>
 38356 character of the starting line, and the marked cursor line is empty or the marked cursor
 38357 references the first character of the marked cursor line, the text region shall consist of all
 38358 lines containing characters from the starting cursor to the line before the marked cursor

- 38359 line, inclusive, and any text copied to a buffer shall be in line mode.
- 38360 4. Otherwise, if the marked cursor line is empty or the marked cursor references a character
38361 at or before the first non-<blank> character of the marked cursor line, the region of text
38362 shall be from the starting cursor to the last character of the line before the marked cursor
38363 line, inclusive, and any text copied to a buffer shall be in character mode.
- 38364 5. Otherwise, the region of text shall be from the starting cursor (inclusive), to the marked
38365 cursor (exclusive), and any text copied to a buffer shall be in character mode.

38366 If not used as a motion command:

38367 *Current line*: Set to the line referenced by the mark.

38368 *Current column*: Set to the last column in which any portion of the character referenced by the
38369 mark is displayed.

38370 **Return to Previous Section**

38371 *Synopsis*: [[

38372 Move the cursor backward through the edit buffer to the first character of the previous section
38373 boundary, *count* times.

38374 If used as a motion command:

- 38375 1. If the starting cursor was at the first character of the starting line or the starting line was
38376 empty, and the first character of the boundary was the first character of the boundary line,
38377 the text region shall consist of the current line up to and including the line where the
38378 *countth* next boundary starts, and any text copied to a buffer shall be in line mode.
- 38379 2. If the boundary was the last line of the edit buffer or the last character of the last line of the
38380 edit buffer, the text region shall consist of the last character in the edit buffer up to and
38381 including the starting character, and any text saved to a buffer shall be in character mode.
- 38382 3. Otherwise, the text region shall consist of the starting character up to but not including the
38383 first character in the *countth* next boundary, and any text copied to a buffer shall be in
38384 character mode.

38385 If the *lisp* option is set, a section boundary is also identified by a line with a leading ' (' .

38386 If not used as a motion command:

38387 *Current line*: Set to the line where the *countth* next boundary in the edit buffer starts.

38388 *Current column*: Set to the last column in which any portion of the first character of the *countth*
38389 next boundary is displayed, or column position 1 if the line is empty.

38390 **Move to Next Section**

38391 *Synopsis*:]]

38392 Move the cursor forward through the edit buffer to the first character of the next section
38393 boundary, *count* times.

38394 If used as a motion command:

- 38395 1. If the starting cursor was at the first character of the starting line or the starting line was
38396 empty, and the first character of the boundary was the first character of the boundary line,
38397 the text region shall consist of the current line up to and including the line where the
38398 *countth* previous boundary starts, and any text copied to a buffer shall be in line mode.

38399 2. If the boundary was the first line of the edit buffer, the text region shall consist of the first
 38400 character in the edit buffer up to but not including the starting character, and any text
 38401 copied to a buffer shall be in character mode.

38402 3. Otherwise, the text region shall consist of the first character in the *count*th previous section
 38403 boundary up to but not including the starting character, and any text copied to a buffer
 38404 shall be in character mode.

38405 If the *lisp* option is set, a section boundary is also identified by a line with a leading ' (' .

38406 If not used as a motion command:

38407 *Current line*: Set to the line where the *count*th previous boundary in the edit buffer starts.

38408 *Current column*: Set to the last column in which any portion of the first character of the *count*th
 38409 previous boundary is displayed, or column position 1 if the line is empty.

38410 **Move to First Non-<blank> Position on Current Line**

38411 *Synopsis*: ^

38412 If used as a motion command:

38413 1. If the line has no non-<blank> characters, or if the cursor is at the first non-<blank>
 38414 character of the line, it shall be an error.

38415 2. If the cursor is before the first non-<blank> character of the line, the text region shall be
 38416 comprised of the current character, up to, but not including, the first non-<blank>
 38417 character of the line.

38418 3. If the cursor is after the first non-<blank> character of the line, the text region shall be from
 38419 the character before the starting cursor up to and including the first non-<blank> character
 38420 of the line.

38421 4. Any text copied to a buffer shall be in character mode.

38422 If not used as a motion command:

38423 *Current line*: Unchanged.

38424 *Current column*: Set to non-<blank>.

38425 **Current and line above**

38426 *Synopsis*: [*count*] _

38427 If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.

38428 If used as a motion command:

38429 1. If *count* is less than 2, the text region shall be the current line.

38430 2. Otherwise, the text region shall include the starting line and the next *count* - 1 lines.

38431 3. Any text copied to a buffer shall be in line mode.

38432 If not used as a motion command:

38433 *Current line*: Set to current line + *count* - 1.

38434 *Current column*: Set to non-<blank>.

38435 **Move Back to Beginning of Sentence**38436 *Synopsis:* [count] (38437 Move backward to the beginning of a sentence. This command shall be equivalent to the [[
38438 command, with the exception that sentence boundaries shall be used instead of section
38439 boundaries.38440 If the *lisp* option is set, a LISP s-expression is considered a sentence for this command.38441 **Move Forward to Beginning of Sentence**38442 *Synopsis:* [count])38443 Move forward to the beginning of a sentence. This command shall be equivalent to the]]
38444 command, with the exception that sentence boundaries shall be used instead of section
38445 boundaries.38446 If the *lisp* option is set, a LISP s-expression is considered a sentence for this command.38447 **Move Back to Preceding Paragraph**38448 *Synopsis:* [count] {38449 Move back to the beginning of the preceding paragraph. This command shall be equivalent to
38450 the [[command, with the exception that paragraph boundaries shall be used instead of section
38451 boundaries.38452 **Move Forward to Next Paragraph**38453 *Synopsis:* [count] }38454 Move forward to the beginning of the next paragraph. This command shall be equivalent to the
38455]] command, with the exception that paragraph boundaries shall be used instead of section
38456 boundaries.38457 **Move to Specific Column Position**38458 *Synopsis:* [count] |38459 For the purposes of this command, lines that are too long for the current display and that have
38460 been folded shall be treated as having a single, 1-based, number of columns.38461 If there are less than *count* columns in which characters from the current line are displayed on
38462 the screen, *count* shall be adjusted to be the last column in which any portion of the line is
38463 displayed on the screen.

38464 If used as a motion command:

- 38465 1. If the line is empty, or the cursor character is the same as the character on the *count*th
38466 column of the line, it shall be an error.
- 38467 2. If the cursor is before the *count*th column of the line, the text region shall be comprised of
38468 the current character, up to but not including the character on the *count*th column of the
38469 line.
- 38470 3. If the cursor is after the *count*th column of the line, the text region shall be from the
38471 character before the starting cursor up to and including the character on the *count*th
38472 column of the line.

38473 4. Any text copied to a buffer shall be in character mode.

38474 If not used as a motion command:

38475 *Current line*: Unchanged.

38476 *Current column*: Set to the last column in which any portion of the character that is displayed in
38477 the *count* column of the line is displayed.

38478 **Reverse Find Character**

38479 *Synopsis*: [*count*] ,

38480 If the last **F**, **f**, **T**, or **t** command was **F**, **f**, **T**, or **t**, this command shall be equivalent to an **f**, **F**, **t**, or
38481 **T** command, respectively, with the specified *count* and the same search character.

38482 If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

38483 **Repeat**

38484 *Synopsis*: [*count*] .

38485 Repeat the last **!**, **<**, **>**, **A**, **C**, **D**, **I**, **J**, **O**, **P**, **R**, **S**, **X**, **Y**, **a**, **c**, **d**, **i**, **o**, **p**, **r**, **s**, **x**, **y**, or **~** command. It shall
38486 be an error if none of these commands have been executed. Commands (other than commands
38487 that enter text input mode) executed as a result of map expansions, shall not change the value of
38488 the last repeatable command.

38489 Repeated commands with associated motion commands shall repeat the motion command as
38490 well; however, any specified *count* shall replace the *count*(s) that were originally specified to the
38491 repeated command or its associated motion command.

38492 If the motion component of the repeated command is **f**, **F**, **t**, or **T**, the repeated command shall
38493 not set the remembered search character for the **;** and **,** commands.

38494 If the repeated command is **p** or **P**, and the buffer associated with that command was a numeric
38495 buffer named with a number less than 9, the buffer associated with the repeated command shall
38496 be set to be the buffer named by the name of the previous buffer logically incremented by 1.

38497 If the repeated character is a text input command, the input text associated with that command
38498 is repeated literally:

- 38499 • Input characters are neither macro or abbreviation-expanded.
- 38500 • Input characters are not interpreted in any special way with the exception that the <newline>
38501 character and the <carriage-return> character, and <control>-T behave as described in **Input**
38502 **Mode Commands in vi** (on page 3235).

38503 *Current line*: Set as described for the repeated command.

38504 *Current column*: Set as described for the repeated command.

38505 **Find Regular Expression**

38506 *Synopsis*: /

38507 If the input line contains no characters, it shall be equivalent to a line containing only the last
38508 regular expression encountered. The enhanced regular expressions supported by *vi* are
38509 described in **Regular Expressions in ex** (on page 2601).

38510 Otherwise, the line shall be interpreted as one or more regular expressions, optionally followed
38511 by an address offset or a *vi z* command.

38512 If the regular expression is not the last regular expression on the line, or if a line offset or **z**
 38513 command is specified, the regular expression shall be terminated by an unescaped `'/'`
 38514 character, which shall not be used as part of the regular expression. If the regular expression is
 38515 not the first regular expression on the line, it shall be preceded by zero or more `<blank>`
 38516 characters, a semicolon, zero or more `<blank>` characters, and a leading `'/'` character, which
 38517 shall not be interpreted as part of the regular expression. It shall be an error to precede any
 38518 regular expression with any characters other than these.

38519 Each search shall begin from the character after the first character of the last match (or, if it is the
 38520 first search, after the cursor). If the **wrapsan** edit option is set, the search shall continue to the
 38521 character before the starting cursor character; otherwise, to the end of the edit buffer. It shall be
 38522 an error if any search fails to find a match, and an informational message to this effect shall be
 38523 displayed.

38524 An optional address offset (see **Addressing in ex** (on page 2571)) can be specified after the last
 38525 regular expression by including a trailing `'/'` character after the regular expression and
 38526 specifying the address offset. This offset will be from the line containing the match for the last
 38527 regular expression specified. It shall be an error if the line offset would indicate a line address
 38528 less than 1 or greater than the last line in the edit buffer. An address offset of zero shall be
 38529 supported. It shall be an error to follow the address offset with any other characters than
 38530 `<blank>` characters.

38531 If not used as a motion command, an optional **z** command (see **Redraw Window** (on page 3234))
 38532 can be specified after the last regular expression by including a trailing `'/'` character after the
 38533 regular expression, zero or more `<blank>` characters, a `'z'`, zero or more `<blank>` characters, an
 38534 optional new **window** edit option value, zero or more `<blank>` characters, and a location
 38535 character. The effect shall be as if the **z** command was executed after the `/` command. It shall be
 38536 an error to follow the **z** command with any other characters than `<blank>` characters.

38537 The remembered search direction shall be set to forward.

38538 If used as a motion command:

- 38539 1. It shall be an error if the last match references the same character in the edit buffer as the
 38540 starting cursor.
- 38541 2. If any address offset is specified, the last match shall be adjusted by the specified offset as
 38542 described previously.
- 38543 3. If the starting cursor is after the last match, then the locations of the starting cursor and the
 38544 last match in the edit buffer shall be logically swapped.
- 38545 4. If any address offset is specified, the text region shall consist of all lines containing
 38546 characters from the starting cursor to the last match line, inclusive, and any text copied to a
 38547 buffer shall be in line mode.
- 38548 5. Otherwise, if the starting line is empty or the starting cursor is at or before the first non-
 38549 `<blank>` character of the starting line, and the last match line is empty or the last match
 38550 starts at the first character of the last match line, the text region shall consist of all lines
 38551 containing characters from the starting cursor to the line before the last match line,
 38552 inclusive, and any text copied to a buffer shall be in line mode.
- 38553 6. Otherwise, if the last match line is empty or the last match begins at a character at or
 38554 before the first non-`<blank>` of the last match line, the region of text shall be from the
 38555 current cursor to the last character of the line before the last match line, inclusive, and any
 38556 text copied to a buffer shall be in character mode.

38557 7. Otherwise, the region of text shall be from the current cursor (inclusive), to the first
38558 character of the last match (exclusive), and any text copied to a buffer shall be in
38559 character mode.

38560 If not used as a motion command:

38561 *Current line*: If a match is found, set to the last matched line plus the address offset, if any;
38562 otherwise, unchanged.

38563 *Current column*: Set to the last column on which any portion of the first character in the last
38564 matched string is displayed, if a match is found; otherwise, unchanged.

38565 **Move to First Character in Line**

38566 *Synopsis*: 0 (zero)

38567 Move to the first character on the current line. The character '0' shall not be interpreted as a
38568 command if it is immediately preceded by a digit.

38569 If used as a motion command:

- 38570 1. If the cursor character is the first character in the line, it shall be an error.
- 38571 2. The text region shall be from the character before the cursor character up to and including
38572 the first character in the line.
- 38573 3. Any text copied to a buffer shall be in character mode.

38574 If not used as a motion command:

38575 *Current line*: Unchanged.

38576 *Current column*: The last column in which any portion of the first character in the line is
38577 displayed, or if the line is empty, unchanged.

38578 **Execute an ex Command**

38579 *Synopsis*: :

38580 Execute one or more *ex* commands.

38581 If any portion of the screen other than the last line of the screen was overwritten by any *ex*
38582 command (except **shell**), *vi* shall display a message indicating that it is waiting for an input from
38583 the user, and shall then read a character. This action may also be taken for other, unspecified
38584 reasons.

38585 If the next character entered is a ':', another *ex* command shall be accepted and executed. Any
38586 other character shall cause the screen to be refreshed and *vi* shall return to command mode.

38587 *Current line*: As specified for the *ex* command.

38588 *Current column*: As specified for the *ex* command.

38589 **Repeat Find**38590 *Synopsis:* [*count*] ;38591 This command shall be equivalent to the last **F**, **f**, **T**, or **t** command, with the specified *count*, and
38592 with the same search character used for the last **F**, **f**, **T**, or **t** command. If there was no previous **F**,
38593 **f**, **T**, or **t** command, it shall be an error.38594 **Shift Left**38595 *Synopsis:* [*count*] < *motion*

38596 If the motion command is the < command repeated:

- 38597 1. If there are less than
- count*
- 1 lines after the current line in the edit buffer, it shall be an
-
- 38598 error.
-
- 38599 2. The text region shall be from the current line, up to and including the next
- count*
- 1 lines.

38600 Shift any line in the text region specified by the *count* and motion command one shiftwidth (see
38601 the *ex* **shiftwidth** option) toward the start of the line, as described by the *ex* < command. The
38602 unshifted lines shall be copied to the unnamed buffer in line mode.38603 *Current line:* If the motion was from the current cursor position toward the end of the edit
38604 buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region
38605 specified by the motion command.38606 *Current column:* Set to non-<blank>.38607 **Shift Right**38608 *Synopsis:* [*count*] > *motion*

38609 If the motion command is the > command repeated:

- 38610 1. If there are less than
- count*
- 1 lines after the current line in the edit buffer, it shall be an
-
- 38611 error.
-
- 38612 2. The text region shall be from the current line, up to and including the next
- count*
- 1 lines.

38613 Shift any line with characters in the text region specified by the *count* and motion command one
38614 shiftwidth (see the *ex* **shiftwidth** option) away from the start of the line, as described by the *ex* >
38615 command. The unshifted lines shall be copied into the unnamed buffer in line mode.38616 *Current line:* If the motion was from the current cursor position toward the end of the edit
38617 buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region
38618 specified by the motion command.38619 *Current column:* Set to non-<blank>.38620 **Scan Backwards for Regular Expression**38621 *Synopsis:* ?38622 Scan backwards; The ? command shall be equivalent to the / command (see **Find Regular**
38623 **Expression** (on page 3216)) with the following exceptions:

- 38624 1. The input prompt shall be a ' ? '.
-
- 38625 2. Each search shall begin from the character before the first character of the last match (or, if
-
- 38626 it is the first search, the character before the cursor character).

- 38627 3. The search direction shall be from the cursor toward the beginning of the edit buffer, and
 38628 the **wrapsan** edit option shall affect whether the search wraps to the end of the edit buffer
 38629 and continues.
- 38630 4. The remembered search direction shall be set to backward.

38631 **Execute**

38632 *Synopsis:* @*buffer*

38633 If the *buffer* is specified as @, the last buffer executed shall be used. If no previous buffer has been
 38634 executed, it shall be an error.

38635 Behave as if the contents of the named buffer were entered as standard input. After each line of a
 38636 line-mode buffer, and all but the last line of a character mode buffer, behave as if a <newline>
 38637 character were entered as standard input.

38638 If an error occurs during this process, an error message shall be written, and no more characters
 38639 resulting from the execution of this command shall be processed.

38640 If a *count* is specified, behave as if that count were entered as user input before the characters
 38641 from the @ buffer were entered.

38642 *Current line:* As specified for the individual commands.

38643 *Current column:* As specified for the individual commands.

38644 **Reverse Case**

38645 *Synopsis:* [*count*] ~

38646 Reverse the case of the current character and the next *count* - 1 characters, such that lowercase
 38647 characters that have uppercase counterparts shall be changed to uppercase characters, and
 38648 uppercase characters that have lowercase counterparts shall be changed to lowercase characters,
 38649 as prescribed by the current locale. No other characters shall be affected by this command.

38650 If there are less than *count* - 1 characters after the cursor in the edit buffer, *count* shall be adjusted
 38651 to the number of characters after the cursor in the edit buffer minus 1.

38652 For the purposes of this command, the next character after the last character on the line shall be
 38653 the next character in the edit buffer.

38654 *Current line:* Set to the line including the (*count*-1)th character after the cursor.

38655 *Current column:* Set to the last column in which any portion of the (*count*-1)th character after the
 38656 cursor is displayed.

38657 **Reindent**

38658 *Synopsis:* [*count*]=[*motion*]

38659 If the *lisp* option is set, reindents the specified lines, as though they were typed in with **lisp** and
 38660 **autoindent** set.

38661 *Current line:* Unchanged.

38662 *Current column:* Move to the first non-<blank> character of the line or the last character if the
 38663 line is a blank line.

38664 **Append**38665 *Synopsis:* [count] a

38666 Enter text input mode after the current cursor position. No characters already in the edit buffer
 38667 shall be affected by this command. A *count* shall cause the input text to be appended *count* -1
 38668 more times to the end of the input.

38669 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
 38670 (on page 3235)).

38671 **Append at End-of-Line**38672 *Synopsis:* [count] A38673 This command shall be equivalent to the *vi* command:

38674 \$ [count] a

38675 (see **Append**).38676 **Move Backward to Preceding Word**38677 *Synopsis:* [count] b

38678 With the exception that words are used as the delimiter instead of bigwords, this command shall
 38679 be equivalent to the **B** command.

38680 **Move Backward to Preceding Bigword**38681 *Synopsis:* [count] B

38682 If the edit buffer is empty or the cursor is on the first character of the edit buffer, it shall be an
 38683 error. If less than *count* bigwords begin between the cursor and the start of the edit buffer, *count*
 38684 shall be adjusted to the number of bigword beginnings between the cursor and the start of the
 38685 edit buffer.

38686 If used as a motion command:

38687 1. The text region shall be from the first character of the *count*th previous bigword beginning
 38688 up to but not including the cursor character.

38689 2. Any text copied to a buffer shall be in character mode.

38690 If not used as a motion command:

38691 *Current line:* Set to the line containing the *current column*.

38692 *Current column:* Set to the last column upon which any part of the first character of the *count*th
 38693 previous bigword is displayed.

38694 **Change**38695 *Synopsis:* [buffer][count] c motion38696 If the motion command is the **c** command repeated:

38697 1. The buffer text shall be in line mode.

38698 2. If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an
 38699 error.

- 38700 3. The text region shall be from the current line up to and including the next *count* -1 lines.
 38701 Otherwise, the buffer text mode and text region shall be as specified by the motion command.
 38702 The replaced text shall be copied into *buffer*, if specified, and into the unnamed buffer. If the text
 38703 to be replaced contains characters from more than a single line, or the buffer text is in line mode,
 38704 the replaced text shall be copied into the numeric buffers as well.
 38705 If the buffer text is in line mode:
 38706 1. Any lines that contain characters in the region shall be deleted, and the editor shall enter
 38707 text input mode at the beginning of a new line which shall replace the first line deleted.
 38708 2. If the **autoindent** edit option is set, **autoindent** characters equal to the **autoindent**
 38709 characters on the first line deleted shall be inserted as if entered by the user.
 38710 Otherwise, if characters from more than one line are in the region of text:
 38711 1. The text shall be deleted.
 38712 2. Any text remaining in the last line in the text region shall be appended to the first line in
 38713 the region, and the last line in the region shall be deleted.
 38714 3. The editor shall enter text input mode after the last character not deleted from the first line
 38715 in the text region, if any; otherwise, on the first column of the first line in the region.
 38716 Otherwise:
 38717 1. If the glyph for ' \$ ' is smaller than the region, the end of the region shall be marked with a
 38718 ' \$ ' .
 38719 2. The editor shall enter text input mode, overwriting the region of text.
 38720 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
 38721 (on page 3235)).
 38722 **Change to End-of-Line**
 38723 *Synopsis:* [*buffer*][*count*] C
 38724 This command shall be equivalent to the *vi* command:
 38725 [*buffer*][*count*] c\$
 38726 See the **c** command.
 38727 **Delete**
 38728 *Synopsis:* [*buffer*][*count*] d *motion*
 38729 If the motion command is the **d** command repeated:
 38730 1. The buffer text shall be in line mode.
 38731 2. If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an
 38732 error.
 38733 3. The text region shall be from the current line up to and including the next *count* -1 lines.
 38734 Otherwise, the buffer text mode and text region shall be as specified by the motion command.
 38735 If in open mode, and the current line is deleted, and the line remains on the display, an '@'
 38736 character shall be displayed as the first glyph of that line.

38737 Delete the region of text into *buffer*, if specified, and into the unnamed buffer. If the text to be
 38738 deleted contains characters from more than a single line, or the buffer text is in line mode, the
 38739 deleted text shall be copied into the numeric buffers, as well.

38740 *Current line*: Set to the first text region line that appears in the edit buffer, unless that line has
 38741 been deleted, in which case it shall be set to the last line in the edit buffer, or line 1 if the edit
 38742 buffer is empty.

38743 *Current column*:

- 38744 1. If the line is empty, set to column position 1.
- 38745 2. Otherwise, if the buffer text is in line mode or the motion was from the cursor toward the
 38746 end of the edit buffer:
 - 38747 a. If a character from the current line is displayed in the current column, set to the last
 38748 column that displays any portion of that character.
 - 38749 b. Otherwise, set to the last column in which any portion of any character in the line is
 38750 displayed.
- 38751 3. Otherwise, if a character is displayed in the column that began the text region, set to the
 38752 last column that displays any portion of that character.
- 38753 4. Otherwise, set to the last column in which any portion of any character in the line is
 38754 displayed.

38755 **Delete to End-of-Line**

38756 *Synopsis*: [*buffer*] D

38757 Delete the text from the current position to the end of the current line; equivalent to the *vi*
 38758 command:

38759 [*buffer*] d\$

38760 **Move to End-of-Word**

38761 *Synopsis*: [*count*] e

38762 With the exception that words are used instead of bigwords as the delimiter, this command shall
 38763 be equivalent to the **E** command.

38764 **Move to End-of-Bigword**

38765 *Synopsis*: [*count*] E

38766 If the edit buffer is empty it shall be an error. If less than *count* bigwords end between the cursor
 38767 and the end of the edit buffer, *count* shall be adjusted to the number of bigword endings between
 38768 the cursor and the end of the edit buffer.

38769 If used as a motion command:

- 38770 1. The text region shall be from the last character of the *count*th next bigword up to and
 38771 including the cursor character.
- 38772 2. Any text copied to a buffer shall be in character mode.

38773 If not used as a motion command:

38774 *Current line*: Set to the line containing the current column.

38775 *Current column*: Set to the last column upon which any part of the last character of the *countth*
 38776 next bigword is displayed.

38777 **Find Character in Current Line (Forward)**

38778 *Synopsis*: [*count*] f *character*

38779 It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

38780 If used as a motion command:

38781 1. The text range shall be from the cursor character up to and including the *countth*
 38782 occurrence of the specified character after the cursor.

38783 2. Any text copied to a buffer shall be in character mode.

38784 If not used as a motion command:

38785 *Current line*: Unchanged.

38786 *Current column*: Set to the last column in which any portion of the *countth* occurrence of the
 38787 specified character after the cursor appears in the line.

38788 **Find Character in Current Line (Reverse)**

38789 *Synopsis*: [*count*] F *character*

38790 It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

38791 If used as a motion command:

38792 1. The text region shall be from the *countth* occurrence of the specified character before the
 38793 cursor, up to, but not including the cursor character.

38794 2. Any text copied to a buffer shall be in character mode.

38795 If not used as a motion command:

38796 *Current line*: Unchanged.

38797 *Current column*: Set to the last column in which any portion of the *countth* occurrence of the
 38798 specified character before the cursor appears in the line.

38799 **Move to Line**

38800 *Synopsis*: [*count*] G

38801 If *count* is not specified, it shall default to the last line of the edit buffer. If *count* is greater than
 38802 the last line of the edit buffer, it shall be an error.

38803 If used as a motion command:

38804 1. The text region shall be from the cursor line up to and including the specified line.

38805 2. Any text copied to a buffer shall be in line mode.

38806 If not used as a motion command:

38807 *Current line*: Set to *count* if *count* is specified; otherwise, the last line.

38808 *Current column*: Set to non-<blank>.

38809 **Move to Top of Screen**38810 *Synopsis:* [count] H38811 If the beginning of the line *count* greater than the first line of which any portion appears on the
38812 display does not exist, it shall be an error.

38813 If used as a motion command:

- 38814 1. If in open mode, the text region shall be the current line.
- 38815 2. Otherwise, the text region shall be from the starting line up to and including (the first line
38816 of the display + *count* - 1).
- 38817 3. Any text copied to a buffer shall be in line mode.

38818 If not used as a motion command:

38819 If in open mode, this command shall set the current column to non-<blank> and do nothing else.

38820 Otherwise, it shall set the current line and current column as follows.

38821 *Current line:* Set to (the first line of the display + *count* - 1).38822 *Current column:* Set to non-<blank>.38823 **Insert Before Cursor**38824 *Synopsis:* [count] i38825 Enter text input mode before the current cursor position. No characters already in the edit buffer
38826 shall be affected by this command. A *count* shall cause the input text to be appended *count* - 1
38827 more times to the end of the input.38828 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
38829 (on page 3235)).38830 **Insert at Beginning of Line**38831 *Synopsis:* [count] I38832 This command shall be equivalent to the *vi* command `^[count]i` command.38833 **Join**38834 *Synopsis:* [count] J

38835 If the current line is the last line in the edit buffer, it shall be an error.

38836 This command shall be equivalent to the *ex* **join** command with no addresses, and an *ex*
38837 command *count* value of 1 if *count* was not specified or if a *count* of 1 was specified, and an *ex*
38838 command *count* value of *count* - 1 for any other value of *count*, except that the current line and
38839 column shall be set as follows.38840 *Current line:* Unchanged.38841 *Current column:* The last column in which any portion of the character following the last
38842 character in the initial line is displayed, or the last character in the line if no characters were
38843 appended.

38844 **Move to Bottom of Screen**38845 *Synopsis:* [count] L38846 If the beginning of the line count less than the last line of which any portion appears on the
38847 display does not exist, it shall be an error.

38848 If used as a motion command:

- 38849 1. If in open mode, the text region shall be the current line.
- 38850 2. Otherwise, the text region shall include all lines from the starting cursor line to (the last
38851 line of the display $-(count - 1)$).
- 38852 3. Any text copied to a buffer shall be in line mode.

38853 If not used as a motion command:

- 38854 1. If in open mode, this command shall set the current column to non-<blank> and do
38855 nothing else.
- 38856 2. Otherwise, it shall set the current line and current column as follows.

38857 *Current line:* Set to (the last line of the display $-(count - 1)$).38858 *Current column:* Set to non-<blank>.38859 **Mark Position**38860 *Synopsis:* m letter38861 This command shall be equivalent to the *ex mark* command with the specified character as an
38862 argument.38863 **Move to Middle of Screen**38864 *Synopsis:* M

38865 The middle line of the display shall be calculated as follows:

38866 $(\text{the top line of the display}) + ((\text{number of lines displayed}) + 1) / 2) - 1$

38867 If used as a motion command:

- 38868 1. If in open mode, the text region shall be the current line.
- 38869 2. Otherwise, the text region shall include all lines from the starting cursor line up to and
38870 including the middle line of the display.
- 38871 3. Any text copied to a buffer shall be in line mode.

38872 If not used as a motion command:

38873 If in open mode, this command shall set the current column to non-<blank> and do nothing else.

38874 Otherwise, it shall set the current line and current column as follows.

38875 *Current line:* Set to the middle line of the display.38876 *Current column:* Set to non-<blank>.

38877 **Repeat Regular Expression Find (Forward)**38878 *Synopsis:* n

38879 If the remembered search direction was forward, the **n** command shall be equivalent to the *vi /*
 38880 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi ?*
 38881 command with no characters entered by the user.

38882 If the **n** command is used as a motion command for the **!** command, the editor shall not enter
 38883 text input mode on the last line on the screen, and shall behave as if the user entered a single **' ! '**
 38884 character as the text input.

38885 **Repeat Regular Expression Find (Reverse)**38886 *Synopsis:* N

38887 Scan for the next match of the last pattern given to */* or *?*, but in the reverse direction; this is the
 38888 reverse of **n**.

38889 If the remembered search direction was forward, the **N** command shall be equivalent to the *vi ?*
 38890 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi /*
 38891 command with no characters entered by the user. If the **N** command is used as a motion
 38892 command for the **!** command, the editor shall not enter text input mode on the last line on the
 38893 screen, and shall behave as if the user entered a single **!** character as the text input.

38894 **Insert Empty Line Below**38895 *Synopsis:* o

38896 Enter text input mode in a new line appended after the current line. A *count* shall cause the input
 38897 text to be appended *count* - 1 more times to the end of the already added text, each time starting
 38898 on a new, appended line.

38899 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
 38900 (on page 3235)).

38901 **Insert Empty Line Above**38902 *Synopsis:* O

38903 Enter text input mode in a new line inserted before the current line. A *count* shall cause the input
 38904 text to be appended *count* - 1 more times to the end of the already added text, each time starting
 38905 on a new, appended line.

38906 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
 38907 (on page 3235)).

38908 **Put from Buffer Following**38909 *Synopsis:* [*buffer*] p

38910 If no *buffer* is specified, the unnamed buffer shall be used.

38911 If the buffer text is in line mode, the text shall be appended below the current line, and each line
 38912 of the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be
 38913 appended *count* - 1 more times to the end of the already added text, each time starting on a new,
 38914 appended line.

38915 If the buffer text is in character mode, the text shall be appended into the current line after the
 38916 cursor, and each line of the buffer other than the first and last shall become a new line in the edit

38917 buffer. A *count* shall cause the buffer text to be appended *count* –1 more times to the end of the
38918 already added text, each time starting after the last added character.

38919 *Current line*: If the buffer text is in line mode, set the line to line +1; otherwise, unchanged.

38920 *Current column*: If the buffer text is in line mode:

- 38921 1. If there is a non-<blank> character in the first line of the buffer, set to the last column on
38922 which any portion of the first non-<blank> character in the line is displayed.
- 38923 2. If there is no non-<blank> character in the first line of the buffer, set to the last column on
38924 which any portion of the last character in the first line of the buffer is displayed.

38925 If the buffer text is in character mode:

- 38926 1. If the text in the buffer is from more than a single line, then set to the last column on which
38927 any portion of the first character from the buffer is displayed.
- 38928 2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion
38929 of the last character from the buffer is displayed.
- 38930 3. Otherwise, set to the first column on which any portion of the first character from the
38931 buffer is displayed.

38932 **Put from Buffer Before**

38933 *Synopsis*: [*buffer*] P

38934 If no *buffer* is specified, the unnamed buffer shall be used.

38935 If the buffer text is in line mode, the text shall be inserted above the current line, and each line of
38936 the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be
38937 appended *count* –1 more times to the end of the already added text, each time starting on a new,
38938 appended line.

38939 If the buffer text is in character mode, the text shall be inserted into the current line before the
38940 cursor, and each line of the buffer other than the first and last shall become a new line in the edit
38941 buffer. A *count* shall cause the buffer text to be appended *count* –1 more times to the end of the
38942 already added text, each time starting after the last added character.

38943 *Current line*: Unchanged.

38944 *Current column*: If the buffer text is in line mode:

- 38945 1. If there is a non-<blank> character in the first line of the buffer, set to the last column on
38946 which any portion of that character is displayed.
- 38947 2. If there is no non-<blank> character in the first line of the buffer, set to the last column on
38948 which any portion of the last character in the first line of the buffer is displayed.

38949 If the buffer text is in character mode:

- 38950 1. If the buffer is the unnamed buffer, set to the last column on which any portion of the last
38951 character from the buffer is displayed.
- 38952 2. Otherwise, set to the first column on which any portion of the first character from the
38953 buffer is displayed.

38954 **Enter ex Mode**38955 *Synopsis:* Q38956 Leave visual or open mode and enter *ex* command mode.38957 *Current line:* Unchanged.38958 *Current column:* Unchanged.38959 **Replace Character**38960 *Synopsis:* [*count*] r *character*38961 **Notes to Reviewers**38962 *This section with side shading will not appear in the final copy. - Ed.*38963 D3, XCU, ERN 270: The description of R is not correct. R is not the same as the i command,
38964 which is what the text describes. Something should be done here when .2b is approved.38965 Replace the *count* characters at and after the cursor with the specified character. If there are less
38966 than *count* characters at and after the cursor on the line, it shall be an error.38967 If character is <control>-V, any next character other than the <newline> shall be stripped of any
38968 special meaning and used as a literal character.38969 If character is <ESC>, no replacement shall be made and the current line and current column
38970 shall be unchanged.38971 If character is <carriage-return> or <newline>, *count* new lines shall be appended to the current
38972 line. All but the last of these lines shall be empty. *count* characters at and after the cursor shall be
38973 discarded, and any remaining characters after the cursor in the current line shall be moved to the
38974 last of the new lines. If the **autoindent** edit option is set, they shall be preceded by the same
38975 number of **autoindent** characters found on the line from which the command was executed.38976 *Current line:* Unchanged unless the replacement character is a <carriage-return> or <newline>
38977 character, in which case it shall be set to line + *count*.38978 *Current column:* Set to the last column position on which a portion of the last replaced character
38979 is displayed, or if the replacement character caused new lines to be created, set to non-<blank>.38980 **Replace Characters**38981 *Synopsis:* R38982 Enter text input mode at the current cursor position. A *count* shall cause the input text to be
38983 appended *count* - 1 more times to the end of the input.38984 *Current line/column:* As specified for the text input commands (see **Input Mode Commands in vi**
38985 (on page 3235)).

38986 Substitute Character

38987 *Synopsis:* [*buffer*][*count*] *s*

38988 This command shall be equivalent to the *vi* command:

38989 [*buffer*][*count*] c<space>

38990 Substitute Lines

38991 *Synopsis:* [*buffer*][*count*] *S*

38992 This command shall be equivalent to the *vi* command:

38993 [*buffer*][*count*] c_

38994 Move Cursor to Before Character (Forward)

38995 *Synopsis:* [*count*] t *character*

38996 It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

38997 If used as a motion command:

- 38998 1. The text region shall be from the cursor up to but not including the *count*th occurrence of
38999 the specified character after the cursor.
- 39000 2. Any text copied to a buffer shall be in character mode.

39001 If not used as a motion command:

39002 *Current line:* Unchanged.

39003 *Current column:* Set to the last column in which any portion of the character before the *count*th
39004 occurrence of the specified character after the cursor appears in the line.

39005 Move Cursor to After Character (Reverse)

39006 *Synopsis:* [*count*] T *character*

39007 It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

39008 If used as a motion command:

- 39009 1. If the character before the cursor is the specified character, it shall be an error.
- 39010 2. The text region shall be from the character before the cursor up to but not including the
39011 *count*th occurrence of the specified character before the cursor.
- 39012 3. Any text copied to a buffer shall be in character mode.

39013 If not used as a motion command:

39014 *Current line:* Unchanged.

39015 *Current column:* Set to the last column in which any portion of the character after the *count*th
39016 occurrence of the specified character before the cursor appears in the line.

39017 **Undo**39018 *Synopsis:* u39019 This command shall be equivalent to the *ex* **undo** command except that the current line and
39020 current column shall be set as follows:39021 *Current line:* Set to the first line added or changed if any; otherwise, move to the line preceding
39022 any deleted text if one exists; otherwise, move to line 1.39023 *Current column:* If undoing an *ex* command, set to the first non-<blank> character.

39024 Otherwise, if undoing a text input command:

39025 1. If the command was an **C**, **c**, **O**, **o**, **R**, **S**, or **s** command, the current column shall be set to
39026 the value it held when the text input command was entered.39027 2. Otherwise, set to the last column in which any portion of the first character after the
39028 deleted text is displayed, or, if no characters follow the text deleted from this line, set to the
39029 last column in which any portion of the last character in the line is displayed, or 1 if the line
39030 is empty.39031 Otherwise, if a single line was modified (that is, not added or deleted) by the **u** command:39032 1. If text was added or changed, set to the last column in which any portion of the first
39033 character added or changed is displayed.39034 2. If text was deleted, set to the last column in which any portion of the first character after
39035 the deleted text is displayed, or, if no characters follow the deleted text, set to the last
39036 column in which any portion of the last character in the line is displayed, or 1 if the line is
39037 empty.

39038 Otherwise, set to non-<blank>.

39039 **Undo Current Line**39040 *Synopsis:* U39041 Restore the current line to its state immediately before the most recent time that it became the
39042 current line.39043 *Current line:* Unchanged.39044 *Current column:* Set to the first column in the line in which any portion of the first character in
39045 the line is displayed.39046 **Move to Beginning of Word**39047 *Synopsis:* [*count*] w39048 With the exception that words are used as the delimiter instead of bigwords, this command shall
39049 be equivalent to the **W** command.

39050 **Move to Beginning of Bigword**39051 *Synopsis:* [*count*] W39052 If the edit buffer is empty, it shall be an error. If there are less than *count* bigwords between the
39053 cursor and the end of the edit buffer, *count* shall be adjusted to move the cursor to the last
39054 bigword in the edit buffer.

39055 If used as a motion command:

- 39056 1. If the associated command is *c*, *count* is 1, and the cursor is on a <blank> character, the
39057 region of text shall be the current character and no further action shall be taken.
- 39058 2. If there are less than *count* bigwords between the cursor and the end of the edit buffer, then
39059 the command shall succeed, and the region of text shall include the last character of the
39060 edit buffer.
- 39061 3. If there are <blank> characters or an end-of-line that precede the *count*th bigword, and the
39062 associated command is *c*, the region of text shall be up to and including the last character
39063 before the preceding <blank> characters or end-of-line.
- 39064 4. If there are <blank> characters or an end-of-line that precede the bigword, and the
39065 associated command is *d* or *y*, the region of text shall be up to and including the last
39066 <blank> character before the start of the bigword or end-of-line.
- 39067 5. Any text copied to a buffer shall be in character mode.

39068 If not used as a motion command:

- 39069 1. If the cursor is on the last character of the edit buffer, it shall be an error.

39070 *Current line:* Set to the line containing the current column.39071 *Current column:* Set to the last column in which any part of the first character of the *count*th next
39072 bigword is displayed.39073 **Delete Character at Cursor**39074 *Synopsis:* [*buffer*][*count*] x39075 Delete the *count* characters at and after the current character into *buffer*, if specified, and into the
39076 unnamed buffer.39077 If the line is empty, it shall be an error. If there are less than *count* characters at and after the
39078 cursor on the current line, *count* shall be adjusted to the number of characters at and after the
39079 cursor.39080 *Current line:* Unchanged.39081 *Current column:* If the line is empty, set to column position 1. Otherwise, if there were *count* or
39082 less characters at and after the cursor on the current line, set to the last column that displays any
39083 part of the last character of the line. Otherwise, unchanged.

39084 **Delete Character Before Cursor**39085 *Synopsis:* `[buffer][count] X`39086 Delete the *count* characters before the current character into *buffer*, if specified, and into the
39087 unnamed buffer.39088 If there are no characters before the current character on the current line, it shall be an error. If
39089 there are less than *count* previous characters on the current line, *count* shall be adjusted to the
39090 number of previous characters on the line.39091 *Current line:* Unchanged.39092 *Current column:* Set to (*current column* – *the width of the deleted characters*).39093 **Yank**39094 *Synopsis:* `[buffer][count] y motion`39095 Copy (yank) the region of text into *buffer*, if specified, and into the unnamed buffer.

39096 If the motion command is the y command repeated:

- 39097 1. The buffer shall be in line mode.
- 39098 2. If there are less than *count* – 1 lines after the current line in the edit buffer, it shall be an
39099 error.
- 39100 3. The text region shall be from the current line up to and including the next *count* – 1 lines.

39101 Otherwise, the buffer text mode and text region shall be as specified by the motion command.

39102 *Current line:* If the motion was from the current cursor position toward the end of the edit
39103 buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region
39104 specified by the motion command.39105 *Current column:*

- 39106 1. If the motion was from the current cursor position toward the end of the edit buffer,
39107 unchanged.
- 39108 2. Otherwise, if the current line is empty, set to column position 1.
- 39109 3. Otherwise, set to the last column that displays any part of the first character in the file that
39110 is part of the text region specified by the motion command.

39111 **Yank Current Line**39112 *Synopsis:* `[buffer][count] Y`39113 This command shall be equivalent to the *vi* command:39114 `[buffer][count] y_`

39115 **Redraw Window**

39116 If in open mode, the **z** command shall have the Synopsis:

39117 *Synopsis:* [*count*] **z**

39118 If *count* is not specified, it shall default to the **window** edit option -1 . The **z** command shall be
 39119 equivalent to the *ex z* command, with a type character of = and a *count* of *count* -2 , except that
 39120 the current line and current column shall be set as follows, and the **window** edit option shall not
 39121 be affected. If the calculation for the *count* argument would result in a negative number, the
 39122 *count* argument to the *ex z* command shall be zero. A blank line shall be written after the last line
 39123 is written.

39124 *Current line:* Unchanged.

39125 *Current column:* Unchanged.

39126 If not in open mode, the **z** command shall have the following Synopsis:

39127 *Synopsis:* [*line*] **z** [*count*] *character*

39128 If *line* is not specified, it shall default to the current line. If *line* is specified, but is greater than the
 39129 number of lines in the edit buffer, it shall default to the number of lines in the edit buffer.

39130 If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in the
 39131 *ex window* command), and the screen shall be redrawn.

39132 *line* shall be placed as specified by the following characters:

39133 <newline>, <carriage-return>

39134 Place the beginning of the line on the first line of the display.

39135 . Place the beginning of the line in the center of the display. The middle line of the display
 39136 shall be calculated as described for the **M** command.

39137 – Place an unspecified portion of the line on the last line of the display.

39138 + If *line* was specified, equivalent to the <newline> case. If *line* was not specified, display a
 39139 screen where the first line of the display shall be (current last line) $+1$. If there are no lines
 39140 after the last line in the display, it shall be an error.

39141 ^ If *line* was specified, display a screen where the last line of the display shall contain an
 39142 unspecified portion of the first line of a display that had an unspecified portion of the
 39143 specified line on the last line of the display. If this calculation results in a line before the
 39144 beginning of the edit buffer, display the first screen of the edit buffer.

39145 Otherwise, display a screen where the last line of the display shall contain an unspecified
 39146 portion of (current first line -1). If this calculation results in a line before the beginning of
 39147 the edit buffer, it shall be an error.

39148 *Current line:* If *line* and the ' ^ ' character were specified:

39149 1. If the first screen was displayed as a result of the command attempting to display lines
 39150 before the beginning of the edit buffer: if the first screen was already displayed,
 39151 unchanged; otherwise, set to (current first line -1).

39152 2. Otherwise, set to the last line of the display.

39153 If *line* and the ' + ' character were specified, set to the first line of the display.

39154 Otherwise, if *line* was specified, set to *line*.

39155 Otherwise, unchanged.

39156 *Current column*: Set to non-`<blank>`.

39157 **Exit**

39158 *Synopsis*: ZZ

39159 This command shall be equivalent to the `ex xit` command with no addresses, trailing `!`, or file
39160 name (see the `ex xit` command).

39161 **Input Mode Commands in vi**

39162 In text input mode, the current line shall consist of zero or more of the following categories:

39163 1. Characters preceding the text input entry point

39164 Characters in this category shall not be modified during text input mode.

39165 2. **autoindent** characters

39166 **autoindent** characters shall be automatically inserted into each line that is created in text
39167 input mode, either as a result of entering a `<newline>` character or `<carriage-return>`
39168 character while in text input mode, or as an effect of the command itself; for example, `O` or
39169 `o` (see the `ex autoindent` command), as if entered by the user.

39170 It shall be possible to erase **autoindent** characters with the `<control>-D` command; it is
39171 unspecified whether they can be erased by `<control>-H`, `<control>-U`, and `<control>-W`
39172 characters. Erasing any **autoindent** character turns the glyph into erase-columns and
39173 deletes the character from the edit buffer, but does not change its representation on the
39174 screen.

39175 3. Text input characters

39176 Text input characters are the characters entered by the user. Erasing any text input
39177 character turns the glyph into erase-columns and deletes the character from the edit buffer,
39178 but does not change its representation on the screen.

39179 Each text input character entered by the user (that does not have a special meaning) shall
39180 be treated as follows:

39181 a. The text input character shall be appended to the last character in the edit buffer
39182 from the first, second, or third categories.

39183 b. If there are no erase-columns on the screen, the text input command was the **R**
39184 command, and characters in the fifth category from the original line follow the
39185 cursor, the next such character shall be deleted from the edit buffer. If the **slowopen**
39186 edit option is not set, the corresponding glyph on the screen shall become erase-
39187 columns.

39188 c. If there are erase-columns on the screen, as many columns as they occupy, or as are
39189 necessary, shall be overwritten to display the text input character. (If only part of a
39190 multi-column glyph is overwritten, the remainder shall be left on the screen, and
39191 continue to be treated as erase-columns; it is unspecified whether the remainder of
39192 the glyph is modified in any way.)

39193 d. If additional screen columns are needed to display the text input character:

39194 1. If the **slowopen** edit option is set, the text input characters shall be displayed
39195 on subsequent screen columns, overwriting any characters displayed in those
39196 columns.

39197 2. Otherwise, any characters currently displayed on or after the column on the
39198 screen where the text input character is to be displayed shall be pushed ahead
39199 the number of screen columns necessary to display the rest of the text input
39200 character.

39201 4. Erase-columns

39202 Erase-columns are not logically part of the edit buffer, appearing only on the screen, and
39203 may be overwritten on the screen by subsequent text input characters. When text input
39204 mode ends, all erase-columns shall no longer appear on the screen.

39205 Erase-columns are initially the region of text specified by the **c** command (see **Change** (on
39206 page 3221)) however, erasing **autoindent** or text input characters causes the glyphs of the
39207 erased characters to be treated as erase-columns.

39208 5. Characters following the text region for the **c** command, or the text input entry point for all
39209 other commands

39210 Characters in this category shall not be modified during text input mode, except as
39211 specified in category 3.b. for the **R** text input command, or as <blank> characters deleted
39212 when a <newline> character or <carriage-return> character is entered.

39213 It is unspecified whether it is an error to attempt to erase past the beginning of a line that was
39214 created by the entry of a <newline> or <carriage-return> character during text input mode. If it
39215 is not an error, the editor shall behave as if the erasing character was entered immediately after
39216 the last text input character entered on the previous line, and all of the characters on the current
39217 line shall be treated as erase-columns.

39218 When text input mode is entered, or after a text input mode character is entered (except as
39219 specified for the special characters below), the cursor shall be positioned as follows:

- 39220 1. On the first column that displays any part of the first erase-column, if one exists
- 39221 2. Otherwise, if the **slowopen** edit option is set, on the first screen column after the last
39222 character in the first, second, or third categories, if one exists
- 39223 3. Otherwise, the first column that displays any part of the first character in the fifth category,
39224 if one exists
- 39225 4. Otherwise, the screen column after the last character in the first, second, or third
39226 categories, if one exists
- 39227 5. Otherwise, on column position 1

39228 The characters that are updated on the screen during text input mode are unspecified, other than
39229 that the last text input character shall always be updated, and, if the **slowopen** edit option is not
39230 set, the current cursor character shall always be updated.

39231 The following specifications are for command characters entered during text input mode.

39232 **NUL**

39233 *Synopsis:* NUL

39234 If the first character of the text input is a NUL, the most recently input text shall be input as if
39235 entered by the user, and then text input mode shall be exited. The text shall be input literally;
39236 that is, characters are neither macro or abbreviation expanded, nor are any characters interpreted
39237 in any special manner. It is unspecified whether implementations shall support more than 256
39238 bytes of remembered input text.

39239 **<control>-D**39240 *Synopsis:* `<control>-D`39241 The `<control>-D` character shall have no special meaning when in text input mode for a line-
39242 oriented command (see **Command Descriptions in vi** (on page 3201)).

39243 This command need not be supported on block-mode terminals.

39244 If the cursor does not follow an **autoindent** character, or an **autoindent** character and a `'0'` or
39245 `'^'` character:39246 1. If the cursor is in column position 1, the `<control>-D` character shall be discarded and no
39247 further action taken.39248 2. Otherwise, the `<control>-D` character shall have no special meaning.39249 If the last input character was a `'0'`, the cursor shall be moved to column position 1.39250 Otherwise, if the last input character was a `'^'`, the cursor shall be moved to column position 1.
39251 In addition, the **autoindent** level for the next input line shall be derived from the same line from
39252 which the **autoindent** level for the current input line was derived.39253 Otherwise, the cursor shall be moved back to the column after the previous shiftwidth (see the
39254 *ex* **shiftwidth** command) boundary.39255 All of the glyphs on columns between the starting cursor position and (inclusively) the ending
39256 cursor position shall become erase-columns as described in **Input Mode Commands in vi** (on
39257 page 3235).39258 *Current line:* Unchanged.39259 *Current column:* Set to 1 if the `<control>-D` was preceded by a `'^'` or `'0'`; otherwise, set to
39260 $(\text{column} - 1) - ((\text{column} - 2) \% \text{shiftwidth})$.39261 **<control>-H**39262 *Synopsis:* `<control>-H`39263 If in text input mode for a line-oriented command, and there are no characters to erase, text
39264 input mode shall be terminated, no further action shall be done for this command, and the
39265 current line and column shall be unchanged.39266 If there are characters other than **autoindent** characters that have been input on the current line
39267 before the cursor, the cursor shall move back one character.39268 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is
39269 implementation-defined whether the `<control>-H` command is an error or if the cursor moves
39270 back one **autoindent** character.39271 Otherwise, if the cursor is in column position 1 and there are previous lines that have been input,
39272 it is implementation-defined whether the `<control>-H` command is an error or if it is equivalent
39273 to entering `<control>-H` after the last input character on the previous input line.

39274 Otherwise, it shall be an error.

39275 All of the glyphs on columns between the starting cursor position and (inclusively) the ending
39276 cursor position shall become erase-columns as described in **Input Mode Commands in vi** (on
39277 page 3235).39278 The current erase character (see *stty*) shall cause an equivalent action to the `<control>-H`
39279 command, unless the previously inserted character was a backslash, in which case it shall be as

39280 if the literal current erase character had been inserted instead of the backslash.

39281 *Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to
39282 line -1.

39283 *Current column:* Set to the first column that displays any portion of the character backed up
39284 over.

39285 **<newline>**

39286 *Synopsis:* <newline>
39287 <carriage-return>
39288 <control>-J
39289 <control>-M

39290 If input was part of a line-oriented command, text input mode shall be terminated and the
39291 command shall continue execution with the input provided.

39292 Otherwise, terminate the current line. If there are no characters other than **autoindent** characters
39293 on the line, all characters on the line shall be discarded. Otherwise, it is unspecified whether the
39294 **autoindent** characters in the line are modified by entering these characters.

39295 Continue text input mode on a new line appended after the current line. If the **slowopen** edit
39296 option is set, the lines on the screen below the current line shall not be pushed down, but the
39297 first of them shall be cleared and shall appear to be overwritten. Otherwise, the lines of the
39298 screen below the current line shall be pushed down.

39299 If the **autoindent** edit option is set, an appropriate number of **autoindent** characters shall be
39300 added as a prefix to the line as described by the *ex autoindent* edit option.

39301 All columns after the cursor that are erase-columns (as described in **Input Mode Commands in**
39302 **vi** (on page 3235)) shall be discarded.

39303 If the **autoindent** edit option is set, all <blank> characters immediately following the cursor shall
39304 be discarded.

39305 All remaining characters after the cursor shall be transferred to the new line, positioned after any
39306 **autoindent** characters.

39307 *Current line:* Set to current line +1.

39308 *Current column:* Set to the first column that displays any portion of the first character after the
39309 **autoindent** characters on the new line, if any, or the first column position after the last
39310 **autoindent** character, if any, or column position 1.

39311 **<control>-T**

39312 *Synopsis:* <control>-T

39313 The <control>-T character shall have no special meaning when in text input mode for a line-
39314 oriented command (see **Command Descriptions in vi** (on page 3201)).

39315 This command need not be supported on block-mode terminals.

39316 Behave as if the user entered the minimum number of <blank> characters necessary to move the
39317 cursor forward to the column position after the next **shiftwidth** (see the *ex shiftwidth*
39318 command) boundary.

39319 *Current line:* Unchanged.

39320 *Current column*: Set to $column + \mathbf{shiftwidth} - ((column - 1) \% \mathbf{shiftwidth})$.

39321 **<control>-U**

39322 *Synopsis*: `<control>-U`

39323 If there are characters other than **autoindent** characters that have been input on the current line
39324 before the cursor, the cursor shall move to the first character input after the **autoindent**
39325 characters.

39326 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is
39327 implementation-defined whether the `<control>-U` command is an error or if the cursor moves to
39328 the first column position on the line.

39329 Otherwise, if the cursor is in column position 1 and there are previous lines that have been input,
39330 it is implementation-defined whether the `<control>-U` command is an error or if it is equivalent
39331 to entering `<control>-U` after the last input character on the previous input line.

39332 Otherwise, it shall be an error.

39333 All of the glyphs on columns between the starting cursor position and (inclusively) the ending
39334 cursor position shall become erase-columns as described in **Input Mode Commands in vi** (on
39335 page 3235).

39336 The current *kill* character (see *stty*) shall cause an equivalent action to the `<control>-U`
39337 command, unless the previously inserted character was a backslash, in which case it shall be as
39338 if the literal current *kill* character had been inserted instead of the backslash.

39339 *Current line*: Unchanged, unless previously input lines are erased, in which case it shall be set to
39340 line -1 .

39341 *Current column*: Set to the first column that displays any portion of the last character backed up
39342 over.

39343 **<control>-V**

39344 *Synopsis*: `<control>-V`

39345 `<control>-Q`

39346 Allow the entry of any subsequent character, other than `<control>-J` or the `<newline>` character,
39347 as a literal character, removing any special meaning that it may have to the editor in text input
39348 mode. If a `<control>-V` or `<control>-Q` is entered before a `<control>-J` or `<newline>` character,
39349 the `<control>-V` or `<control>-Q` character shall be discarded, and the `<control>-J` or `<newline>`
39350 shall behave as described in the `<newline>` command character during input mode.

39351 For purposes of the display only, the editor shall behave as if a `'^'` character was entered, and
39352 the cursor shall be positioned as if overwriting the `'^'` character. When a subsequent character
39353 is entered, the editor shall behave as if that character was entered instead of the original
39354 `<control>-V` or `<control>-Q` character.

39355 *Current line*: Unchanged.

39356 *Current column*: Unchanged.

- 39357 **<control>-W**
- 39358 *Synopsis:* `<control>-W`
- 39359 If there are characters other than **autoindent** characters that have been input on the current line
39360 before the cursor, the cursor shall move back over the last word preceding the cursor (including
39361 any `<blank>` characters between the end of the last word and the current cursor); the cursor shall
39362 not move to before the first character after the end of any **autoindent** characters.
- 39363 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is
39364 implementation-defined whether the `<control>-W` command is an error or if the cursor moves to
39365 the first column position on the line.
- 39366 Otherwise, if the cursor is in column position 1 and there are previous lines that have been input,
39367 it is implementation-defined whether the `<control>-W` command is an error or if it is equivalent
39368 to entering `<control>-W` after the last input character on the previous input line.
- 39369 Otherwise, it shall be an error.
- 39370 All of the glyphs on columns between the starting cursor position and (inclusively) the ending
39371 cursor position shall become erase-columns as described in **Input Mode Commands in vi** (on
39372 page 3235).
- 39373 *Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to
39374 line `-1`.
- 39375 *Current column:* Set to the first column that displays any portion of the last character backed up
39376 over.
- 39377 **<ESC>**
- 39378 *Synopsis:* `<ESC>`
- 39379 If input was part of a line-oriented command:
- 39380 1. If *interrupt* was entered, text input mode shall be terminated and the editor shall return to
39381 command mode. The terminal shall be alerted.
- 39382 **Notes to Reviewers**
- 39383 *This section with side shading will not appear in the final copy. - Ed.*
- 39384 D3, XCU, ERN 274 says the character ESC is not an interrupt character, so why is point 1
39385 here? This will need to be revisited when .2b is approved. I believe this is covered in
39386 Rationale; see later.
- 39387 2. If `<ESC>` was entered, text input mode shall be terminated and the command shall
39388 continue execution with the input provided.
- 39389 Otherwise, terminate text input mode and return to command mode.
- 39390 Any **autoindent** characters entered on newly created lines that have no other characters shall be
39391 deleted.
- 39392 Any leading **autoindent** and `<blank>` characters on newly created lines shall be rewritten to be
39393 the minimum number of `<blank>` characters possible.
- 39394 The screen shall be redisplayed as necessary to match the contents of the edit buffer.
- 39395 *Current line:* Unchanged.

39396 *Current column:*

- 39397 1. If there are text input characters on the current line, the column shall be set to the last
39398 column where any portion of the last text input character is displayed.
- 39399 2. Otherwise, if a character is displayed in the current column, unchanged.
- 39400 3. Otherwise, set to column position 1.

39401 **EXIT STATUS**

39402 The following exit values shall be returned:

- 39403 0 Successful completion.
- 39404 >0 An error occurred.

39405 **CONSEQUENCES OF ERRORS**

39406 When any error is encountered and the standard input is not a terminal device file, *vi* shall not
39407 write the file or return to command or text input mode, and shall terminate with a non-zero exit
39408 status.

39409 Otherwise, when an unrecoverable error is encountered it shall be equivalent to a SIGHUP
39410 asynchronous event.

39411 Otherwise, when an error is encountered, the editor shall behave as specified in **Command**
39412 **Descriptions in vi** (on page 3201).

39413 **APPLICATION USAGE**

39414 None.

39415 **EXAMPLES**

39416 None.

39417 **RATIONALE**

39418 See the RATIONALE for *ex* for more information on *vi*. Major portions of the *vi* utility
39419 specification point to *ex* to avoid inadvertent divergence. While *ex* and *vi* have historically been
39420 implemented as a single utility, this is not required by IEEE Std. 1003.1-200x.

39421 It is recognized that portions of *vi* would be difficult, if not impossible, to implement
39422 satisfactorily on a block-mode terminal, or a terminal without any form of cursor addressing,
39423 thus it is not a mandatory requirement that such features should work on all terminals. It is the
39424 intention, however, that a *vi* implementation should provide the full set of capabilities on all
39425 terminals capable of supporting them.

39426 Historically, *vi* exited immediately if the standard input was not a terminal.
39427 IEEE Std. 1003.1-200x permits, but does not require, this behavior. An end-of-file condition is not
39428 equivalent to an end-of-file character. A common end-of-file character, <control>-D, is
39429 historically a *vi* command.

39430 The text in the STANDARD OUTPUT section reflects the usage of the verb *display* in this section;
39431 some implementations of *vi* use standard output to write to the terminal, but
39432 IEEE Std. 1003.1-200x does not require that to be the case.

39433 Historically, implementations reverted to open mode if the terminal was incapable of
39434 supporting full visual mode. IEEE Std. 1003.1-200x requires this behavior. Historically, the open
39435 mode of *vi* behaved roughly equivalently to the visual mode, with the exception that only a
39436 single physical line from the edit buffer was kept current at any time. This line was normally
39437 displayed on the next-to-last line of a terminal with cursor addressing (and the last line
39438 performed its normal visual functions for line-oriented commands and messages). In addition,
39439 some few commands behaved differently in open mode than in visual mode.

39440 IEEE Std. 1003.1-200x requires conformance to historical practice.

39441 Historically, *ex* and *vi* implementations have expected text to proceed in the usual
39442 European/Latin order of left to right, top to bottom. There is no requirement in
39443 IEEE Std. 1003.1-200x that this be the case. The specification was deliberately written using
39444 words like “before”, “after”, “first”, and “last” in order to permit implementations to support
39445 the natural text order of the language.

39446 Historically, lines past the end of the edit buffer were marked with single tilde (‘~’) characters;
39447 that is, if the one-based display was 20 lines in length, and the last line of the file was on line one,
39448 then lines 2-20 would contain only a single ‘~’ character.

39449 Historically, the *vi* editor attempted to display only complete lines at the bottom of the screen (it
39450 did display partial lines at the top of the screen). If a line was too long to fit in its entirety at the
39451 bottom of the screen, the screen lines where the line would have been displayed were displayed
39452 as single ‘@’ characters, instead of displaying part of the line. IEEE Std. 1003.1-200x permits, but
39453 does not require, this behavior. Implementations are encouraged to attempt always to display a
39454 complete line at the bottom of the screen when doing scrolling or screen positioning by physical
39455 lines.

39456 Historically, lines marked with ‘@’ were also used to minimize output to dumb terminals over
39457 slow lines; that is, changes local to the cursor were updated, but changes to lines on the screen
39458 that were not close to the cursor were simply marked with an ‘@’ sign instead of being updated
39459 to match the current text. IEEE Std. 1003.1-200x permits, but does not require this feature
39460 because it is used ever less frequently as terminals become smarter and connections are faster.

39461 **Initialization in *ex* and *vi***

39462 Historically, *vi* always had a line in the edit buffer, even if the edit buffer was “empty”. For
39463 example:

- 39464 1. The *ex* command = executed from visual mode wrote “1” when the buffer was empty.
- 39465 2. Writes from visual mode of an empty edit buffer wrote files of a single character (a
39466 <newline> character), while writes from *ex* mode of an empty edit buffer wrote empty
39467 files.
- 39468 3. Put and read commands into an empty edit buffer left an empty line at the top of the edit
39469 buffer.

39470 For consistency, IEEE Std. 1003.1-200x does not permit any of these behaviors.

39471 Historically, *vi* did not always return the terminal to its original modes; for example, ICRNL was
39472 modified if it was not originally set. IEEE Std. 1003.1-200x does not permit this behavior.

39473 **Command Descriptions in *vi***

39474 Motion commands are among the most complicated aspects of *vi* to describe. With some
39475 exceptions, the text region and buffer type effect of a motion command on a *vi* command are
39476 described on a case-by-case basis. The descriptions of text regions in IEEE Std. 1003.1-200x are
39477 not intended to imply direction; that is, an inclusive region from line *n* to line *n*+5 is identical to
39478 a region from line *n*+5 to line *n*. This is of more than academic interest—movements to marks
39479 can be in either direction, and, if the **wrapsan** option is set, so can movements to search points.
39480 Historically, lines are always stored into buffers in text order; that is, from the start of the edit
39481 buffer to the end. IEEE Std. 1003.1-200x requires conformance to historical practice.

39482 Historically, command counts were applied to any associated motion, and were multiplicative
39483 to any supplied motion count. For example, **2cw** is the same as **c2w**, and **2c3w** is the same as

39484 **c6w**. IEEE Std. 1003.1-200x requires this behavior. Historically, *vi* commands that used
39485 bigwords, words, paragraphs, and sentences as objects treated groups of empty lines, or lines
39486 that contained only <blank> characters, inconsistently. Some commands treated them as a single
39487 entity, while others treated each line separately. For example, the **w**, **W**, and **B** commands treated
39488 groups of empty lines as individual words; that is, the command would move the cursor to each
39489 new empty line. The **e** and **E** commands treated groups of empty lines as a single word; that is,
39490 the first use would move past the group of lines. The **b** command would just beep at the user, or
39491 if done from the start of the line as a motion command, fail in unexpected ways. If the lines
39492 contained only (or ended with) <blank> characters, the **w** and **W** commands would just beep at
39493 the user, the **E** and **e** commands would treat the group as a single word, and the **B** and **b**
39494 commands would treat the lines as individual words. For consistency and simplicity of
39495 specification, IEEE Std. 1003.1-200x requires that all *vi* commands treat groups of empty or
39496 <blank> character-filled lines as a single entity, and that movement through lines ending with
39497 <blank> characters be consistent with other movements.

39498 Historically, *vi* documentation indicated that any number of double quotes were skipped after
39499 punctuation marks at sentence boundaries; however, implementations only skipped single
39500 quotes. IEEE Std. 1003.1-200x requires both to be skipped.

39501 Historically, the first and last characters in the edit buffer were word boundaries. This historical
39502 practice is required by IEEE Std. 1003.1-200x.

39503 Historically, *vi* attempted to update the minimum number of columns on the screen possible,
39504 which could lead to misleading information being displayed. IEEE Std. 1003.1-200x makes no
39505 requirements other than that the current character being entered is displayed correctly, leaving
39506 all other decisions in this area up to the implementation.

39507 Historically, lines were arbitrarily folded between columns of any characters that required
39508 multiple column positions on the screen, with the exception of tabs, which terminated at the
39509 right-hand margin. IEEE Std. 1003.1-200x permits the former and requires the latter.
39510 Implementations that do not arbitrarily break lines between columns of characters that occupy
39511 multiple column positions should not permit the cursor to rest on a column that does not
39512 contain any part of a character.

39513 The historical *vi* had a problem in that all movements were by physical lines, not by logical, or
39514 screen, lines. This is often the right thing to do; for example, single line movements, such as **j** or
39515 **k**, should work on physical lines. Commands like **dj**, or **j.**, where **.** is a change command, only
39516 make sense for physical lines. It is not, however, the right thing to do for screen motion or
39517 scrolling commands like <control>-D, <control>-F, and **H**. If the window is fairly small, using
39518 physical lines in these cases can result in completely random motion; for example, **1<control>-D**
39519 can result in a completely changed screen, without any overlap. This is clearly not what the user
39520 wanted. The problem is even worse in the case of the **H**, **L**, and **M** commands—as they position
39521 the cursor at the first non-<blank> character of the line, they may all refer to the same location in
39522 large lines, and will result in no movement at all.

39523 In addition, if the line is larger than the screen, using physical lines can make it impossible to
39524 display parts of the line—there are not any commands that do not display the beginning of the
39525 line in historical *vi*, and if both the beginning and end of the line cannot be on the screen at
39526 the same time, the user suffers. Finally, the page and half-page scrolling commands historically
39527 moved to the first non-<blank> character in the new line. If the line is approximately the same
39528 size as the screen, this is inadequate because the cursor before and after a <control>-D command
39529 will refer to the same location on the screen.

39530 Implementations of *ex* and *vi* exist that do not have these problems because the relevant
39531 commands (<control>-B, <control>-D, <control>-F, <control>-U, <control>-Y, <control>-E, **H**, **L**,
39532 and **M**) operate on logical screen lines, not physical edit buffer lines.

39533 IEEE Std. 1003.1-200x does not permit this behavior by default because the standard developers
39534 believed that users would find it too confusing. However, historical practice has been relaxed.
39535 For example, *ex* and *vi* historically attempted, albeit sometimes unsuccessfully, to never put part
39536 of a line on the last lines of a screen; for example, if a line would not fit in its entirety, no part of
39537 the line was displayed, and the screen lines corresponding to the line contained single '@'
39538 characters. This behavior is permitted, but not required by IEEE Std. 1003.1-200x, so that it is
39539 possible for implementations to support long lines in small screens more reasonably without
39540 changing the commands to be logically (instead of physically) oriented. IEEE Std. 1003.1-200x
39541 also permits implementations to refuse to edit any edit buffer containing a line that will not fit
39542 on the screen in its entirety.

39543 The display area (for example, the value of the **window** edit option) has historically been
39544 “grown”, or expanded, to display new text when local movements are done in displays where
39545 the number of lines displayed is less than the maximum possible. Expansion has historically
39546 been the first choice, when the target line is less than the maximum possible expansion value
39547 away. Scrolling has historically been the next choice, done when the target line is less than half a
39548 display away, and otherwise, the screen was redrawn. There were exceptions, however, in that
39549 *ex* commands generally always caused the screen to be redrawn. IEEE Std. 1003.1-200x does not
39550 specify a standard behavior because there may be external issues, such as connection speed, the
39551 number of characters necessary to redraw as opposed to scroll, or terminal capabilities that
39552 implementations will have to accommodate.

39553 The current line in IEEE Std. 1003.1-200x maps one-to-one to a physical line in the file. The
39554 current column does not. There are two different column values that are described by
39555 IEEE Std. 1003.1-200x. The first is the current column value as set by many of the *vi* commands.
39556 This value is remembered for the lifetime of the editor. The second column value is the actual
39557 position on the screen where the cursor rests. The two are not always the same. For example,
39558 when the cursor is backed by a multi-column character, the actual cursor position on the screen
39559 has historically been the last column of the character in command mode, and the first column of
39560 the character in input mode.

39561 Commands that set the current line, but that do not set the current cursor value (for example, **j**
39562 and **k**) attempt to get as close as possible to the remembered column position, so that the cursor
39563 tends to restrict itself to a vertical column as the user moves around in the edit buffer.
39564 IEEE Std. 1003.1-200x requires conformance to historical practice, requiring that the physical
39565 location of the cursor on the screen be adjusted from the current column value as necessary to
39566 support this historical behavior.

39567 Historically, only a single line (and for some terminals, a single line minus 1 column) of
39568 characters could be entered by the user for the line oriented commands; that is, **:**, **!**, **/**, or **?**.
39569 IEEE Std. 1003.1-200x permits, but does not require, this limitation.

39570 Historically, “soft” errors in *vi* caused the terminal to be alerted, but no error message was
39571 displayed. As a general rule, no error message was displayed for errors in command execution
39572 in *vi*, when the error resulted from the user attempting an invalid or impossible action, or when
39573 a searched-for object was not found. Examples of soft errors included **h** at the left margin,
39574 <control>-B or **[** at the beginning of the file, **2G** at the end of the file, and so on. In addition,
39575 errors such as **%**, **]**, **}**, **)**, **N**, **n**, **f**, **F**, **t**, and **T** failing to find the searched-for object were soft as well.
39576 Less consistently, **/** and **?** displayed an error message if the pattern was not found, **/**, **?**, **N**, and **n**
39577 displayed an error message if no previous regular expression had been specified, and **;** did not
39578 display an error message if no previous **f**, **F**, **t**, or **T** command had occurred. Also, behavior in
39579 this area might reasonably be based on a runtime evaluation of the speed of a network
39580 connection. Finally, some implementations have provided error messages for soft errors in
39581 order to assist naive users, based on the value of a verbose edit option. IEEE Std. 1003.1-200x
39582 does not list specific errors for which an error message shall be displayed. Implementations

39583 should conform to historical practice in the absence of any strong reason to diverge.

39584 **Page Backwards**

39585 The <control>-B and <control>-F commands historically considered it an error to attempt to
 39586 page past the beginning or end of the file, whereas the <control>-D and <control>-U commands
 39587 simply moved to the beginning or end of the file. For consistency, IEEE Std. 1003.1-200x requires
 39588 the latter behavior for all four commands. All four commands still consider it an error if the
 39589 current line is at the beginning (<control>-B, <control>-U) or end (<control>-F, <control>-D) of
 39590 the file. Historically, the <control>-B and <control>-F commands skip two lines in order to
 39591 include overlapping lines when a single command is entered. This makes less sense in the
 39592 presence of a *count*, as there will be, by definition, no overlapping lines. The actual calculation
 39593 used by historical implementations of the *vi* editor for <control>-B was:

39594 $((\text{current first line}) - \text{count} \times (\text{window edit option})) + 2$

39595 and for <control>-F was:

39596 $((\text{current first line}) + \text{count} \times (\text{window edit option})) - 2$

39597 This calculation does not work well when intermixing commands with and without counts; for
 39598 example, **3**<control>-F is not equivalent to entering the <control>-F command three times, and is
 39599 not reversible by entering the <control>-B command three times. For consistency with other *vi*
 39600 commands that take counts, IEEE Std. 1003.1-200x requires a different calculation.

39601 **Scroll Forward**

39602 The 4BSD and System V implementations of *vi* differed on the initial value used by the **scroll**
 39603 command. 4BSD used:

39604 $((\text{window edit option}) + 1) / 2$

39605 while System V used the value of the **scroll** edit option. The System V version is specified by
 39606 IEEE Std. 1003.1-200x because the standard developers believed that it was more intuitive and
 39607 permitted the user a method of setting the scroll value initially without also setting the number
 39608 of lines that are displayed.

39609 **Scroll Forward by Line**

39610 Historically, the <control>-E and <control>-Y commands considered it an error if the last and
 39611 first lines, respectively, were already on the screen. IEEE Std. 1003.1-200x requires conformance
 39612 to historical practice. Historically, the <control>-E and <control>-Y commands had no effect in
 39613 open mode. For simplicity and consistency of specification, IEEE Std. 1003.1-200x requires that
 39614 they behave as usual, albeit with a single line screen.

39615 **Clear and Redisplay**

39616 The historical <control>-L command refreshed the screen exactly as it was supposed to be
 39617 currently displayed, replacing any '@' characters for lines that had been deleted but not
 39618 updated on the screen with refreshed '@' characters. The intent of the <control>-L command is
 39619 to refresh when the screen has been accidentally overwritten; for example, by a **write** command
 39620 from another user, or modem noise.

39621 **Redraw Screen**

39622 The historical <control>-R command redisplayed only when necessary to update lines that had
 39623 been deleted but not updated on the screen and that were flagged with '@' characters. There is
 39624 no requirement that the screen be in any way refreshed if no lines of this form are currently
 39625 displayed. IEEE Std. 1003.1-200x permits implementations to extend this command to refresh
 39626 lines on the screen flagged with '@' characters because they are too long to be displayed in the
 39627 current framework; however, the current line and column need not be modified.

39628 **Search for tagstring**

39629 Historically, the first non-<blank> character at or after the cursor was the first character, and all
 39630 subsequent characters that were word characters, up to the end of the line, were included. For
 39631 example, with the cursor on the leading space or on the '#' character in the text "#bar@", the
 39632 tag was "#bar". On the character 'b' it was "bar", and on the 'a', it was "ar".
 39633 IEEE Std. 1003.1-200x requires this behavior.

39634 **Replace Text with Results from Shell Command**

39635 Historically, the <, >, and ! commands considered most cursor motions other than line-oriented
 39636 motions an error; for example, the command >/foo<CR> succeeded, while the command >|
 39637 failed, even though the text region described by the two commands might be identical. For
 39638 consistency, all three commands only consider entire lines and not partial lines, and the region is
 39639 defined as any line that contains a character that was specified by the motion.

39640 **Move to Matching Character**

39641 Other matching characters have been left implementation-defined in order to allow extensions
 39642 such as matching '<' and '>' for searching HTML, or #ifdef, #else, and #endif for searching C
 39643 source.

39644 **Repeat Substitution**

39645 IEEE Std. 1003.1-200x requires that any c and g flags specified to the previous substitute
 39646 command be ignored; however, the r flag may still apply, if supported by the implementation.

39647 **Return to Previous (Context or Section)**

39648 The [[,]], (,), {, and } commands are all affected by "section boundaries", but in some historical
 39649 implementations not all of the commands recognize the same section boundaries. This is a bug,
 39650 not a feature, and a unique section-boundary algorithm was not described for each command.
 39651 One special case that is preserved is that the sentence command moves to the end of the last line
 39652 of the edit buffer while the other commands go to the beginning, in order to preserve the
 39653 traditional character cut semantics of the sentence command. Historically, vi section boundaries
 39654 at the beginning and end of the edit buffer were the first non-<blank> character on the first and
 39655 last lines of the edit buffer if one exists; otherwise, the last character of the first and last lines of
 39656 the edit buffer if one exists. To increase consistency with other section locations, this has been
 39657 simplified by IEEE Std. 1003.1-200x to the first character of the first and last lines of the edit
 39658 buffer, or the first and the last lines of the edit buffer if they are empty.

39659 Sentence boundaries were problematic in the historical vi. They were not only the boundaries as
 39660 defined for the section and paragraph commands, but they were the first non-<blank> character
 39661 that occurred after those boundaries, as well. Historically, the vi section commands were
 39662 documented as taking an optional window size as a count preceding the command. This was not
 39663 implemented in historical versions, so IEEE Std. 1003.1-200x requires that the count repeat the
 39664 command, for consistency with other vi commands.

39665 **Repeat**

39666 Historically, mapped commands other than text input commands could not be repeated using
39667 the **period** command. IEEE Std. 1003.1-200x requires conformance to historical practice.

39668 The restrictions on the interpretation of special characters (for example, <control>-H) in the
39669 repetition of text input mode commands is intended to match historical practice. For example,
39670 given the input sequence:

```
39671 iab<control>-H<control>-H<control>-Hdef<escape>
```

39672 the user should be informed of an error when the sequence is first entered, but not during a
39673 command repetition. The character <control>-T is specifically exempted from this restriction.
39674 Historical implementations of *vi* ignored <control>-T characters that were input in the original
39675 command during command repetition. IEEE Std. 1003.1-200x prohibits this behavior.

39676 **Find Regular Expression**

39677 Historically, commands did not affect the line searched to or from if the motion command was a
39678 search (*/*, *?*, **N**, **n**) and the final position was the start/end of the line. There were some special
39679 cases and *vi* was not consistent. IEEE Std. 1003.1-200x does not permit this behavior, for
39680 consistency. Historical implementations permitted, but were unable to handle searches as
39681 motion commands that wrapped (that is, due to the edit option **wrapsan**) to the original
39682 location. IEEE Std. 1003.1-200x requires that this behavior be treated as an error.

39683 Historically, the syntax `"/RE/0"` was used to force the command to cut text in line mode.
39684 IEEE Std. 1003.1-200x requires conformance to historical practice.

39685 Historically, in open mode, a **z** specified to a search command redisplayed the current line
39686 instead of displaying the current screen with the current line highlighted. For consistency and
39687 simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39688 Historically, trailing **z** commands were permitted and ignored if entered as part of a search used
39689 as a motion command. For consistency and simplicity of specification, IEEE Std. 1003.1-200x
39690 does not permit this behavior.

39691 **Execute an ex Command**

39692 Historically, *vi* implementations restricted the commands that could be entered on the colon
39693 command line (for example, **append** and **change**), and some other commands were known to
39694 cause them to fail catastrophically. For consistency, IEEE Std. 1003.1-200x does not permit these
39695 restrictions. When executing an *ex* command by entering `:`, it is not possible to enter a <newline>
39696 character as part of the command because it is considered the end of the command. A different
39697 approach is to enter *ex* command mode by using the *vi* **Q** command (and later resuming visual
39698 mode with the *ex* **vi** command). In *ex* command mode, the single-line limitation does not exist.
39699 So, for example, the following is valid:

```
39700 Q
39701 s/break here/break\
39702 here/
39703 vi
```

39704 IEEE Std. 1003.1-200x requires that, if the *ex* command overwrites any part of the screen that
39705 would be erased by a refresh, *vi* pauses for a character from the user. Historically, this character
39706 could be any character; for example, a character input by the user before the message appeared,
39707 or even a mapped character. This is probably a bug, but implementations that have tried to be
39708 more rigorous by requiring that the user enter a specific character, or that the user enter a
39709 character after the message was displayed, have been forced by user indignation back into

39710 historical behavior. IEEE Std. 1003.1-200x requires conformance to historical practice.

39711 **Shift Left (Right)**

39712 Refer to the Rationale for the ! and / commands. Historically, the < and > commands sometimes
39713 moved the cursor to the first non-<blank> character (for example if the command was repeated
39714 or with _ as the motion command), and sometimes left it unchanged. IEEE Std. 1003.1-200x does
39715 not permit this inconsistency, requiring instead that the cursor always move to the first non-
39716 <blank> character. Historically, the < and > commands did not support buffer arguments,
39717 although some implementations allow the specification of an optional buffer. This behavior is
39718 neither required nor disallowed by IEEE Std. 1003.1-200x.

39719 **Execute**

39720 Historically, buffers could execute other buffers, and loops, infinite and otherwise, were
39721 possible. IEEE Std. 1003.1-200x requires conformance to historical practice. The **buffer* syntax of
39722 *ex* is not required in *vi*, because it is not historical practice and has been used in some *vi*
39723 implementations to support additional scripting languages.

39724 **Reverse Case**

39725 Historically, the ~ command ignored any associated *count*, and acted only on the characters in
39726 the current line. For consistency with other *vi* commands, IEEE Std. 1003.1-200x requires that an
39727 associated *count* act on the next *count* characters, and that the command move to subsequent
39728 lines if warranted by *count*, to make it possible to modify large pieces of text in a reasonably
39729 efficient manner. There exist *vi* implementations that optionally require an associated motion
39730 command for the ~ command. Implementations supporting this functionality are encouraged to
39731 base it on the **tildedop** edit option and handle the text regions and cursor positioning identically
39732 to the **yank** command.

39733 **Append**

39734 Historically, *counts* specified to the **A**, **a**, **I**, and **i** commands repeated the input of the first line
39735 *count* times, and did not repeat the subsequent lines of the input text. IEEE Std. 1003.1-200x
39736 requires that the entire text input be repeated *count* times.

39737 **Move Backward to Preceding Word**

39738 Historically, *vi* became confused if word commands were used as motion commands in empty
39739 files. IEEE Std. 1003.1-200x requires that this be an error. Historical implementations of *vi* had a
39740 large number of bugs in the word movement commands, and they varied greatly in behavior in
39741 the presence of empty lines, “words” made up of a single character, and lines containing only
39742 <blank> characters. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does
39743 not permit this behavior.

39744 **Change to End-of-Line**

39745 Some historical implementations of the **C** command did not behave as described by
39746 IEEE Std. 1003.1-200x when the **\$** key was remapped because they were implemented by
39747 pushing the **\$** key onto the input queue and reprocessing it. IEEE Std. 1003.1-200x does not
39748 permit this behavior. Historically, the **C**, **S**, and **s** commands did not copy replaced text into the
39749 numeric buffers. For consistency and simplicity of specification, IEEE Std. 1003.1-200x requires
39750 that they behave like their respective **c** commands in all respects.

39751 Delete

39752 Historically, lines in open mode that were deleted were scrolled up, and an @ glyph written over
39753 the beginning of the line. In the case of terminals that are incapable of the necessary cursor
39754 motions, the editor erased the deleted line from the screen. IEEE Std. 1003.1-200x requires
39755 conformance to historical practice; that is, if the terminal cannot display the '@' character, the
39756 line cannot remain on the screen.

39757 Delete to End-of-Line

39758 Some historical implementations of the **D** command did not behave as described by
39759 IEEE Std. 1003.1-200x when the **\$** key was remapped because they were implemented by
39760 pushing the **\$** key onto the input queue and reprocessing it. IEEE Std. 1003.1-200x does not
39761 permit this behavior.

39762 Join

39763 An historical oddity of *vi* is that the commands **J**, **1J**, and **2J** are all equivalent.
39764 IEEE Std. 1003.1-200x requires conformance to historical practice. The *vi* **J** command is specified
39765 in terms of the *ex* **join** command with an *ex* command *count* value. The address correction for a
39766 *count* that is past the end of the edit buffer is necessary for historical compatibility for both *ex*
39767 and *vi*.

39768 Mark Position

39769 Historical practice is that only lowercase letters, plus '' and ''', could be used to mark a
39770 cursor position. IEEE Std. 1003.1-200x requires conformance to historical practice, but
39771 encourages implementations to support other characters as marks as well.

39772 Repeat Regular Expression Find (Forward and Reverse)

39773 Historically, the **N** and **n** commands could not be used as motion components for the **c**
39774 command. With the exception of the **cN** command, which worked if the search crossed a line
39775 boundary, the text region would be discarded, and the user would not be in text input mode. For
39776 consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39777 Insert Empty Line (Below and Above)

39778 Historically, counts to the **O** and **o** commands were used as the number of physical lines to
39779 open, if the terminal was dumb and the **slowopen** option was not set. This was intended to
39780 minimize traffic over slow connections and repainting for dumb terminals. IEEE Std. 1003.1-200x
39781 does not permit this behavior, requiring that a *count* to the open command behave as for other
39782 text input commands. This change to historical practice was made for consistency, and because a
39783 superset of the functionality is provided by the **slowopen** edit option.

39784 Put from Buffer (Following and Before)

39785 Historically, *counts* to the **p** and **P** commands were ignored if the buffer was a line mode buffer,
39786 but were (mostly) implemented as described in IEEE Std. 1003.1-200x if the buffer was a
39787 character mode buffer. Because implementations exist that do not have this limitation, and
39788 because pasting lines multiple times is generally useful, IEEE Std. 1003.1-200x requires that *count*
39789 be supported for all **p** and **P** commands.

39790 Historical implementations of *vi* were widely known to have major problems in the **p** and **P**
39791 commands, particularly when unusual regions of text were copied into the edit buffer. The
39792 standard developers viewed these as bugs, and they are not permitted for consistency and

39793 simplicity of specification.

39794 Historically, a **P** or **p** command (or an *ex put* command executed from open or visual mode)
39795 executed in an empty file, left an empty line as the first line of the file. For consistency and
39796 simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39797 **Replace Character**

39798 Historically, the **r** command did not correctly handle the *erase* and *word erase* characters as
39799 arguments, nor did it handle an associated *count* greater than 1 with a <carriage-return>
39800 character argument, for which it replaced *count* characters with a single <newline> character.
39801 IEEE Std. 1003.1-200x does not permit these inconsistencies.

39802 Historically, the **r** command permitted the <control>-V escaping of entered characters, such as
39803 <ESC> and the <carriage-return> character; however, it required two leading <control>-V
39804 characters instead of one. IEEE Std. 1003.1-200x requires that this be changed for consistency
39805 with the other text input commands of *vi*.

39806 Historically, it is an error to enter the **r** command if there are less than *count* characters at or after
39807 the cursor in the line. While a reasonable and unambiguous extension would be to permit the **r**
39808 command on empty lines, it would require that too large a *count* be adjusted to match the
39809 number of characters at or after the cursor for consistency, which is sufficiently different from
39810 historical practice to be avoided. IEEE Std. 1003.1-200x requires conformance to historical
39811 practice.

39812 **Replace Characters**

39813 Historically, if there were **autoindent** characters in the line on which the **R** command was run,
39814 and **autoindent** was set, the first <newline> character would be properly indented and no
39815 characters would be replaced by the <newline> character. Each additional <newline> character
39816 would replace *n* characters, where *n* was the number of characters that were needed to indent
39817 the rest of the line to the proper indentation level. This behavior is a bug and is not permitted by
39818 IEEE Std. 1003.1-200x.

39819 **Undo**

39820 Historical practice for cursor positioning after undoing commands was mixed. In most cases,
39821 when undoing commands that affected a single line, the cursor was moved to the start of added
39822 or changed text, or immediately after deleted text. However, if the user had moved from the line
39823 being changed, the column was either set to the first non-<blank> character, returned to the
39824 origin of the command, or remained unchanged. When undoing commands that affected
39825 multiple lines or entire lines, the cursor was moved to the first character in the first line restored.
39826 As an example of how inconsistent this was, a search, followed by an **o** text input command,
39827 followed by an **undo** would return the cursor to the location where the **o** command was entered,
39828 but a **cw** command followed by an **o** command followed by an **undo** would return the cursor to
39829 the first non-<blank> character of the line. IEEE Std. 1003.1-200x requires the most useful of
39830 these behaviors, and discards the least useful, in the interest of consistency and simplicity of
39831 specification.

39832

Yank

39833

39834

39835

39836

39837

39838

39839

39840

39841

39842

Historically, the **yank** command did not move to the end of the motion if the motion was in the forward direction. It moved to the end of the motion if the motion was in the backward direction, except for the **_** command, or for the **G** and **'** commands when the end of the motion was on the current line. This was further complicated by the fact that for a number of motion commands, the **yank** command moved the cursor but did not update the screen; for example, a subsequent command would move the cursor from the end of the motion, even though the cursor on the screen had not reflected the cursor movement for the **yank** command. IEEE Std. 1003.1-200x requires that all **yank** commands associated with backward motions move the cursor to the end of the motion for consistency, and specifically, to make **'** commands as motions consistent with search patterns as motions.

39843

Yank Current Line

39844

39845

39846

39847

Some historical implementations of the **Y** command did not behave as described by IEEE Std. 1003.1-200x when the **'_'** key was remapped because they were implemented by pushing the **'_'** key onto the input queue and reprocessing it. IEEE Std. 1003.1-200x does not permit this behavior.

39848

Redraw Window

39849

39850

39851

39852

39853

39854

Historically, the **z** command always redrew the screen. This is permitted but not required by IEEE Std. 1003.1-200x, because of the frequent use of the **z** command in macros such as **map n nz** for screen positioning, instead of its use to change the screen size. The standard developers believed that expanding or scrolling the screen offered a better interface for users. The ability to redraw the screen is preserved if the optional new window size is specified, and in the **<control>-L** and **<control>-R** commands.

39855

39856

39857

The semantics of **z^** are confusing at best. Historical practice is that the screen before the screen that ended with the specified line is displayed. IEEE Std. 1003.1-200x requires conformance to historical practice.

39858

39859

39860

39861

39862

Historically, the **z** command would not display a partial line at the top or bottom of the screen. If the partial line would normally have been displayed at the bottom of the screen, the command worked, but the partial line was replaced with **'@'** characters. If the partial line would normally have been displayed at the top of the screen, the command would fail. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39863

39864

Historically, the **z** command with a line specification of 1 ignored the command. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39865

39866

39867

Historically, the **z** command did not set the cursor column to the first non-**<blank>** character for the character if the first screen was to be displayed, and was already displayed. For consistency and simplicity of specification, IEEE Std. 1003.1-200x does not permit this behavior.

39868

Input Mode Commands in vi

39869

39870

39871

39872

39873

Historical implementations of **vi** did not permit the the user to erase more than a single line of input, or to use normal erase characters such as *line erase*, *worderase*, and *erase* to erase **autoindent** characters. As there exist implementations of **vi** that do not have these limitations, both behaviors are permitted, but only historical practice is required. In the case of these extensions, **vi** is required to pause at the **autoindent** and previous line boundaries.

39874

39875

Historical implementations of **vi** updated only the portion of the screen where the current cursor character was displayed. For example, consider the **vi** input keystrokes:

39876 iabcd<escape>0C<tab>

39877 Historically, the <tab> character would overwrite the characters "abcd" when it was displayed.
39878 Other implementations replace only the 'a' character with the <tab> character, and then push
39879 the rest of the characters ahead of the cursor. Both implementations have problems. The
39880 historical implementation is probably visually nicer for the above example; however, for the
39881 keystrokes:

39882 iabcd<ESC>0R<tab><ESC>

39883 the historical implementation results in the string "bcd" disappearing and then magically
39884 reappearing when the <ESC> character is entered. IEEE Std. 1003.1-200x requires the former
39885 behavior when overwriting erase-columns; that is, overwriting characters that are no longer
39886 logically part of the edit buffer, and the latter behavior otherwise.

39887 Historical implementations of *vi* discarded the <control>-D and <control>-T characters when
39888 they were entered at places where their command functionality was not appropriate.
39889 IEEE Std. 1003.1-200x requires that the <control>-T functionality always be available, and that
39890 <control>-D be treated as any other key when not operating on **autoindent** characters.

39891 NUL

39892 Some historical implementations of *vi* limited the number of characters entered using the NUL
39893 input character to 256 bytes. IEEE Std. 1003.1-200x permits this limitation; however,
39894 implementations are encouraged to remove this limit.

39895 <control>-D

39896 See also Rationale for the input mode command <newline>. The hidden assumptions in the
39897 <control>-D command (and in the *vi* **autoindent** specification in general) is that <space>
39898 characters take up a single column on the screen and that <tab> characters are comprised of an
39899 integral number of <space> characters.

39900 <newline>

39901 Implementations are permitted to rewrite **autoindent** characters in the line when <newline>,
39902 <carriage-return>, <control>-D, and <control>-T are entered, or when the **shift** commands are
39903 used, because historical implementations have both done so and found it necessary to do so. For
39904 example, a <control>-D when the cursor is preceded by a single <tab> character, with **tabstop**
39905 set to 8, and **shiftwidth** set to 3, will result in the <tab> character being replaced by several
39906 <space> characters.

39907 <control>-T

39908 See also the Rationale for the input mode command <newline>. Historically, <control>-T only
39909 worked if no non-<blank> characters had yet been input in the current input line. In addition,
39910 the characters inserted by <control>-T were treated as **autoindent** characters, and could not be
39911 erased using normal user erase characters. Because implementations exist that do not have
39912 these limitations, and as moving to a column boundary is generally useful, IEEE Std. 1003.1-200x
39913 requires that both limitations be removed.

- 39914 **<control>-V**
- 39915 Historically, *vi* used **^V**, regardless of the value of the literal-next character of the terminal.
39916 IEEE Std. 1003.1-200x requires conformance to historical practice.
- 39917 The uses described for **<control>-V** can also be accomplished with **<control>-Q**, which is useful
39918 on terminals that use **<control>-V** for the down-arrow function. However, most historical
39919 implementations use **<control>-Q** for the *termios* START character, so the editor will generally
39920 not receive the **<control>-Q** unless **stty ixon** mode is set to off. (In addition, some historical
39921 implementations of *vi* explicitly set **ixon** mode to on, so it was difficult for the user to set it to
39922 off.) Any of the command characters described in IEEE Std. 1003.1-200x can be made ineffective
39923 by their selection as *termios* control characters, using the *stty* utility or other methods described
39924 in the System Interfaces volume of IEEE Std. 1003.1-200x.
- 39925 **<ESC>**
- 39926 Historically, SIGINT alerted the terminal when used to end input mode. This behavior is
39927 permitted, but not required, by IEEE Std. 1003.1-200x.
- 39928 **FUTURE DIRECTIONS**
- 39929 None.
- 39930 **SEE ALSO**
- 39931 *ex*, *stty*
- 39932 **CHANGE HISTORY**
- 39933 First released in Issue 2.
- 39934 **Issue 4**
- 39935 Aligned with the ISO/IEC 9945-2:1993 standard.
- 39936 **Issue 5**
- 39937 FUTURE DIRECTIONS section added.
- 39938 **Issue 6**
- 39939 This utility is now marked as part of the User Portability Utilities option.
- 39940 The APPLICATION USAGE section is added.
- 39941 The obsolescent SYNOPSIS is removed.
- 39942 The following new requirements on POSIX implementations derive from alignment with the
39943 Single UNIX Specification:
- 39944
 - The **lisp** mode is added.
- 39945
 - The **reindent** command description is added.
- 39946 The *vi* utility has been extensively rewritten for alignment with the IEEE P1003.2b draft
39947 standard.

39948 **NAME**

39949 wait — await process completion

39950 **SYNOPSIS**39951 wait [*pid...*]39952 **DESCRIPTION**

39953 When an asynchronous list (see Section 2.9.3.1 (on page 2259)) is started by the shell, the process
 39954 ID of the last command in each element of the asynchronous list shall become known in the
 39955 current shell execution environment; see Section 2.13 (on page 2273).

39956 If the *wait* utility is invoked with no operands, it shall wait until all process IDs known to the
 39957 invoking shell have terminated and exit with a zero exit status.

39958 If one or more *pid* operands are specified that represent known process IDs, the *wait* utility shall
 39959 wait until all of them have terminated. If one or more *pid* operands are specified that represent
 39960 unknown process IDs, *wait* shall treat them as if they were known process IDs that exited with
 39961 exit status 127. The exit status returned by the *wait* utility shall be the exit status of the process
 39962 requested by the last *pid* operand.

39963 The known process IDs are applicable only for invocations of *wait* in the current shell execution
 39964 environment.

39965 **OPTIONS**

39966 None.

39967 **OPERANDS**

39968 The following operand shall be supported:

39969 *pid* One of the following:

39970 1. The unsigned decimal integer process ID of a command, for which the utility
 39971 is to wait for the termination.

39972 2. A job control job ID (see the Base Definitions volume of
 39973 IEEE Std. 1003.1-200x, Section 3.205, Job Control Job ID) that identifies a
 39974 background process group to be waited for. The job control job ID notation is
 39975 applicable only for invocations of *wait* in the current shell execution
 39976 environment; see Section 2.13 (on page 2273). The exit status of *wait* shall be
 39977 determined by the last command in the pipeline.

39978 **Note:** The job control job ID type of *pid* is only available on systems
 39979 supporting the User Portability Utilities option.

39980 **STDIN**

39981 Not used.

39982 **INPUT FILES**

39983 None.

39984 **ENVIRONMENT VARIABLES**39985 The following environment variables shall affect the execution of *wait*:

39986 *LANG* Provide a default value for the internationalization variables that are unset or null.
 39987 If *LANG* is unset or null, the corresponding value from the implementation-
 39988 defined default locale shall be used. If any of the internationalization variables
 39989 contains an invalid setting, the utility shall behave as if none of the variables had
 39990 been defined.

39991 *LC_ALL* If set to a non-empty string value, override the values of all the other
 39992 internationalization variables.

39993 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
 39994 characters (for example, single-byte as opposed to multi-byte characters in
 39995 arguments).

39996 *LC_MESSAGES*
 39997 Determine the locale that should be used to affect the format and contents of
 39998 diagnostic messages written to standard error.

39999 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

40000 **ASYNCHRONOUS EVENTS**

40001 Default.

40002 **STDOUT**

40003 Not used.

40004 **STDERR**

40005 Used only for diagnostic messages.

40006 **OUTPUT FILES**

40007 None.

40008 **EXTENDED DESCRIPTION**

40009 None.

40010 **EXIT STATUS**

40011 If one or more operands were specified, all of them have terminated or were not known by the
 40012 invoking shell, and the status of the last operand specified is known, then the exit status of *wait*
 40013 shall be the exit status information of the command indicated by the last operand specified. If
 40014 the process terminated abnormally due to the receipt of a signal, the exit status shall be greater
 40015 than 128 and shall be distinct from the exit status generated by other signals, but the exact value
 40016 is unspecified. (See the *kill -l* option.) Otherwise, the *wait* utility shall exit with one of the
 40017 following values:

40018 0 The *wait* utility was invoked with no operands and all process IDs known by the
 40019 invoking shell have terminated.

40020 1-126 The *wait* utility detected an error.

40021 127 The command identified by the last *pid* operand specified is unknown.

40022 **CONSEQUENCES OF ERRORS**

40023 Default.

40024 **APPLICATION USAGE**

40025 On most implementations, *wait* is a shell built-in. If it is called in a subshell or separate utility
 40026 execution environment, such as one of the following:

40027 (wait)
 40028 nohup wait ...
 40029 find . -exec wait ... \;

40030 it returns immediately because there are no known process IDs to wait for in those
 40031 environments.

40032 Historical implementations of interactive shells have discarded the exit status of terminated
 40033 background processes before each shell prompt. Therefore, the status of background processes
 40034 was usually lost unless it terminated while *wait* was waiting for it. This could be a serious

40035 problem when a job that was expected to run for a long time actually terminated quickly with a
 40036 syntax or initialization error because the exit status returned was usually zero if the requested
 40037 process ID was not found. This volume of IEEE Std. 1003.1-200x requires the implementation to
 40038 keep the status of terminated jobs available until the status is requested, so that scripts like:

```
40039     j1&
40040     p1=$!
40041     j2&
40042     wait $p1
40043     echo Job 1 exited with status $?
40044     wait $!
40045     echo Job 2 exited with status $?
```

40046 works without losing status on any of the jobs. The shell is allowed to discard the status of any
 40047 process that it determines the application cannot get the process ID from the shell. It is also
 40048 required to remember only {CHILD_MAX} number of processes in this way. Since the only way
 40049 to get the process ID from the shell is by using the '!' shell parameter, the shell is allowed to
 40050 discard the status of an asynchronous list if "\$!" was not referenced before another
 40051 asynchronous list was started. (This means that the shell only has to keep the status of the last
 40052 asynchronous list started if the application did not reference "\$!". If the implementation of the
 40053 shell is smart enough to determine that a reference to "\$!" was not saved anywhere that the
 40054 application can retrieve it later, it can use this information to trim the list of saved information.
 40055 Note also that a successful call to *wait* with no operands discards the exit status of all
 40056 asynchronous lists.)

40057 If the exit status of *wait* is greater than 128, there is no way for the application to know if the
 40058 waited-for process exited with that value or was killed by a signal. Since most utilities exit with
 40059 small values, there is seldom any ambiguity. Even in the ambiguous cases, most applications
 40060 just need to know that the asynchronous job failed; it does not matter whether it detected an
 40061 error and failed or was killed and did not complete its job normally.

40062 EXAMPLES

40063 Although the exact value used when a process is terminated by a signal is unspecified, if it is
 40064 known that a signal terminated a process, a script can still reliably figure out which signal using
 40065 *kill* as shown by the following script:

```
40066     sleep 1000&
40067     pid=$!
40068     kill -kill $pid
40069     wait $pid
40070     echo $pid was terminated by a SIG$(kill -l $?) signal.
```

40071 If the following sequence of commands is run in less than 31 seconds:

```
40072     sleep 257 | sleep 31 &
40073     jobs -l %%
```

40074 either of the following commands returns the exit status of the second *sleep* in the pipeline:

```
40075     wait <pid of sleep 31>
40076     wait %%
```

40077 RATIONALE

40078 The description of *wait* does not refer to the *waitpid()* function from the System Interfaces
 40079 volume of IEEE Std. 1003.1-200x because that would needlessly overspecify this interface.
 40080 However, the wording means that *wait* is required to wait for an explicit process when it is given
 40081 an argument so that the status information of other processes is not consumed. Historical

40082 implementations use the *wait()* function defined in the System Interfaces volume of
40083 IEEE Std. 1003.1-200x until *wait()* returns the requested process ID or finds that the requested
40084 process does not exist. Because this means that a shell script could not reliably get the status of
40085 all background children if a second background job was ever started before the first job finished,
40086 it is recommended that the *wait* utility use a method such as the functionality provided by the
40087 *waitpid()* function.

40088 The ability to wait for multiple *pid* operands was adopted from the KornShell.

40089 This new functionality was added because it is needed to determine the exit status of any
40090 asynchronous list accurately. The only compatibility problem that this change creates is for a
40091 script like

```
40092 while sleep 60 do  
40093     job& echo Job started $(date) as $! done
```

40094 which causes the shell to monitor all of the jobs started until the script terminates or runs out of
40095 memory. This would not be a problem if the loop did not reference "\$!" or if the script would
40096 occasionally *wait* for jobs it started.

40097 **FUTURE DIRECTIONS**

40098 None.

40099 **SEE ALSO**

40100 *sh*, the System Interfaces volume of IEEE Std. 1003.1-200x, *waitpid()*

40101 **CHANGE HISTORY**

40102 First released in Issue 2.

40103 **Issue 4**

40104 Aligned with the ISO/IEC 9945-2:1993 standard.

40105 **NAME**40106 `wc` — word, line, and byte or character count40107 **SYNOPSIS**40108 `wc [-c|-m][-lw][file...]`40109 **DESCRIPTION**40110 The `wc` utility shall read one or more input files and, by default, write the number of <newline>
40111 characters, words, and bytes contained in each input file to the standard output.40112 The utility also shall write a total count for all named files, if more than one input file is
40113 specified.40114 The `wc` utility shall consider a *word* to be a non-zero-length string of characters delimited by
40115 white space.40116 **OPTIONS**40117 The `wc` utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
40118 12.2, Utility Syntax Guidelines.

40119 The following options shall be supported:

40120 `-c` Write to the standard output the number of bytes in each input file.40121 `-l` Write to the standard output the number of <newline> characters in each input
40122 file.40123 `-m` Write to the standard output the number of characters in each input file.40124 `-w` Write to the standard output the number of words in each input file.40125 When any option is specified, `wc` shall report only the information requested by the specified
40126 options.40127 **OPERANDS**

40128 The following operand shall be supported:

40129 *file* A path name of an input file. If no *file* operands are specified, the standard input
40130 shall be used.40131 **STDIN**40132 The standard input shall be used only if no *file* operands are specified. See the INPUT FILES
40133 section.40134 **INPUT FILES**

40135 The input files may be of any type.

40136 **ENVIRONMENT VARIABLES**40137 The following environment variables shall affect the execution of `wc`:40138 *LANG* Provide a default value for the internationalization variables that are unset or null.
40139 If *LANG* is unset or null, the corresponding value from the implementation-
40140 defined default locale shall be used. If any of the internationalization variables
40141 contains an invalid setting, the utility shall behave as if none of the variables had
40142 been defined.40143 *LC_ALL* If set to a non-empty string value, override the values of all the other
40144 internationalization variables.40145 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
40146 characters (for example, single-byte as opposed to multi-byte characters in
40147 arguments and input files) and which characters are defined as white space

- 40148 characters.
- 40149 **LC_MESSAGES**
- 40150 Determine the locale that should be used to affect the format and contents of
- 40151 diagnostic messages written to standard error and informative messages written to
- 40152 standard output.
- 40153 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC_MESSAGES**.
- 40154 **ASYNCHRONOUS EVENTS**
- 40155 Default.
- 40156 **STDOUT**
- 40157 By default, the standard output shall contain an entry for each input file of the form:
- 40158 "%d %d %d %s\n", <newlines>, <words>, <bytes>, <file>
- 40159 If the **-m** option is specified, the number of characters shall replace the <bytes> field in this
- 40160 format.
- 40161 If any options are specified and the **-l** option is not specified, the number of <newline>
- 40162 characters shall not be written.
- 40163 If any options are specified and the **-w** option is not specified, the number of words shall not be
- 40164 written.
- 40165 If any options are specified and neither **-c** nor **-m** is specified, the number of bytes or characters
- 40166 shall not be written.
- 40167 If no input *file* operands are specified, no name shall be written and no <blank> characters
- 40168 preceding the path name shall be written.
- 40169 If more than one input *file* operand is specified, an additional line shall be written, of the same
- 40170 format as the other lines, except that the word **total** (in the POSIX locale) shall be written instead
- 40171 of a path name and the total of each column shall be written as appropriate. Such an additional
- 40172 line, if any, is written at the end of the output.
- 40173 **STDERR**
- 40174 Used only for diagnostic messages.
- 40175 **OUTPUT FILES**
- 40176 None.
- 40177 **EXTENDED DESCRIPTION**
- 40178 None.
- 40179 **EXIT STATUS**
- 40180 The following exit values shall be returned:
- 40181 0 Successful completion.
- 40182 >0 An error occurred.
- 40183 **CONSEQUENCES OF ERRORS**
- 40184 Default.

40185 APPLICATION USAGE

40186 The **-m** option is not a switch, but an option at the same level as **-c**. Thus, to produce the full
40187 default output with character counts instead of bytes, the command required is:

40188 `wc -mlw`

40189 EXAMPLES

40190 None.

40191 RATIONALE

40192 The output file format pseudo-*printf()* string differs from the the System V version of *wc*:

40193 `"%7d%7d%7d %s\n"`

40194 which produces possibly ambiguous and unparseable results for very large files, as it assumes no
40195 number shall exceed six digits.

40196 Some historical implementations use only <space>, <tab>, and <newline> as word separators.
40197 The equivalent of the ISO C standard *isspace()* function is more appropriate.

40198 The **-c** option stands for “character” count, even though it counts bytes. This stems from the
40199 sometimes erroneous historical view that bytes and characters are the same size. Due to
40200 international requirements, the **-m** option (reminiscent of “multi-byte”) was added to obtain
40201 actual character counts.

40202 Early proposals only specified the results when input files were text files. The current
40203 specification more closely matches historical practice. (Bytes, words, and <newline>s are
40204 counted separately and the results are written when an end-of-file is detected.)

40205 Historical implementations of the *wc* utility only accepted one argument to specify the options
40206 **-c**, **-l**, and **-w**. Some of them also had multiple occurrences of an option cause the
40207 corresponding count to be written multiple times and had the order of specification of the
40208 options affect the order of the fields on output, but did not document either of these. Because
40209 common usage either specifies no options or only one option, and because none of this was
40210 documented, the changes required by this volume of IEEE Std. 1003.1-200x should not break
40211 many historical applications (and do not break any historical portable applications).

40212 FUTURE DIRECTIONS

40213 None.

40214 SEE ALSO

40215 *cksum*

40216 CHANGE HISTORY

40217 First released in Issue 2.

40218 Issue 4

40219 Aligned with the ISO/IEC 9945-2: 1993 standard.

40220 **NAME**40221 *what* — identify SCCS files (**DEVELOPMENT**)40222 **SYNOPSIS**40223 XSI *what* [-s] *file...*

40224

40225 **DESCRIPTION**

40226 The *what* utility shall search the given files for all occurrences of the pattern that *get* (see *get* (on page 2685)) substitutes for %Z% ("@(#) ") and shall write to standard output what follows until

40227 the first occurrence of one of the following:

40228

40229 " > newline \ NUL

40230 **OPTIONS**

40231 The *what* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section

40232 12.2, Utility Syntax Guidelines.

40233 The following option is supported:

40234 -s Quit after finding the first occurrence of the pattern in each file.

40235 **OPERANDS**

40236 The following operands shall be supported:

40237 *file* A path name of a file to search.40238 **STDIN**

40239 Not used.

40240 **INPUT FILES**

40241 The input files are of any file type.

40242 **ENVIRONMENT VARIABLES**40243 The following environment variables shall affect the execution of *what*:

40244 *LANG* Provide a default value for the internationalization variables that are unset or null.

40245 If *LANG* is unset or null, the corresponding value from the implementation-

40246 defined default locale shall be used. If any of the internationalization variables

40247 contains an invalid setting, the utility shall behave as if none of the variables had

40248 been defined.

40249 *LC_ALL* If set to a non-empty string value, override the values of all the other

40250 internationalization variables.

40251 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as

40252 characters (for example, single-byte as opposed to multi-byte characters in

40253 arguments and input files).

40254 *LC_MESSAGES*

40255 Determine the locale that should be used to affect the format and contents of

40256 diagnostic messages written to standard error.

40257 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.40258 **ASYNCHRONOUS EVENTS**

40259 Default.

40260 **STDOUT**

40261 The standard output shall consist of the following for each *file* operand:

40262 "%s:\n\t%s\n", <pathname>, <identification string>

40263 **STDERR**

40264 Used only for diagnostic messages.

40265 **OUTPUT FILES**

40266 None.

40267 **EXTENDED DESCRIPTION**

40268 None.

40269 **EXIT STATUS**

40270 The following exit values shall be returned:

40271 0 Any matches were found.

40272 1 Otherwise.

40273 **CONSEQUENCES OF ERRORS**

40274 Default.

40275 **APPLICATION USAGE**

40276 The *what* utility is intended to be used in conjunction with the SCCS command *get*, which automatically inserts identifying information, but it can also be used where the information is inserted by any other means.

40279 When the string "@(#)" is included in a library routine in a shared library, it might not be found
40280 in an **a.out** file using that library routine.

40281 **EXAMPLES**

40282 If the C-language program in file **f.c** contains:

40283 char ident[] = "@(#)identification information";

40284 and **f.c** is compiled to yield **f.o** and **a.out**, then the command:

40285 what f.c f.o a.out

40286 writes:

40287 f.c:
40288 identification information

40289 ...

40290 f.o:
40291 identification information

40292 ...

40293 a.out:
40294 identification information

40295 ...

40296 **RATIONALE**

40297 None.

40298 **FUTURE DIRECTIONS**

40299 None.

40300 **SEE ALSO**

40301 *get*

40302 **CHANGE HISTORY**

40303 First released in Issue 2.

40304 **Issue 4**

40305 Format reorganized.

40306 Utility Syntax Guidelines support mandated.

40307 Internationalized environment variable support mandated.

40308 NAME

40309 **who** — display who is on the system

40310 SYNOPSIS

40311 UP **who** [-mTu]

40312

40313 XSI **who** [-mu]-s[-bHlprt][*file*]40314 **who** [-mTu][-abdHlprt][*file*]40315 **who** -q [*file*]40316 **who** am i40317 **who** am I

40318

40319 DESCRIPTION

40320 The *who* utility shall list various pieces of information about accessible users. The domain of
40321 accessibility is implementation-defined.40322 XSI Based on the options given, *who* can also list the user's name, terminal line, login time, elapsed
40323 time since activity occurred on the line, and the process ID of the command interpreter for each
40324 current system user.

40325 OPTIONS

40326 The *who* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
40327 12.2, Utility Syntax Guidelines.40328 The following options shall be supported. The metavariables, such as *<line>*, refer to fields
40329 described in the STDOUT section.40330 XSI **-a** Process the implementation-defined database or named file with the **-b**, **-d**, **-l**, **-p**,
40331 **-r**, **-t**, **-T** and **-u** options turned on.40332 XSI **-b** Write the time and date of the last reboot.40333 XSI **-d** Write a list of all processes that have expired and not been respawned by the *init*
40334 system process. The *<exit>* field appears for dead processes and contains the
40335 termination and exit values of the dead process. This can be useful in determining
40336 why a process terminated.40337 XSI **-H** Write column headings above the regular output.40338 XSI **-l** (The letter ell.) List only those lines on which the system is waiting for someone to
40339 login. The *<name>* field is **LOGIN** in such cases. Other fields are the same as for
40340 user entries except that the *<state>* field does not exist.40341 **-m** Output only information about the current terminal.40342 XSI **-p** List any other process that is currently active and has been previously spawned by
40343 *init*.40344 XSI **-q** (Quick.) List only the names and the number of users currently logged on. When
40345 this option is used, all other options are ignored.40346 XSI **-r** Write the current *run-level* of the *init* process.40347 XSI **-s** List only the *<name>*, *<line>*, and *<time>* fields. This is the default case.40348 XSI **-t** Indicate the last change to the system clock.

40349 **-T** Show the state of each terminal, as described in the STDOUT section.

40350 XSI **-u** This option lists only those users who are currently logged in. Output the user's "idle time" in addition to any other information. The idle time is the time since any activity occurred on the user's terminal. The method of determining this is unspecified. The *<name>* is the user's login name. The *<line>* is the name of the line as found in the directory */dev*. The *<time>* is the time that the user logged in. The *<activity>* is the number of hours and minutes since activity last occurred on that particular line. A dot indicates that the terminal has seen activity in the last minute and is therefore "current". If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked *<old>*. This field is useful when trying to determine whether a person is working at the terminal or not. The *<pid>* is the process ID of the user's login process.

40361 OPERANDS

40362 XSI The following operands shall be supported:

40363 **am i, am I** In the POSIX locale, limit the output to describing the invoking user, equivalent to the **-m** option. The **am** and **i** or **I** must be separate arguments.

40365 *file* Specify a path name of a file to substitute for the implementation-defined database of logged-on users that *who* uses by default.

40367 STDIN

40368 Not used.

40369 INPUT FILES

40370 None.

40371 ENVIRONMENT VARIABLES

40372 The following environment variables shall affect the execution of *who*:

40373 **LANG** Provide a default value for the internationalization variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined.

40378 **LC_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

40380 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

40383 **LC_MESSAGES** Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

40386 **LC_TIME** Determine the locale used for the format and contents of the date and time strings.

40387 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.

40388 ASYNCHRONOUS EVENTS

40389 Default.

40390 **STDOUT**

40391 XSI OF XSI-conformant systems shall write the default information to the standard output in the
 40392 following general format:

40393 `<name>[<state>]<line><time>[<activity>][<pid>][<comment>][<exit>]`

40394 The following format shall be used for the **-T** option:

40395 `"%s %c %s %s\n" <name>, <terminal state>, <terminal name>,
 40396 <time of login>`

40397 where *<terminal state>* is one of the following characters:

- 40398 + The terminal allows write access to other users.
- 40399 - The terminal denies write access to other users.
- 40400 ? The terminal write-access state cannot be determined.

40401 In the POSIX locale, the *<time of login>* shall be equivalent in format to the output of:

40402 `date +"%b %e %H:%M"`

40403 If the **-u** option is used with **-T**, the idle time shall be added to the end of the previous format in
 40404 an unspecified format.

40405 **STDERR**

40406 Used only for diagnostic messages.

40407 **OUTPUT FILES**

40408 None.

40409 **EXTENDED DESCRIPTION**

40410 None.

40411 **EXIT STATUS**

40412 The following exit values shall be returned:

- 40413 0 Successful completion.
- 40414 >0 An error occurred.

40415 **CONSEQUENCES OF ERRORS**

40416 Default.

40417 **APPLICATION USAGE**

40418 The name *init* used for the system process is the most commonly used on historical systems, but
 40419 it may vary.

40420 The “domain of accessibility” referred to is a broad concept that permits interpretation either on
 40421 a very secure basis or even to allow a network-wide implementation like the historical *rwho*.

40422 **EXAMPLES**

40423 None.

40424 **RATIONALE**

40425 Due to differences between historical implementations, the base options provided were a
 40426 compromise to allow users to work with those functions. The standard developers also
 40427 considered removing all the options, but felt that these options offered users valuable
 40428 functionality. Additional options to match historical systems are available on XSI-conformant
 40429 systems.

- 40430 It is recognized that the *who* command may be of limited usefulness, especially in a multi-level
40431 secure environment. The standard developers considered, however, that having some standard
40432 method of determining the “accessibility” of other users would aid user portability.
- 40433 No format was specified for the default *who* output for systems not supporting the XSI
40434 Extension. In such a user-oriented command, designed only for human use, this was not
40435 considered to be a deficiency.
- 40436 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, and *write* require
40437 that they use the same format.
- 40438 **FUTURE DIRECTIONS**
- 40439 None.
- 40440 **SEE ALSO**
- 40441 *msg*
- 40442 **CHANGE HISTORY**
- 40443 First released in Issue 2.
- 40444 **Issue 4**
- 40445 Aligned with the ISO/IEC 9945-2:1993 standard.
- 40446 **Issue 6**
- 40447 This utility is now marked as part of the User Portability Utilities option.

40448 NAME

40449 write — write to another user

40450 SYNOPSIS

40451 UP write *user_name* [*terminal*]

40452

40453 DESCRIPTION

40454 The *write* utility shall read lines from the user's standard input and write them to the terminal of
40455 another user. When first invoked, it shall write the message:40456 **Message from** *sender-login-id* (*sending-terminal*) [*date*]...40457 to *user_name*. When it has successfully completed the connection, the sender's terminal shall be
40458 alerted twice to indicate that what the sender is typing is being written to the recipient's
40459 terminal.

40460 If the recipient wants to reply, this can be accomplished by typing:

40461 write *sender-login-id* [*sending-terminal*]40462 upon receipt of the initial message. Whenever a line of input as delimited by a NL, EOF, or EOL
40463 special character (see the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General
40464 Terminal Interface) is accumulated while in canonical input mode, the accumulated data shall be
40465 written on the other user's terminal. Characters shall be processed as follows:

- 40466
- Typing the <alert> character shall write the alert character to the recipient's terminal.
 - 40467 • Typing the erase and kill characters shall affect the sender's terminal in the manner described
40468 by the **termios** interface in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11,
40469 General Terminal Interface.
 - 40470 • Typing the interrupt or end-of-file characters shall cause *write* to write an appropriate
40471 message ("EOT\n" in the POSIX locale) to the recipient's terminal and exit.
 - 40472 • Typing characters from *LC_CTYPE* classifications **print** or **space** shall cause those characters
40473 to be sent to the recipient's terminal.
 - 40474 • When and only when the *stty iexten* local mode is enabled, the existence and processing of
40475 additional special control characters and multi-byte or single-byte functions is
40476 implementation-defined.
 - 40477 • Typing other non-printable characters shall cause implementation-defined sequences of
40478 printable characters to be written to the recipient's terminal.

40479 To write to a user who is logged in more than once, the *terminal* argument can be used to indicate
40480 which terminal to write to; otherwise, the recipient's terminal is selected in an implementation-
40481 defined manner and an informational message is written to the sender's standard output,
40482 indicating which terminal was chosen.40483 Permission to be a recipient of a *write* message can be denied or granted by use of the *mesg*
40484 utility. However, a user's privilege may further constrain the domain of accessibility of other
40485 users' terminals. The *write* utility shall fail when the user lacks the appropriate privileges to
40486 perform the requested action.

40487 OPTIONS

40488 None.

40489 **OPERANDS**

40490 The following operands shall be supported:

40491 *user_name* Login name of the person to whom the message shall be written. The application
40492 shall ensure that this operand is of the form returned by the *who* utility.

40493 *terminal* Terminal identification in the same format provided by the *who* utility.

40494 **STDIN**

40495 Lines to be copied to the recipient's terminal is read from standard input.

40496 **INPUT FILES**

40497 None.

40498 **ENVIRONMENT VARIABLES**

40499 The following environment variables shall affect the execution of *write*:

40500 *LANG* Provide a default value for the internationalization variables that are unset or null.
40501 If *LANG* is unset or null, the corresponding value from the implementation-
40502 defined default locale shall be used. If any of the internationalization variables
40503 contains an invalid setting, the utility shall behave as if none of the variables had
40504 been defined.

40505 *LC_ALL* If set to a non-empty string value, override the values of all the other
40506 internationalization variables.

40507 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
40508 characters (for example, single-byte as opposed to multi-byte characters in
40509 arguments and input files). If the recipient's locale does not use an *LC_CTYPE*
40510 equivalent to the sender's, the results are undefined.

40511 *LC_MESSAGES*

40512 Determine the locale that should be used to affect the format and contents of
40513 diagnostic messages written to standard error and informative messages written to
40514 standard output.

40515 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

40516 **ASYNCHRONOUS EVENTS**

40517 If an interrupt signal is received, *write* shall write an appropriate message on the recipient's
40518 terminal and exits with a status of zero. It shall take the standard action for all other signals.

40519 **STDOUT**

40520 An informational message shall be written to standard output if a recipient is logged in more
40521 than once.

40522 **STDERR**

40523 Used only for diagnostic messages.

40524 **OUTPUT FILES**

40525 The recipient's terminal is used for output.

40526 **EXTENDED DESCRIPTION**

40527 None.

40528 **EXIT STATUS**

40529 The following exit values shall be returned:

40530 0 Successful completion.

40531 >0 The addressed user is not logged on or the addressed user denies permission.

40532 **CONSEQUENCES OF ERRORS**

40533 Default.

40534 **APPLICATION USAGE**

40535 The *talk* utility is considered by some users to be a more usable utility on full-screen terminals.

40536 **EXAMPLES**

40537 None.

40538 **RATIONALE**

40539 The *write* utility was included in this volume of IEEE Std. 1003.1-200x since it can be
40540 implemented on all terminal types. The standard developers considered the *talk* utility, which
40541 cannot be implemented on certain terminals, to be a “better” communications interface. Both of
40542 these programs are in widespread use on historical implementations. Therefore, the standard
40543 developers decided that both utilities should be specified.

40544 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*
40545 require that they all use or accept the same format.

40546 **FUTURE DIRECTIONS**

40547 None.

40548 **SEE ALSO**

40549 *mesg*, *talk*, *who*, the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 11, General
40550 Terminal Interface

40551 **CHANGE HISTORY**

40552 First released in Issue 2.

40553 **Issue 4**

40554 Aligned with the ISO/IEC 9945-2:1993 standard.

40555 **Issue 5**

40556 FUTURE DIRECTIONS section added.

40557 **Issue 6**

40558 This utility is now marked as part of the User Portability Utilities option.

40559 The normative text is reworded to avoid use of the term “must” for application requirements.

40560 NAME

40561 xargs — construct argument lists and invoke utility

40562 SYNOPSIS

```
40563 xsi xargs [-t][-p][-E eofstr][-I replstr][-L number][-n number [-x]]
40564 [-s size][utility [argument...]]
```

40565 DESCRIPTION

40566 The *xargs* utility shall construct a command line consisting of the *utility* and *argument* operands
 40567 specified followed by as many arguments read in sequence from standard input as fit in length
 40568 and number constraints specified by the options. The *xargs* utility shall then invoke the
 40569 constructed command line and wait for its completion. This sequence shall be repeated until one
 40570 of the following occurs:

- 40571 • An end-of-file condition is detected on standard input.
- 40572 • The logical end-of-file string (see the **-E eofstr** option) is found on standard input after
 40573 double-quote processing, apostrophe processing, and backslash escape processing (see next
 40574 paragraph).
- 40575 • An invocation of a constructed command line returns an exit status of 255.

40576 The application shall ensure that arguments in the standard input are separated by unquoted
 40577 <blank> characters, or unescaped <blank> characters or <newline> characters. A string of zero
 40578 or more non-double-quote (' " ') and non-<newline> characters can be quoted by enclosing
 40579 them in double-quotes. A string of zero or more non-apostrophe (' \ ' ') and non-<newline>
 40580 characters can be quoted by enclosing them in apostrophes. Any unquoted character can be
 40581 escaped by preceding it with a backslash. The utility shall be executed one or more times until
 40582 the end-of-file is reached or the logical end-of file string is found. The results are unspecified if
 40583 the utility named by *utility* attempts to read from its standard input.

40584 The generated command line length shall be the sum of the size in bytes of the utility name and
 40585 each argument treated as strings, including a null byte terminator for each of these strings. The
 40586 *xargs* utility shall limit the command line length such that when the command line is invoked,
 40587 the combined argument and environment lists (see the *exec* family of functions in the System
 40588 Interfaces volume of IEEE Std. 1003.1-200x) shall not exceed {ARG_MAX}-2 048 bytes. Within
 40589 this constraint, if neither the **-n** nor the **-s** option is specified, the default command line length
 40590 shall be at least {LINE_MAX}.

40591 OPTIONS

40592 The *xargs* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 40593 12.2, Utility Syntax Guidelines.

40594 The following options shall be supported:

40595 **-E eofstr** Use *eofstr* as the logical end-of-file string. If **-E** is not specified, it is unspecified
 40596 whether the logical end-of-file string is the underscore character (' _ ') or the end-
 40597 of-file string capability is disabled. When *eofstr* is the null string, the logical end-
 40598 of-file string capability shall be disabled and underscore characters shall be taken
 40599 literally.

40600 xsi **-I replstr** Insert mode: *utility* is executed for each line from standard input, taking the entire
 40601 line as a single argument, inserting it in *arguments* for each occurrence of *replstr*. A
 40602 maximum of five arguments in *arguments* can each contain one or more instances
 40603 of *replstr*. Any <blank> characters at the beginning of each line shall be ignored.
 40604 Constructed arguments cannot grow larger than 255 bytes. Option **-x** is forced on.

- 40605 XSI **-L number** The *utility* shall be executed for each non-empty *number* lines of arguments from standard input. The last invocation of *utility* shall be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first <newline> character unless the last character of the line is a <blank> character; a trailing <blank> character signals continuation to the next non-empty line, inclusive. The **-L** and **-n** options are mutually-exclusive; the last one specified shall take effect.
- 40606
- 40607
- 40608
- 40609
- 40610
- 40611 **-n number** Invoke *utility* using as many standard input arguments as possible, up to *number* (a positive decimal integer) arguments maximum. Fewer arguments shall be used if:
- 40612
- The command line length accumulated exceeds the size specified by the **-s** option (or {LINE_MAX} if there is no **-s** option).
 - The last iteration has fewer than but not zero, operands remaining.
- 40613
- 40614
- 40615
- 40616 **-p** Prompt mode: the user is asked whether to execute *utility* at each invocation. Trace mode (**-t**) is turned on to write the command instance to be executed, followed by a prompt to standard error. An affirmative response read from */dev/tty* shall execute the command; otherwise, that particular invocation of *utility* shall be skipped.
- 40617
- 40618
- 40619
- 40620
- 40621 **-s size** Invoke *utility* using as many standard input arguments as possible yielding a command line length less than *size* (a positive decimal integer) bytes. Fewer arguments shall be used if:
- 40622
- 40623
- The total number of arguments exceeds that specified by the **-n** option.
 - The total number of lines exceeds that specified by the **-L** option.
 - End-of-file is encountered on standard input before *size* bytes are accumulated.
- 40624
- 40625 XSI
- 40626
- 40627 Values of *size* up to at least {LINE_MAX} bytes shall be supported, provided that the constraints specified in the DESCRIPTION are met. It shall not be considered an error if a value larger than that supported by the implementation or exceeding the constraints specified in the DESCRIPTION is given; *xargs* shall use the largest value it supports within the constraints.
- 40628
- 40629
- 40630
- 40631
- 40632 **-t** Enable trace mode. Each generated command line shall be written to standard error just prior to invocation.
- 40633
- 40634 **-x** Terminate if a command line containing *number* arguments (see the **-n** option above) or *number* lines (see the **-L** option above) will not fit in the implied or specified size (see the **-s** option above).
- 40635 XSI
- 40636

40637 OPERANDS

40638 The following operands shall be supported:

- 40639 *utility* The name of the utility to be invoked, found by search path using the *PATH* environment variable, described in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables. If *utility* is omitted, the default shall be the *echo* utility. If the *utility* operand names any of the special built-in utilities in Section 2.15 (on page 2276), the results are undefined.
- 40640
- 40641
- 40642
- 40643
- 40644 *argument* An initial option or operand for the invocation of *utility*.

40645 STDIN

40646 The standard input shall be a text file. The results are unspecified if an end-of-file condition is detected immediately following an escaped <newline> character.

40647

40648 **INPUT FILES**

40649 The file `/dev/tty` is used to read responses required by the `-p` option.

40650 **ENVIRONMENT VARIABLES**

40651 The following environment variables shall affect the execution of *xargs*:

40652 **LANG** Provide a default value for the internationalization variables that are unset or null. If *LANG* is unset or null, the corresponding value from the implementation-defined default locale shall be used. If any of the internationalization variables contains an invalid setting, the utility shall behave as if none of the variables had been defined.

40657 **LC_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

40659 **LC_COLLATE**

40660 Determine the locale for the behavior of ranges, equivalence classes and multi-character collating elements used in the extended regular expression defined for the **yesexpr** locale keyword in the *LC_MESSAGES* category.

40663 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes used in the extended regular expression defined for the **yesexpr** locale keyword in the *LC_MESSAGES* category.

40668 **LC_MESSAGES**

40669 Determine the locale for the processing of affirmative responses and that should be used to affect the format and contents of diagnostic messages written to standard error.

40672 XSI **NLS_PATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.

40673 **PATH** Determine the location of *utility*, as described in the Base Definitions volume of IEEE Std. 1003.1-200x, Chapter 8, Environment Variables.

40675 **ASYNCHRONOUS EVENTS**

40676 Default.

40677 **STDOUT**

40678 Not used.

40679 **STDERR**

40680 Used for diagnostic messages and the `-t` and `-p` options. If the `-t` option is specified, the *utility* and its constructed argument list shall be written to standard error, as it will be invoked, prior to invocation. If `-p` is specified, a prompt of the following format shall be written (in the POSIX locale):

40684 " ? . . . "

40685 at the end of the line of the output from `-t`.

40686 **OUTPUT FILES**

40687 None.

40688 **EXTENDED DESCRIPTION**

40689 None.

40690 **EXIT STATUS**

40691 The following exit values shall be returned:

- 40692 0 All invocations of *utility* returned exit status zero.
- 40693 1-125 A command line meeting the specified requirements could not be assembled, one or more of the invocations of *utility* returned a non-zero exit status, or some other error occurred.
- 40694
- 40695
- 40696 126 The utility specified by *utility* was found but could not be invoked.
- 40697 127 The utility specified by *utility* could not be found.

40698 **CONSEQUENCES OF ERRORS**

40699 If a command line meeting the specified requirements cannot be assembled, the utility cannot be invoked, an invocation of the utility is terminated by a signal, or an invocation of the utility exits with exit status 255, the *xargs* utility shall write a diagnostic message and exit without processing any remaining input.

40700

40701

40702

40703 **APPLICATION USAGE**

40704 The 255 exit status allows a utility being used by *xargs* to tell *xargs* to terminate if it knows no further invocations using the current data stream succeeds. Thus, *utility* should explicitly *exit* with an appropriate value to avoid accidentally returning with 255.

40705

40706

40707 Note that input is parsed as lines; <blank> characters separate arguments. If *xargs* is used to bundle output of commands like *find dir -print* or *ls* into commands to be executed, unexpected results are likely if any file names contain any <blank> characters or <newline> characters. This can be fixed by using *find* to call a script that converts each file found into a quoted string that is then piped to *xargs*. Note that the quoting rules used by *xargs* are not the same as in the shell. They were not made consistent here because existing applications depend on the current rules and the shell syntax is not fully compatible with it. An easy rule that can be used to transform any string into a quoted form that *xargs* interprets correctly is to precede each character in the string with a backslash.

40708

40709

40710

40711

40712

40713

40714

40715

40716 On implementations with a large value for {ARG_MAX}, *xargs* may produce command lines longer than {LINE_MAX}. For invocation of utilities, this is not a problem. If *xargs* is being used to create a text file, users should explicitly set the maximum command line length with the *-s* option.

40717

40718

40719

40720 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish “failure to find a utility” from “invoked utility exited with an error indication”. The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for “normal error conditions” and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for any other reason.

40721

40722

40723

40724

40725

40726

40727

40728

40729

40730 **EXAMPLES**

- 40731 1. The following command combines the output of the parenthesised commands onto one line, which is then written to the end-of-file **log**:
- 40732
- 40733

```
(logname; date; printf "%s\n" "$0 $*") | xargs >>log
```
- 40734 2. The following command invokes *diff* with successive pairs of arguments originally typed as command line arguments (assuming there are no embedded <blank> characters in the
- 40735

40736 elements of the original argument list):
 40737 `printf "%s\n" "$*" | xargs -n 2 -x diff`
 40738 3. The user is asked which files in the current directory shall be archived. The files are
 40739 archived into **arch**; *a*, one at a time, or *b*, many at a time.
 40740 *a.* `ls | xargs -p -L 1 ar -r arch`
 40741 *b.* `ls | xargs -p -L 1 | xargs ar -r arch`
 40742 4. The following executes with successive pairs of arguments originally typed as command
 40743 line arguments:
 40744 `echo $* | xargs -n 2 diff`
 40745 5. On XSI-conformant systems, the following moves all files from directory **\$1** to directory **\$2**,
 40746 and echo each move command just before doing it:
 40747 `ls $1 | xargs -I {} -t mv $1/{ } $2/{ }`

40748 RATIONALE

40749 The *xargs* utility was usually found only in System V-based systems; BSD systems included an
 40750 *apply* utility that provided functionality similar to *xargs -n number*. The SVID lists *xargs* as a
 40751 software development extension. This volume of IEEE Std. 1003.1-200x does not share the view
 40752 that it is used only for development, and therefore it is not optional.

40753 The classic application of the *xargs* utility is in conjunction with the *find* utility to reduce the
 40754 number of processes launched by a simplistic use of the *find-exec* combination. The *xargs* utility
 40755 is also used to enforce an upper limit on memory required to launch a process. With this basis in
 40756 mind, this volume of IEEE Std. 1003.1-200x selected only the minimal features required.

40757 Although the 255 exit status is mostly an accident of historical implementations, it allows a
 40758 utility being used by *xargs* to tell *xargs* to terminate if it knows no further invocations using the
 40759 current data stream shall succeed. Any non-zero exit status from a utility falls into the 1-125
 40760 range when *xargs* exits. There is no statement of how the various non-zero utility exit status
 40761 codes are accumulated by *xargs*. The value could be the addition of all codes, their highest
 40762 value, the last one received, or a single value such as 1. Since no algorithm is arguably better
 40763 than the others, and since many of the standard utilities say little more (portably) than
 40764 “pass/fail”, no new algorithm was invented.

40765 Several other *xargs* options were withdrawn because simple alternatives already exist within this
 40766 volume of IEEE Std. 1003.1-200x. For example, the *-e eofstr* option can be replaced by features of
 40767 *sed*. The *-i replstr* option can be just as efficiently performed using a shell *for* loop. Since *xargs*
 40768 calls an *exec* function with each input line, the *-i* option does not usually exploit the grouping
 40769 capabilities of *xargs*.

40770 The requirement that *xargs* never produce command lines such that invocation of *utility* is
 40771 within 2 048 bytes of hitting the POSIX *exec* {ARG_MAX} limitations is intended to guarantee
 40772 that the invoked utility has room to modify its environment variables and command line
 40773 arguments and still be able to invoke another utility. Note that the minimum {ARG_MAX}
 40774 allowed by the System Interfaces volume of IEEE Std. 1003.1-200x is 4 096 bytes and the
 40775 minimum value allowed by the this volume of IEEE Std. 1003.1-200x is 2 048 bytes; therefore, the
 40776 2 048 bytes difference seems reasonable. Note, however, that *xargs* may never be able to invoke a
 40777 utility if the environment passed in to *xargs* comes close to using {ARG_MAX} bytes.

40778 The version of *xargs* required by this volume of IEEE Std. 1003.1-200x is required to wait for the
 40779 completion of the invoked command before invoking another command. This was done because
 40780 historical scripts using *xargs* assumed sequential execution. Implementations wanting to provide

40781 parallel operation of the invoked utilities are encouraged to add an option enabling parallel
 40782 invocation, but should still wait for termination of all of the children before *xargs* terminates
 40783 normally.

40784 The `-e` option was omitted from the ISO POSIX-2:1993 standard in the belief that the *eofstr*
 40785 option-argument was recognized only when it was on a line by itself and before quote and
 40786 escape processing were performed, and that the logical end-of-file processing was only enabled
 40787 if a `-e` option was specified. In that case, a simple *sed* script could be used to duplicate the `-e`
 40788 functionality. Further investigation revealed that:

- 40789 • The logical end-of-file string was checked for after quote and escape processing, making a *sed*
 40790 script that provided equivalent functionality much more difficult to write.
- 40791 • The default was to perform logical end-of-file processing with an underscore as the logical
 40792 end-of-file string.

40793 To correct this misunderstanding, the `-E eofstr` option was adopted from the X/Open Portability
 40794 Guide. Users should note that the description of the `-E` option matches historical documentation
 40795 of the `-e` option (which was not adopted because it did not support the Utility Syntax
 40796 Guidelines), by saying that if *eofstr* is the null string, logical end-of-file processing is disabled.
 40797 Historical implementations of *xargs* actually did not disable logical end-of-file processing; they
 40798 treated a null argument found in the input as a logical end-of-file string. (A null *string* argument
 40799 could be generated using single or double quotes (' ' or " "). Since this behavior was not
 40800 documented historically, it is considered to be a bug.

40801 FUTURE DIRECTIONS

40802 None.

40803 SEE ALSO

40804 *echo*

40805 CHANGE HISTORY

40806 First released in Issue 2.

40807 Issue 4

40808 Aligned with the ISO/IEC 9945-2:1993 standard.

40809 Issue 5

40810 Second FUTURE DIRECTION added.

40811 Issue 6

40812 The obsolescent `-e`, `-i`, and `-l` options are removed.

40813 The following new requirements on POSIX implementations derive from alignment with the
 40814 Single UNIX Specification:

- 40815 • The `-p` option is added.
- 40816 • In the INPUT FILES section, the file `/dev/tty` is used to read responses required by the `-p`
 40817 option.
- 40818 • The STDERR section is updated to describe the `-p` option.

40819 The description of the `-E` option is aligned with the ISO POSIX-2:1993 standard.

40820 The normative text is reworded to avoid use of the term “must” for application requirements.

40821 NAME

40822 yacc — yet another compiler compiler (DEVELOPMENT)

40823 SYNOPSIS

40824 yacc [-dltv][-b *file_prefix*][-p *sym_prefix*] *grammar*

40825 DESCRIPTION

40826 The *yacc* utility shall read a description of a context-free grammar in *file* and write C source code,
 40827 conforming to the ISO C standard, to a code file, and optionally header information into a
 40828 header file, in the current directory. The C code shall define a function and related routines and
 40829 macros for an automaton that executes a parsing algorithm meeting the requirements in
 40830 **Algorithms** (on page 3288).

40831 The form and meaning of the grammar are described in the EXTENDED DESCRIPTION section.

40832 The C source code and header file shall be produced in a form suitable as input for the C
 40833 compiler (see *c99* (on page 2425)).

40834 OPTIONS

40835 The *yacc* utility shall conform to the Base Definitions volume of IEEE Std. 1003.1-200x, Section
 40836 12.2, Utility Syntax Guidelines.

40837 The following options shall be supported:

40838 **-b *file_prefix*** Use *file_prefix* instead of *y* as the prefix for all output file names. The code file
 40839 *y.tab.c*, the header file *y.tab.h* (created when **-d** is specified), and the description
 40840 file *y.output* (created when **-v** is specified), shall be changed to *file_prefix.tab.c*,
 40841 *file_prefix.tab.h*, and *file_prefix.output*, respectively.

40842 **-d** Write the header file; by default only the code file is written. The **#define**
 40843 statements that associate the token codes assigned by *yacc* with the user-declared
 40844 token names. This allows source files other than *y.tab.c* to access the token codes.

40845 **-l** Produce a code file that does not contain any **#line** constructs. If this option is not
 40846 present, it is unspecified whether the code file or header file contains **#line**
 40847 directives. This should only be used after the grammar and the associated actions
 40848 are fully debugged.

40849 **-p *sym_prefix*** Use *sym_prefix* instead of *yy* as the prefix for all external names produced by *yacc*.
 40850 The names affected shall include the functions *yyparse*, *yylex*, and *yyerror*, and the
 40851 variables *yylval*, *yychar*, and *yydebug*. (In the remainder of this section, the six
 40852 symbols cited are referenced using their default names only as a notational
 40853 convenience.) Local names may also be affected by the **-p** option; however, the **-p**
 40854 option shall not affect **#define** symbols generated by *yacc*.

40855 **-t** Modify conditional compilation directives to permit compilation of debugging
 40856 code in the code file. Runtime debugging statements shall always be contained in
 40857 the code file, but by default conditional compilation directives prevent their
 40858 compilation.

40859 **-v** Write a file containing a description of the parser and a report of conflicts
 40860 generated by ambiguities in the grammar.

40861 OPERANDS

40862 The following operand is required:

40863 *grammar* A path name of a file containing instructions, hereafter called *grammar*, for which a
 40864 parser is to be created. The format for the grammar is described in the EXTENDED
 40865 DESCRIPTION section.

40866 **STDIN**

40867 Not used.

40868 **INPUT FILES**40869 The file *grammar* shall be a text file formatted as specified in the EXTENDED DESCRIPTION
40870 section.40871 **ENVIRONMENT VARIABLES**40872 The following environment variables shall affect the execution of *yacc*:40873 **LANG** Provide a default value for the internationalization variables that are unset or null.
40874 If *LANG* is unset or null, the corresponding value from the implementation-
40875 defined default locale shall be used. If any of the internationalization variables
40876 contains an invalid setting, the utility shall behave as if none of the variables had
40877 been defined.40878 **LC_ALL** If set to a non-empty string value, override the values of all the other
40879 internationalization variables.40880 **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
40881 characters (for example, single-byte as opposed to multi-byte characters in
40882 arguments and input files).40883 **LC_MESSAGES**40884 Determine the locale that should be used to affect the format and contents of
40885 diagnostic messages written to standard error.40886 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC_MESSAGES*.40887 The *LANG* and *LC_** variables affect the execution of the *yacc* utility as stated. The *main* function
40888 defined in **Yacc Library** (on page 3288) shall call:40889 `setlocale(LC_ALL, " ")`40890 and thus, the program generated by *yacc* also shall be affected by the contents of these variables
40891 at runtime.40892 **ASYNCHRONOUS EVENTS**

40893 Default.

40894 **STDOUT**

40895 Not used.

40896 **STDERR**40897 If shift/reduce or reduce/reduce conflicts are detected in *grammar*, *yacc* writes a report of those
40898 conflicts to the standard error in an unspecified format.

40899 Standard error is also used for diagnostic messages.

40900 **OUTPUT FILES**40901 The code file, the header file, and the description file shall be text files. All are described in the
40902 following sections.

40903 **Code File**

40904 This file shall contain the C source code for the *yyparse* routine. It shall contain code for the
40905 various semantic actions with macro substitution performed on them as described in the
40906 EXTENDED DESCRIPTION section. It also shall contain a copy of the **#define** statements in the
40907 header file. If a **%union** declaration is used, the declaration for YYSTYPE shall be also included
40908 in this file.

40909 **Header File**

40910 The header file shall contain **#define** statements that associate the token numbers with the token
40911 names. This allows source files other than the code file to access the token codes. If a **%union**
40912 declaration is used, the declaration for YYSTYPE and an *extern YYSTYPE yylval* declaration shall
40913 be also included in this file.

40914 **Description File**

40915 The description file shall be a text file containing a description of the state machine
40916 corresponding to the parser, using an unspecified format. Limits for internal tables (see **Limits**
40917 (on page 3288)) shall also be reported, in an implementation-defined manner. (Some
40918 implementations may use dynamic allocation techniques and have no specific limit values to
40919 report.)

40920 **EXTENDED DESCRIPTION**

40921 The *yacc* command accepts a language that is used to define a grammar for a target language to
40922 be parsed by the tables and code generated by *yacc*. The language accepted by *yacc* as a
40923 grammar for the target language is described below using the *yacc* input language itself.

40924 The input *grammar* includes rules describing the input structure of the target language and code
40925 to be invoked when these rules are recognized to provide the associated semantic action. The
40926 code to be executed shall appear as bodies of text that are intended to be C-language code. The
40927 C-language inclusions are presumed to form a correct function when processed by *yacc* into its
40928 output files. The code included in this way shall be executed during the recognition of the target
40929 language.

40930 Given a grammar, the *yacc* utility generates the files described in the OUTPUT FILES section.
40931 The code file can be compiled and linked using *cc* or *c99*. If the declaration and programs
40932 sections of the grammar file did not include definitions of *main*, *yylex*, and *yyerror*, the compiled
40933 output requires linking with externally supplied version of those functions. Default versions of
40934 *main* and *yyerror* are supplied in the *yacc* library and can be linked in by using the *-l y* operand to
40935 *c99*. The *yacc* library interfaces need not support interfaces with other than the default *yy*
40936 symbol prefix. The application provides the lexical analyzer function, *yylex*; the *lex* utility is
40937 specifically designed to generate such a routine.

40938 **Input Language**

40939 The application shall ensure that every specification file consists of three sections in order:
40940 *declarations*, *grammar rules*, and *programs*, separated by double percent signs ("%"). The
40941 declarations and programs sections can be empty. If the latter is empty, the preceding "%"
40942 mark separating it from the rules section can be omitted.

40943 The input is free form text following the structure of the grammar defined below.

40944 **Lexical Structure of the Grammar**

40945 The characters <blank>, <newline>, and <form-feed> shall be ignored, except that the
 40946 application shall ensure that they do not appear in names or multi-character reserved symbols.
 40947 Comments shall be enclosed in `"/* . . . */"`, and can appear wherever a name is valid.

40948 Names are of arbitrary length, made up of letters, periods ('. '), underscores ('_ '), and non-
 40949 initial digits. Uppercase and lowercase letters are distinct. Portable applications shall not use
 40950 names beginning in `yy` or `YY` since the `yacc` parser uses such names. Many of the names appear
 40951 in the final output of `yacc`, and thus they should be chosen to conform with any additional rules
 40952 created by the C compiler to be used. In particular they appear in `#define` statements.

40953 A literal shall consist of a single character enclosed in single-quotes ('\' '). All of the escape
 40954 sequences supported for character constants by the ISO C standard shall be supported by `yacc`.

40955 The relationship with the lexical analyzer is discussed in detail below.

40956 The application shall ensure that the NUL character is not used in grammar rules or literals.

40957 **Declarations Section**

40958 The declarations section is used to define the symbols used to define the target language and
 40959 their relationship with each other. In particular, much of the additional information required to
 40960 resolve ambiguities in the context-free grammar for the target language is provided here.

40961 Usually `yacc` assigns the relationship between the symbolic names it generates and their
 40962 underlying numeric value. The declarations section makes it possible to control the assignment
 40963 of these values.

40964 It is also possible to keep semantic information associated with the tokens currently on the parse
 40965 stack in a user-defined C-language **union**, if the members of the union are associated with the
 40966 various names in the grammar. The declarations section provides for this as well.

40967 The first group of declarators below all take a list of names as arguments. That list can optionally
 40968 be preceded by the name of a C union member (called a *tag* below) appearing within '`<`' and
 40969 '`>`'. (As an exception to the typographical conventions of the rest of this volume of
 40970 IEEE Std. 1003.1-200x, in this case `<tag>` does not represent a metavariable, but the literal angle
 40971 bracket characters surrounding a symbol.) The use of *tag* specifies that the tokens named on this
 40972 line shall be of the same C type as the union member referenced by *tag*. This is discussed in
 40973 more detail below.

40974 For lists used to define tokens, the first appearance of a given token can be followed by a
 40975 positive integer (as a string of decimal digits). If this is done, the underlying value assigned to it
 40976 for lexical purposes is taken to be that number.

40977 `%token [<tag>] name [number][name [number]]. . .`

40978 Declares *names* to be a token. If *tag* is present, the C type for all tokens on this line shall be
 40979 declared to be the type referenced by *tag*. If a positive integer, *number*, follows a *name*, that
 40980 value shall be assigned to the token.

40981 `%left [<tag>] name [number][name [number]]. . .`

40982 `%right [<tag>] name [number][name [number]]. . .`

40983 Declares *name* to be a token, and assigns precedence to it. One or more lines, each beginning
 40984 with one of these symbols, can appear in this section. All tokens on the same line have the
 40985 same precedence level and associativity; the lines are in order of increasing precedence or
 40986 binding strength. `%left` denotes that the operators on that line are left associative, and
 40987 `%right` similarly denotes right associative operators. If *tag* is present, it shall declare a C
 40988 type for *names* as described for `%token`.

40989 %nonassoc [<tag>] *name* [*number*][*name* [*number*]]. . .
40990 Declares *name* to be a token, and indicates that this cannot be used associatively. If the
40991 parser encounters associative use of this token it reports an error. If *tag* is present, it shall
40992 declare a C type for *names* as described for **%token**.

40993 %type [<tag>] *name*. . .
40994 Declares that union member *names* are non-terminals, and thus it is required to have a *tag*
40995 field at its beginning. Because it deals with non-terminals only, assigning a token number or
40996 using a literal is also prohibited. If this construct is present, *yacc* shall perform type
40997 checking; if this construct is not present, the parse stack shall hold only the **int** type.

40998 Every name used in *grammar* undefined by a **%token**, **%left**, **%right**, or **%nonassoc** declaration is
40999 assumed to represent a non-terminal symbol. The *yacc* utility shall report an error for any non-
41000 terminal symbol that does not appear on the left side of at least one grammar rule.

41001 Once the type, precedence, or token number of a name is specified, it shall not be changed. If the
41002 first declaration of a token does not assign a token number, *yacc* shall assign a token number.
41003 Once this assignment is made, the token number shall not be changed by explicit assignment.

41004 The following declarators do not follow the previous pattern.

41005 %start *name*
41006 Declares the non-terminal *name* to be the *start symbol*, which represents the largest, most
41007 general structure described by the grammar rules. By default, it is the left-hand side of the
41008 first grammar rule; this default can be overridden with this declaration.

41009 %union { *body of union (in C)* }
41010 Declares the *yacc* value stack to be a union of the various types of values desired. By default,
41011 the values returned by actions (see below) and the lexical analyzer shall be integers. The
41012 *yacc* utility keeps track of types, and it shall insert corresponding union member names in
41013 order to perform strict type checking of the resulting parser.

41014 Alternatively, given that at least one <tag> construct is used, the union can be declared in a
41015 header file (which shall be included in the declarations section by using an **#include**
41016 construct within **{** and **}**), and a **typedef** used to define the symbol **YYSTYPE** to
41017 represent this union. The effect of **%union** is to provide the declaration of **YYSTYPE** directly
41018 from the *yacc* input.

41019 %{ ... %}
41020 C-language declarations and definitions can appear in the declarations section, enclosed by
41021 these marks. These statements shall be copied into the code file, and have global scope
41022 within it so that they can be used in the rules and program sections.

41023 The application shall ensure that the declarations section is terminated by the token **%%**.

41024 **Grammar Rules in yacc**

41025 The rules section defines the context-free grammar to be accepted by the function *yacc* generates,
41026 and associates with those rules C-language actions and additional precedence information. The
41027 grammar is described below, and a formal definition follows.

41028 The rules section is comprised of one or more grammar rules. A grammar rule has the form:

41029 A : BODY ;

41030 The symbol **A** represents a non-terminal name, and **BODY** represents a sequence of zero or
41031 more *names*, *literals*, and *semantic actions* that can then be followed by optional *precedence rules*.
41032 Only the names and literals participate in the formation of the grammar; the semantic actions
41033 and precedence rules are used in other ways. The colon and the semicolon are *yacc* punctuation.

41034 If there are several successive grammar rules with the same left-hand side, the vertical bar ' | '
 41035 can be used to avoid rewriting the left-hand side; in this case the semicolon appears only after
 41036 the last rule. The BODY part can be empty (or empty of names and literals) to indicate that the
 41037 non-terminal symbol matches the empty string.

41038 The yacc utility assigns a unique number to each rule. Rules using the vertical bar notation are
 41039 distinct rules. The number assigned to the rule appears in the description file.

41040 The elements comprising a BODY are:

41041 *name, literal*

41042 These form the rules of the grammar: *name* is either a *token* or a *non-terminal*; *literal*
 41043 stands for itself (less the lexically required quotation marks).

41044 *semantic action*

41045 With each grammar rule, the user can associate actions to be performed each time
 41046 the rule is recognized in the input process. (Note that the word "action" can also
 41047 refer to the actions of the parser—shift, reduce, and so on.)

41048 These actions can return values and can obtain the values returned by previous
 41049 actions. These values are kept in objects of type YYSTYPE (see %union). The
 41050 result value of the action shall be kept on the parse stack with the left-hand side of
 41051 the rule, to be accessed by other reductions as part of their right-hand side. By
 41052 using the <tag> information provided in the declarations section, the code
 41053 generated by yacc can be strictly type checked and contain arbitrary information. In
 41054 addition, the lexical analyzer can provide the same kinds of values for tokens, if
 41055 desired.

41056 An action is an arbitrary C statement and as such can do input or output, call
 41057 subprograms and alter external variables. An action is one or more C statements
 41058 enclosed in curly braces ' { ' and ' } '.

41059 Certain pseudo-variables can be used in the action. These are macros for access to
 41060 data structures known internally to yacc.

41061 **\$\$** The value of the action can be set by assigning it to **\$\$**. If type
 41062 checking is enabled and the type of the value to be assigned cannot
 41063 be determined, a diagnostic message may be generated.

41064 **\$number** This refers to the value returned by the component specified by the
 41065 token *number* in the right side of a rule, reading from left to right;
 41066 *number* can be zero or negative. If it is, it refers to the data associated
 41067 with the name on the parser's stack preceding the leftmost symbol of
 41068 the current rule. (That is, "\$0" refers to the name immediately
 41069 preceding the leftmost name in the current rule, to be found on the
 41070 parser's stack and "\$-1" refers to the symbol to *its* left.) If *number*
 41071 refers to an element past the current point in the rule, or beyond the
 41072 bottom of the stack, the result is undefined. If type checking is
 41073 enabled and the type of the value to be assigned cannot be
 41074 determined, a diagnostic message may be generated.

41075 **\$(tag)number**

41076 These correspond exactly to the corresponding symbols without the
 41077 *tag* inclusion, but allow for strict type checking (and preclude
 41078 unwanted type conversions). The effect is that the macro is expanded
 41079 to use *tag* to select an element from the YYSTYPE union (using
 41080 *dataname.tag*). This is particularly useful if *number* is not positive.


```

41126      {...}          This indicates C-language source code, with the possible inclusion of '$'
41127                        macros as discussed previously.

41128      /* Grammar for the input to yacc. */
41129      /* Basic entries. */
41130      /* The following are recognized by the lexical analyzer. */

41131      %token    IDENTIFIER    /* Includes identifiers and literals */
41132      %token    C_IDENTIFIER  /* identifier (but not literal)
41133                        followed by a :. */
41134      %token    NUMBER        /* [0-9][0-9]* */

41135      /* Reserved words : %type=>TYPE %left=>LEFT, and so on */

41136      %token    LEFT RIGHT NONASSOC TOKEN PREC TYPE START UNION

41137      %token    MARK          /* The %% mark. */
41138      %token    LCURL        /* The %{ mark. */
41139      %token    RCURL        /* The }% mark. */

```

41140 **Notes to Reviewers**

```

41141      This section with side shading will not appear in the final copy. - Ed.

41142      D3, XCU, ERN 293: An interpretation has been filed against 1003.2 and is likely to change "%}" to
41143      "%}.".

41144      /* 8-bit character literals stand for themselves; */
41145      /* tokens have to be defined for multi-byte characters. */

41146      %start    spec

41147      %%

41148      spec    : defs MARK rules tail
41149              ;
41150      tail    : MARK
41151              {
41152              /* In this action, set up the rest of the file. */
41153              }
41154              | /* Empty; the second MARK is optional. */
41155              ;
41156      defs    : /* Empty. */
41157              | defs def
41158              ;
41159      def     : START IDENTIFIER
41160              | UNION
41161              {
41162              /* Copy union definition to output. */
41163              }
41164              | LCURL
41165              {
41166              /* Copy C code to output file. */
41167              }
41168              | RCURL
41169              | rword tag nlist
41170              ;
41171      rword   : TOKEN

```

```

41172         | LEFT
41173         | RIGHT
41174         | NONASSOC
41175         | TYPE
41176         ;
41177     tag   : /* Empty: union tag ID optional. */
41178         | '<' IDENTIFIER '>'
41179         ;
41180     nlist : nmno
41181         | nlist nmno
41182         ;
41183     nmno  : IDENTIFIER          /* Note: literal invalid with % type. */
41184         | IDENTIFIER NUMBER    /* Note: invalid with % type. */
41185         ;

41186     /* Rule section */

41187     rules : C_IDENTIFIER rbody prec
41188         | rules rule
41189         ;
41190     rule  : C_IDENTIFIER rbody prec
41191         | '|' rbody prec
41192         ;
41193     rbody : /* empty */
41194         | rbody IDENTIFIER
41195         | rbody act
41196         ;
41197     act   : '{'
41198         | {
41199             /* Copy action, translate $$, and so on. */
41200         }
41201         | '}'
41202         ;
41203     prec  : /* Empty */
41204         | PREC IDENTIFIER
41205         | PREC IDENTIFIER act
41206         | prec ';'
41207         ;

```

41208 Conflicts

41209 The parser produced for an input grammar may contain states in which conflicts occur. The
41210 conflicts occur because the grammar is not LALR(1). An ambiguous grammar always contains at
41211 least one LALR(1) conflict. The yacc utility shall resolve all conflicts, using either default rules or
41212 user-specified precedence rules.

41213 Conflicts are either shift/reduce conflicts or reduce/reduce conflicts. A shift/reduce conflict is
41214 where, for a given state and lookahead symbol, both a shift action and a reduce action are
41215 possible. A reduce/reduce conflict is where, for a given state and lookahead symbol, reductions
41216 by two different rules are possible.

41217 The rules below describe how to specify what actions to take when a conflict occurs. Not all
41218 shift/reduce conflicts can be successfully resolved this way because the conflict may be due to
41219 something other than ambiguity, so incautious use of these facilities can cause the language

41220 accepted by the parser to be much different from that which was intended. The description file
 41221 shall contain sufficient information to understand the cause of the conflict. Where ambiguity is
 41222 the reason either the default or explicit rules should be adequate to produce a working parser.

41223 The declared precedences and associativities (see **Declarations Section** (on page 3280)) are used
 41224 to resolve parsing conflicts as follows:

- 41225 1. A precedence and associativity is associated with each grammar rule; it is the precedence
 41226 and associativity of the last token or literal in the body of the rule. If the **%prec** keyword is
 41227 used, it overrides this default. Some grammar rules might not have both precedence and
 41228 associativity.
- 41229 2. If there is a shift/reduce conflict, and both the grammar rule and the input symbol have
 41230 precedence and associativity associated with them, then the conflict is resolved in favor of
 41231 the action (shift or reduce) associated with the higher precedence. If the precedences are
 41232 the same, then the associativity is used; left associative implies reduce, right associative
 41233 implies shift, and non-associative implies an error in the string being parsed.
- 41234 3. When there is a shift/reduce conflict that cannot be resolved by rule 2, the shift is done.
 41235 Conflicts resolved this way are counted in the diagnostic output described in **Error**
 41236 **Handling**.
- 41237 4. When there is a reduce/reduce conflict, a reduction is done by the grammar rule that
 41238 occurs earlier in the input sequence. Conflicts resolved this way are counted in the
 41239 diagnostic output described in **Error Handling**.

41240 Conflicts resolved by precedence or associativity shall not be counted in the shift/reduce and
 41241 reduce/reduce conflicts reported by yacc on either standard error or in the description file.

41242 **Error Handling**

41243 The token **error** shall be reserved for error handling. The name **error** can be used in grammar
 41244 rules. It indicates places where the parser can recover from a syntax error. The default value of
 41245 **error** shall be 256. Its value can be changed using a **%token** declaration. The lexical analyzer
 41246 should not return the value of **error**.

41247 The parser shall detect a syntax error when it is in a state where the action associated with the
 41248 lookahead symbol is **error**. A semantic action can cause the parser to initiate error handling by
 41249 executing the macro YYERROR. When YYERROR is executed, the semantic action passes
 41250 control back to the parser. YYERROR cannot be used outside of semantic actions.

41251 When the parser detects a syntax error, it normally calls *yyerror* with the character string
 41252 "syntax error" as its argument. The call shall not be made if the parser is still recovering
 41253 from a previous error when the error is detected. The parser is considered to be recovering from
 41254 a previous error until the parser has shifted over at least three normal input symbols since the
 41255 last error was detected or a semantic action has executed the macro *yyerrok*. The parser shall not
 41256 call *yyerror* when YYERROR is executed.

41257 The macro function YYRECOVERING shall return 1 if a syntax error has been detected and the
 41258 parser has not yet fully recovered from it. Otherwise, zero shall be returned.

41259 When a syntax error is detected by the parser, the parser shall check if a previous syntax error
 41260 has been detected. If a previous error was detected, and if no normal input symbols have been
 41261 shifted since the preceding error was detected, the parser checks if the lookahead symbol is an
 41262 endmarker (see **Interface to the Lexical Analyzer** (on page 3287)). If it is, the parser shall return
 41263 with a non-zero value. Otherwise, the lookahead symbol shall be discarded and normal parsing
 41264 shall resume.

41265 When YYERROR is executed or when the parser detects a syntax error and no previous error has
41266 been detected, or at least one normal input symbol has been shifted since the previous error was
41267 detected, the parser shall pop back one state at a time until the parse stack is empty or the
41268 current state allows a shift over **error**. If the parser empties the parse stack, it shall return with a
41269 non-zero value. Otherwise, it shall shift over **error** and then resume normal parsing. If the parser
41270 reads a lookahead symbol before the error was detected, that symbol shall still be the lookahead
41271 symbol when parsing is resumed.

41272 The macro *yyerrok* in a semantic action shall cause the parser to act as if it has fully recovered
41273 from any previous errors. The macro *yyclearin* shall cause the parser to discard the current
41274 lookahead token. If the current lookahead token has not yet been read, *yyclearin* shall have no
41275 effect.

41276 The macro YYACCEPT shall cause the parser to return with the value zero. The macro
41277 YYABORT shall cause the parser to return with a non-zero value.

41278 **Interface to the Lexical Analyzer**

41279 The *yylex* function is an integer-valued function that returns a *token number* representing the kind
41280 of token read. If there is a value associated with the token returned by *yylex* (see the discussion
41281 of *tag* above), it shall be assigned to the external variable *yyval*.

41282 If the parser and *yylex* do not agree on these token numbers, reliable communication between
41283 them cannot occur. For (one character) literals, the token is simply the numeric value of the
41284 character in the current character set. The numbers for other tokens can either be chosen by *yacc*,
41285 or chosen by the user. In either case, the **#define** construct of C is used to allow *yylex* to return
41286 these numbers symbolically. The **#define** statements are put into the code file, and the header
41287 file if that file is requested. The set of characters permitted by *yacc* in an identifier is larger than
41288 that permitted by C. Token names found to contain such characters shall not be included in the
41289 **#define** declarations.

41290 If the token numbers are chosen by *yacc*, the tokens other than literals shall be assigned numbers
41291 greater than 256, although no order is implied. A token can be explicitly assigned a number by
41292 following its first appearance in the declarations section with a number. Names and literals not
41293 defined this way retain their default definition. All token numbers assigned by *yacc* shall be
41294 unique and distinct from the token numbers used for literals and user-assigned tokens. If
41295 duplicate token numbers cause conflicts in parser generation, *yacc* shall report an error;
41296 otherwise, it is unspecified whether the token assignment is accepted or an error is reported.

41297 The end of the input is marked by a special token called the *endmarker*, which has a token
41298 number that is zero or negative. (These values are invalid for any other token.) All lexical
41299 analyzers shall return zero or negative as a token number upon reaching the end of their input. If
41300 the tokens up to, but excluding, the endmarker form a structure that matches the start symbol,
41301 the parser shall accept the input. If the endmarker is seen in any other context, it shall be
41302 considered an error.

41303 **Completing the Program**

41304 In addition to *yparse* and *yylex*, the functions *yyerror* and *main* are required to make a complete
41305 program. The application can supply *main* and *yyerror*, or those routines can be obtained from
41306 the *yacc* library.

41307 Yacc Library

41308 The following functions appear only in the *yacc* library accessible through the `-l y` operand to *cc*
41309 or *c99*; they can therefore be redefined by a portable application:

41310 `int main(void)`

41311 This function shall call *yparse* and exit with an unspecified value. Other actions within this
41312 function are unspecified.

41313 `int yyerror(const char *s)`

41314 This function shall write the NUL-terminated argument to standard error, followed by a
41315 `<newline>` character.

41316 The order of the `-l y` and `-l l` operands given to *cc* or *c99* is significant; the application shall
41317 either provide its own *main* function or ensure that `-l y` precedes `-l l`.

41318 Debugging the Parser

41319 The parser generated by *yacc* shall have diagnostic facilities in it that can be optionally enabled
41320 at either compile time or at runtime (if enabled at compile time). The compilation of the runtime
41321 debugging code is under the control of `YYDEBUG`, a preprocessor symbol. If `YYDEBUG` has a
41322 non-zero value, the debugging code shall be included. If its value is zero, the code shall not be
41323 included.

41324 In parsers where the debugging code has been included, the external `int yydebug` can be used to
41325 turn debugging on (with a non-zero value) and off (zero value) at runtime. The initial value of
41326 *yydebug* shall be zero.

41327 When `-t` is specified, the code file shall be built such that, if `YYDEBUG` is not already defined at
41328 compilation time (using the *c99* `-D YYDEBUG` option, for example), `YYDEBUG` shall be set
41329 explicitly to 1. When `-t` is not specified, the code file shall be built such that, if `YYDEBUG` is not
41330 already defined, it shall be set explicitly to zero.

41331 The format of the debugging output is unspecified but includes at least enough information to
41332 determine the shift and reduce actions, and the input symbols. It also provides information
41333 about error recovery.

41334 Algorithms

41335 The parser constructed by *yacc* implements an LALR(1) parsing algorithm as documented in the
41336 literature. It is unspecified whether the parser is table-driven or direct-coded.

41337 A parser generated by *yacc* shall never request an input symbol from *yylex* while in a state where
41338 the only actions other than the error action are reductions by a single rule.

41339 The literature of parsing theory defines these concepts.

41340 Limits

41341 The *yacc* utility may have several internal tables. The minimum maximums for these tables are
41342 shown in the following table. The exact meaning of these values is implementation-defined. The
41343 implementation shall define the relationship between these values and between them and any
41344 error messages that the implementation may generate should it run out of space for any internal
41345 structure. An implementation may combine groups of these resources into a single pool as long
41346 as the total available to the user does not fall below the sum of the sizes specified by this section.

41347

Table 4-22 Internal Limits in yacc

41348

41349

41350

41351

41352

41353

41354

41355

41356

41357

41358

41359

41360

41361

41362

41363

41364

Limit	Minimum Maximum	Description
{NTERMS}	126	Number of tokens.
{NNONTERM}	200	Number of non-terminals.
{NPROD}	300	Number of rules.
{NSTATES}	600	Number of states.
{MEMSIZE}	5 200	Length of rules. The total length, in names (tokens and non-terminals), of all the rules of the grammar. The left-hand side is counted for each rule, even if it is not explicitly repeated, as specified in Grammar Rules in yacc (on page 3281).
{ACTSIZE}	4 000	Number of actions. “Actions” here (and in the description file) refer to parser actions (shift, reduce, and so on) not to semantic actions defined in Grammar Rules in yacc (on page 3281).

41365 **EXIT STATUS**

41366 The following exit values shall be returned:

41367 0 Successful completion.

41368 >0 An error occurred.

41369 **CONSEQUENCES OF ERRORS**

41370 If any errors are encountered, the run is aborted and yacc exits with a non-zero status. Partial
 41371 code files and header files files may be produced. The summary information in the description
 41372 file always shall be produced if the `-v` flag is present.

41373 **APPLICATION USAGE**

41374 Historical implementations experience name conflicts on the names `yacc.tmp`, `yacc.acts`,
 41375 `yacc.debug`, `y.tab.c`, `y.tab.h`, and `y.output` if more than one copy of yacc is running in a single
 41376 directory at one time. The `-b` option was added to overcome this problem. The related problem
 41377 of allowing multiple yacc parsers to be placed in the same file was addressed by adding a `-p`
 41378 option to override the previously hard-coded `yy` variable prefix.

41379 The description of the `-p` option specifies the minimal set of function and variable names that
 41380 cause conflict when multiple parsers are linked together. `YYSTYPE` does not need to be changed.
 41381 Instead, the programmer can use `-b` to give the header files for different parsers different names,
 41382 and then the file with the `yylex` for a given parser can include the header for that parser. Names
 41383 such as `yyclearerr` do not need to be changed because they are used only in the actions; they do
 41384 not have linkage. It is possible that an implementation has other names, either internal ones for
 41385 implementing things such as `yyclearerr`, or providing non-standard features that it wants to
 41386 change with `-p`.

41387 Unary operators that are the same token as a binary operator in general need their precedence
 41388 adjusted. This is handled by the `%prec` advisory symbol associated with the particular grammar
 41389 rule defining that unary operator. (See **Grammar Rules in yacc** (on page 3281).) Applications
 41390 are not required to use this operator for unary operators, but the grammars that do not require it
 41391 are rare.

41392 **EXAMPLES**

41393 Access to the *yacc* library is obtained with library search operands to *cc* or *c99*. To use the *yacc* library *main*:

```
41395 c99 y.tab.c -l y
```

41396 Both the *lex* library and the *yacc* library contain *main*. To access the *yacc main*:

```
41397 c99 y.tab.c lex.yy.c -l y -l l
```

41398 This ensures that the *yacc* library is searched first, so that its *main* is used.

41399 The historical *yacc* libraries have contained two simple functions that are normally coded by the application programmer. These library functions are similar to the following code:

```
41401 #include <locale.h>
41402 int main(void)
41403 {
41404     extern int yyparse();
41405     setlocale(LC_ALL, "");
41406     /* If the following parser is one created by lex, the
41407        application must be careful to ensure that LC_CTYPE
41408        and LC_COLLATE are set to the POSIX locale. */
41409     (void) yyparse();
41410     return (0);
41411 }
41412 #include <stdio.h>
41413 int yyerror(const char *msg)
41414 {
41415     (void) fprintf(stderr, "%s\n", msg);
41416     return (0);
41417 }
```

41418 **RATIONALE**

41419 The references in **Referenced Documents** (on page xv) may be helpful in constructing the parser generator. The referenced DeRemer and Pennello article (along with the works it references) describes a technique to generate parsers that conform to this volume of IEEE Std. 1003.1-200x. Work in this area continues to be done, so implementors should consult current literature before doing any new implementations. The original Knuth article is the theoretical basis for this kind of parser, but the tables it generates are impractically large for reasonable grammars and should not be used. The “equivalent to” wording is intentional to assure that the best tables that are LALR(1) can be generated.

41427 There has been confusion between the class of grammars, the algorithms needed to generate parsers, and the algorithms needed to parse the languages. They are all reasonably orthogonal. In particular, a parser generator that accepts the full range of LR(1) grammars need not generate a table any more complex than one that accepts SLR(1) (a relatively weak class of LR grammars) for a grammar that happens to be SLR(1). Such an implementation need not recognize the case, either; table compression can yield the SLR(1) table (or one even smaller than that) without recognizing that the grammar is SLR(1). The speed of an LR(1) parser for any class is dependent more upon the table representation and compression (or the code generation if a direct parser is generated) than upon the class of grammar that the table generator handles.

41436 The speed of the parser generator is somewhat dependent upon the class of grammar it handles. However, the original Knuth article algorithms for constructing LR parsers was judged by its

41438 author to be impractically slow at that time. Although full LR is more complex than LALR(1), as
41439 computer speeds and algorithms improve, the difference (in terms of acceptable wall-clock
41440 execution time) is becoming less significant.

41441 Potential authors are cautioned that the referenced DeRemer and Pennello article previously
41442 cited identifies a bug (an over-simplification of the computation of LALR(1) lookahead sets) in
41443 some of the LALR(1) algorithm statements that preceded it to publication. They should take the
41444 time to seek out that paper, as well as current relevant work, particularly Aho's.

41445 The **-b** option was added to provide a portable method for permitting *yacc* to work on multiple
41446 separate parsers in the same directory. If a directory contains more than one *yacc* grammar, and
41447 both grammars are constructed at the same time (by, for example, a parallel *make* program),
41448 conflict results. While the solution is not historical practice, it corrects a known deficiency in
41449 historical implementations. Corresponding changes were made to all sections that referenced
41450 the file names **y.tab.c** (now "the code file"), **y.tab.h** (now "the header file"), and **y.output** (now
41451 "the description file").

41452 The grammar for *yacc* input is based on System V documentation. The textual description shows
41453 there that the `' ; '` is required at the end of the rule. The grammar and the implementation do not
41454 require this. (The use of **C_IDENTIFIER** causes a reduce to occur in the right place.)

41455 Also, in that implementation, the constructs such as **%token** can be terminated by a semicolon,
41456 but this is not permitted by the grammar. The keywords such as **%token** can also appear in
41457 uppercase, which is again not discussed. In most places where `'%'` is used, `'\'` can be
41458 substituted, and there are alternate spellings for some of the symbols (for example, **%LEFT** can
41459 be `"%<"` or even `"\<"`).

41460 Historically, `<tag>` can contain any characters except `'>'`, including white space, in the
41461 implementation. However, since the *tag* must reference a ISO C standard union member, in
41462 practice conforming implementations need to support only the set of characters for ISO C
41463 standard identifiers in this context.

41464 Some historical implementations are known to accept actions that are terminated by a period.
41465 Historical implementations often allow `'$'` in names. A conforming implementation does not
41466 need to support either of these behaviors.

41467 Deciding when to use **%prec** illustrates the difficulty in specifying the behavior of *yacc*. There
41468 may be situations in which the *grammar* is not, strictly speaking, in error, and yet *yacc* cannot
41469 interpret it unambiguously. The resolution of ambiguities in the grammar can in many instances
41470 be resolved by providing additional information, such as using **%type** or **%union** declarations. It
41471 is often easier and it usually yields a smaller parser to take this alternative when it is
41472 appropriate.

41473 The size and execution time of a program produced without the runtime debugging code is
41474 usually smaller and slightly faster in historical implementations.

41475 Statistics messages from several historical implementations include the following types of
41476 information:

41477 *n*/512 terminals, *n*/300 non-terminals
 41478 *n*/600 grammar rules, *n*/1 500 states
 41479 *n* shift/reduce, *n* reduce/reduce conflicts reported
 41480 *n*/350 working sets used
 41481 Memory: states, etc. *n*/15 000, parser *n*/15 000
 41482 *n*/600 distinct lookahead sets
 41483 *n* extra closures
 41484 *n* shift entries, *n* exceptions
 41485 *n* goto entries
 41486 *n* entries saved by goto default
 41487 Optimizer space used: input *n*/15 000, output *n*/15 000
 41488 *n* table entries, *n* zero
 41489 Maximum spread: *n*, Maximum offset: *n*

41490 The report of internal tables in the description file is left implementation-defined because all
 41491 aspects of these limits are also implementation-defined. Some implementations may use
 41492 dynamic allocation techniques and have no specific limit values to report.

41493 The format of the **y.output** file is not given because specification of the format was not seen to
 41494 enhance applications portability. The listing is primarily intended to help human users
 41495 understand and debug the parser; use of **y.output** by a portable application script would be
 41496 unusual. Furthermore, implementations have not produced consistent output and no popular
 41497 format was apparent. The format selected by the implementation should be human-readable, in
 41498 addition to the requirement that it be a text file.

41499 Standard error reports are not specifically described because they are seldom of use to portable
 41500 applications and there was no reason to restrict implementations.

41501 Some implementations recognize "*= {*" as equivalent to '*{*' because it appears in historical
 41502 documentation. This construction was recognized and documented as obsolete as long ago as
 41503 1978, in the referenced *Yacc: Yet Another Compiler-Compiler*. This volume of IEEE Std. 1003.1-200x
 41504 chose to leave it as obsolete and omit it.

41505 Multi-byte characters should be recognized by the lexical analyzer and returned as tokens. They
 41506 should not be returned as multi-byte character literals. The token **error** that is used for error
 41507 recovery is normally assigned the value 256 in the historical implementation. Thus, the token
 41508 value 256, which used in many multi-byte character sets, is not available for use as the value of a
 41509 user-defined token.

41510 **FUTURE DIRECTIONS**

41511 None.

41512 **SEE ALSO**

41513 *c99*, *lex*

41514 **CHANGE HISTORY**

41515 First released in Issue 2.

41516 **Issue 4**

41517 Aligned with the ISO/IEC 9945-2: 1993 standard.

41518 **Issue 5**

41519 FUTURE DIRECTIONS section added.

41520 **Issue 6**

41521 Minor changes have been added to align with the IEEE P1003.2b draft standard.

41522 The normative text is reworded to avoid use of the term “must” for application requirements.

41523 **NAME**

41524 zcat — expand and concatenate data

41525 **SYNOPSIS**

41526 XSI zcat [*file...*]

41527

41528 **DESCRIPTION**

41529 The *zcat* utility shall write to standard output the uncompressed form of files that have been
41530 compressed using the *compress* utility. It is the equivalent of *uncompress -c*. Input files are not
41531 affected.

41532 **OPTIONS**

41533 None.

41534 **OPERANDS**

41535 The following operand shall be supported:

41536 *file* The path name of a file previously processed by the *compress* utility. If *file* already
41537 has the *.Z* suffix specified, it is used as submitted. Otherwise, the *.Z* suffix is
41538 appended to the file name prior to processing.

41539 **STDIN**

41540 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '- '.

41541 **INPUT FILES**

41542 Input files shall be compressed files that are in the format produced by the *compress* utility.

41543 **ENVIRONMENT VARIABLES**

41544 The following environment variables shall affect the execution of *zcat*:

41545 *LANG* Provide a default value for the internationalization variables that are unset or null.
41546 If *LANG* is unset or null, the corresponding value from the implementation-
41547 defined default locale shall be used. If any of the internationalization variables
41548 contains an invalid setting, the utility shall behave as if none of the variables had
41549 been defined.

41550 *LC_ALL* If set to a non-empty string value, override the values of all the other
41551 internationalization variables.

41552 *LC_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as
41553 characters (for example, single-byte as opposed to multi-byte characters in
41554 arguments).

41555 *LC_MESSAGES*
41556 Determine the locale that should be used to affect the format and contents of
41557 diagnostic messages written to standard error.

41558 *NLSPATH* Determine the location of message catalogs for the processing of *LC_MESSAGES*.

41559 **ASYNCHRONOUS EVENTS**

41560 Default.

41561 **STDOUT**

41562 The compressed files given as input shall be written on standard output in their uncompressed
41563 form.

41564 **STDERR**

41565 Used only for diagnostic messages.

41566 **OUTPUT FILES**

41567 None.

41568 **EXTENDED DESCRIPTION**

41569 None.

41570 **EXIT STATUS**

41571 The following exit values shall be returned:

41572 0 Successful completion.

41573 >0 An error occurred.

41574 **CONSEQUENCES OF ERRORS**

41575 Default.

41576 **APPLICATION USAGE**

41577 None.

41578 **EXAMPLES**

41579 None.

41580 **RATIONALE**

41581 None.

41582 **FUTURE DIRECTIONS**

41583 None.

41584 **SEE ALSO**

41585 *compress, uncompress*

41586 **CHANGE HISTORY**

41587 First released in Issue 4.

