WG14 N1380
INCITS PL22.11/09-0001

ISO/JTC1/SC22/WG14 AND INCITS PL22.11
SEPTEMBER 2008 MEETING MINUTES (DRAFT)

Meeting Location:

IBM Toronto Software Lab
8200 Warden Avenue
Markham, Ontario L6G 1C7

Host Contact information:
Edison Kwok

Meeting Host: Standards Canada, IBM Canada
Meeting Dates: 30 March - 3 April 2009
Meeting Times:
30 March 2009: 09:30-12:00 lunch 13:30-17:00
31 March 2009: 09:00-12:00 lunch 13:30-17:00
01 April 2009: 09:00-12:00 lunch 13:30-17:00
02 April 2009: 09:00-12:00 lunch 13:30-17:00
03 April 2009: 09:00-12:00

1. OPENING ACTIVITIES

1.1 Opening Comments (Benito, Kwok)

Edison Kwok welcomed us to IBM Labs. The meeting is sponsored by Standards Canada and IBM.

1.2 Introduction of Participants/Roll Call

John Benito	Blue Pilot	USA	WG14 Convener
Larry Jones	Siemens PLM Software	USA	Project Editor
David Svoboda	CMU/SEI (CERT)	USA	
Robert Secord (phone)	CMU/SEI (CERT)	USA	
Tana L. Plauger	Dinkumware, Ltd	USA	
P. J. Plauger	Dinkumware, Ltd	USA	
Edison Kwok	IBM	USA/CANADA	HOD - CANADA
Raymond Mak	IBM	USA	
David Keaton	Self	USA	
Arjun Bijanki (phone)	Microsoft	USA	
Andrey Gusev	Oracle	USA	
Barry Hedquist	Perennial	USA	Secretary
Tom Plum	Plum Hall	USA	
Bill Seymour	self	USA	
Clark Nelson	Intel	USA	
Randy Meyers	Silverhill Systems	USA	PL22.11 Chair
Fred Tydeman	Tydeman Consulting	USA	
Hans Boehm (phone)	Hewlett Packard	USA	

Blaine Garst	Apple	USA	
Douglas Walls	Sun Microsystems	USA	HOD - USA
Jim Thomas (phone)	Hewlett Packard	USA	
Hugh Redelmeier	Self	CANADA	
Rajan Bhakta	IBM	CANADA	
Walter Banks	Bytecraft	CANADA	
Ettoer Tiotto	IBM	CANADA	
Calin Cascaval	IBM	CANADA	
Kathy Yellick	UC Berkeley	USA	

1.3 Selection of Meeting Secretary

Barry Hedquist is Meeting Secretary.

1.4 Procedures for this Meeting (Benito)

The Chair, John Benito, announced the procedures are as per normal. Everyone is encouraged to participate in straw polls. INCITS PL22.11 members are reminded of the requirement to follow the INCITS Anti-Trust Guidelines which can be viewed at http://www.incits.org/inatrust.htm.

All 'N' document numbers in these minutes refer to JTC1 SC22/WG14 documents unless otherwise noted.

Straw polls are an informal mechanism used to determine if there is consensus within the meeting to pursue a particular technical approach or even drop a matter for lack of consensus. Participation by everyone is encouraged to allow for a discussion of diverse technical approaches. Straw polls are not formal votes, and do not in any way represent any National Body position. National Body positions are only established in accordance with the procedures established within and by each National Body.

The term "WP" means the latest draft version of the revision to the C Standard, also known as C1X.

1.5 Identification of National Bodies (Benito)

US, Canada

1.6 Identification of PL22.11 Voting Members (Tydeman)

See List above. 10 of 14 possible are present. HP is on-line (teleconference)
Voted to reestablish Voting Rights for HP, and CMU/SCI CERT (Benito, Tydeman)
HP (Benito, Tydeman) 11-0-0 APPROVED
CMU/SEI (Benito, Tydeman) 12-0-0 APPROVED (HP added to voting)

1.7 Approval of Previous Minutes (N1346) (Hedquist)

Several comments for typos, etc. Minutes approved as modified. Nxxxx

1.8 Review of Action Items and Resolutions (Hedquist)

ACTION: David Keaton and Clark Nelson to produce words for Rationale w.r.t. Sequence Points and the Sequenced Before relationship.
DROP

ACTION: Keith to develop attribute placement requirements as a part of a template for attributes.
DROP

ACTION: Clark, Mark and Nick to write detailed attribute syntax proposal, focused on existing practice and the current list of attributes intended to be standardized.
REASSIGN – David Svoboda

ACTION: Tom Plum to develop a fully-formed proposal for try-finally.
C++ did not adopt the finally clause, and Tom does not see a way to go forward w/o C++ adoption. Tom asked to be relieve of this item.
DROP

ACTION: David Keaton to rework his previous proposal on anonymous unions, adding anonymous structures as appropriate. OPEN – in progress

ACTION: Convenor to produce a “Technical Corrigendum” document describing how to apply all as yet unapplied defect resolutions to the C1x working paper. N1359 DONE

ACTION: Randy to write examples for both DR 340 and 342 to allow vendors to evaluate what their implementations actually do in these cases. OPEN

ACTION: Rich to write a fully-formed proposal on benign re-type def'ing.
DONE N1360

ACTION: EDITOR update the sequence point annex in the WP.
DONE

ACTION: Convenor to forward, N1337 DTR 24731-2, to SC22 for PDTR Ballot.

DONE. Ballot Passed. Awaiting complete document. Editorial comments from Canada.

ACTION: Tom to produce revision to N1331 based on the discussions held in this session.
DONE N1350

ACTION: PJ to review c16rtomb in TR19769 (N1326) and add clarification if needed.
DONE N1373

ACTION: Nick to solicit a paper from Austin Group members with proposed words for suggested changes to N1325.
DROP

ACTION: Clark to revise N1329 to make it implementation defined for an implementation to support access a thread_local object from more than one thread, and to change the spelling of "thread_local" to "_Thread_local".

DONE N1364

ACTION: Ulrich and Nick to put together a paper on an alternate approach to implementing type-generic macros.
OPEN – PJ to pick up.
ACTION - PJ to put together a paper on an alternate approach to implementing type-generic macros.

ACTION: Nick to turn the concept of a reserved name for STDC into a paper.
DONE (N1345).

ACTION: Fred to rewrite N1317 for C++ capability.
DONE N1352

ACTION: Clark to revise N1284 for a review committee.
DONE

ACTION: Convenor to organize the mid-meeting in 6.0.1.
DONE

ACTION: Tom and Arjun revise N1332, proposal for new library APIs for EncodePointer() and DecodePointer().
Tom proposed a different approach to this. Add to agenda. CLOSED

ACTION: Arjun to revise N1335 to add alignof
OPEN

1.8 Approval of Agenda (N1366)

Revisions to Agenda:
Added items:

4.24 Adding EPOLE to math Library Functions
4.25 Namespace Reservation for the C Standard
4.26 Encoding and Decoding Function Pointers
4.27 Unicode and Raw String Literals

Agenda approved as modified.

1.9 Information on Future Meetings and Mailings.

1.9.1 Future Meeting Schedule

26-30 October, 2009. Santa Cruz, CA USA (N1368)

Room rate will stay the same over the weekend ($79) for those attend the C++ Meeting.

Fall 2009 - Santa Cruz, CA, Plantronics / ANSI host.

There are no meeting hosts for 2010. We would like to get a host in Europe for March/April time frame. General plan is to have two meetings per year, with a possible teleconference in between. *Italy (University of Florence) is interested in hosting a meeting in 2010, as is the*

PL22.11 member CERT, and there is also another PL22.11 member that has shown interest in hosting.

Looking at Florence Italy early 2010.

1.9.2 Future Mailing Schedule

Post Markham mailing: 2009-05-01
Pre Santa Cruz mailing: 2009-09-28

2. LIAISON ACTIVITIES

2.1 WG14 / PL22.11 (Benito, Walls, Meyers)

PL22 meeting dismissed any notion that the C committee would be left powerless, but almost 1/3 of the committee is made up of PL22.11 members.

Decimal FP TR 24732 has been submitted for publication.

Only TR24731-2 is open.

Major work item now is the C1x revision.

2.2 WG21 / PL22.16

In the memory model, WG14 wanted to make the initial state of an atomic_flag indeterminate if it isn't initialized with ATOMIC_FLAG_INIT. WG21 would now like to make the same change, but this topic wasn't the subject of any NB comment. For the sake of having a paper trail on this, WG21 would like WG14 to make this the subject of an official liaison statement to WG21.

ACTION – Clark to write a liaison paper to WG21 regarding the making the initial state of an atomic_flag indeterminate if it isn't initialized with ATOMIC_FLAG_INIT.
DONE

WG21 is in ballot resolution for the CD ballot, and will be for the rest of this year.

The C++ Special Math FCD is out for ballot.

2.3 Linux Foundation (Stoughton)

Nothing to report.

2.4 WG11 (Wakker)

No Report

2.5 WG23: Vulnerability (Plum)

TR balloted and approved with comments. Ballot resolution underway for the next 1-2 meetings. The US TAG for WG23 is PL 22, and several members of PL22.11 are on the TAG.

2.6 Austin Group - (Stoughton)

No report

2.7 Other Liaison Reports

3. EDITOR REPORTS

3.1 Report of the Rationale Editor (Benito)

No rationale has been developed for items being added to the WP. It is the paper author's who are responsible for writing the rationale.

3.2 Report of the Project Editor (N1363) (Jones/Stoughton)

3. Still Pending 1. N1333 Unicode and Raw String Literals This will require extensive editing to integrate with the changes from TR 19769 and to remove C++-isms. Also note that additional changes are needed to separate universal-character-name from escape-sequence in addition to those in the paper.

WP also contains a list of papers applied to WP. Change bars are from release to release.

Headers that are NOT required for a function should NOT be listed in the synopsis. Listing them implies they are required. We want them to be consistent. Only the headers required for the function declaration shown in the synopsis, and any others required to make the declaration valid.

ACTION – Jones: Review use of headers required for the function declaration shown in the synopsis, and any others required to make the declaration valid.

Fred - Need to remove 'and constants' from 0, 1, 2 bullet items for FLT_EVAL_METHOD. [5.2.4.2.2#9].

ACTION – PE Need to remove 'and constants' from 0, 1, 2 bullet items for FLT_EVAL_METHOD. [5.2.4.2.2#9].

We probably need to find a back-up editor.

3.3 Status of TR24731, Bounds Checking (Meyers)

Nothing new to report.

3.4 Status of TR24732, Decimal Floating Point (Kwok)

TR 24732 is now published. Type 2, must be paid for.

3.5 Status of Draft IS 24747, Special Math (Plauger)

Sent to ITTF for publication as an IS.

3.6 Status of TR 24731-2 (Stoughton)

DTR Ballot passed. One issue remains – final ballot. Need final version to submit for publication to ITTF.

3.7 Status of TR18037, Embedded Processors (Wakker)

Revised. Now available at ISO web site. It is not free.

4. DOCUMENT REVIEW

4.1 Committee Draft - March 1, 2009 (N1362, N1363) (Jones)

Should the wchar_t encoding be implementation-defined if it's not ISO/IEC 10646, too?
Yes

There is a conflict between the description of **quick_exit**, which says that file buffers are not flushed, and the description of **_Exit**, which says that whether file buffers are flushed is implementation-defined.
Say nothing about what happens with quick_exit. It calls _Exit.

4.2 Parallel memory sequencing model proposal (N1349,) (Nelson)

N1349 proposes to incorporate the same memory sequencing model being adopted by C++. For rationale and explanation of the memory model overall, see N1276. For additional rationale and explanation of the design of the atomic (library) facilities, see N1284, the predecessor of this paper.

[discussion]
Paper adds the rules to specify what optimizations are or are not legal during synchronization. Bulk of the changes to this paper are editorial to fit C style.

Blaine: These are really operators, so assigning one of them to a function pointer will fail? Yes, that's true.
PJ: There are ways to deal with that problem via compiler magic.
So, you can take the address of ? Yes. These are really type generic macros similar to tgmath. This discussion is really about back-filling.

Concerns:
Blaine: atomic is not part of the type system
Clark: making the atomic operations function calls was a deliberate decision to allow a library only implementation . Throttle optimization enough to make it work. Also reflective of common practice.

Blaine: Too many optional synchronization modes – which is not common practice. Concern is that folks will choose the wrong one, tests will work fine, but errors will occur in the field. Promote practices that will lead to correct programming, rather that providing too much rope. They are hard to get right in any case. Those that need additional options already know how to do that. Use only options that are known to be correct.

Clark: This looks the way it does for consistency with C++, but this feature is not at all intended for novice programmers. They would be better served to use mutexes (another paper), than using this feature.

Tom: We initially decided to let C++ be the home for development of these features, while we watched to make sure C did not get harmed. We tried to synchronize this work at the lowest level, which make the C level the wizard level, while C++ is at a higher level.

Blaine: Simple capabilities, such as 'fetch and add' cannot be done w/o diving into the low level material in this paper. It's simply too complicated for non-guru programmers. They will dive into it, and they will make mistakes. Add underscores ??

Clark: Possibly an interesting idea.

Tom: Having a rationale available would help make it easier for newer folks to understand the how and why of where we are.

Doug: Thanked Blaine for helping him to finally understand what this proposal is about.

PJ: Rationale and tutorial needed to answer "What you do with this stuff is…. ". This is not stuff to be used to write code with, and we should say so.

Hans can point to some explanation of details that drove the overall design which might help accomplish PJ's goal.

Blaine: Do not understand the intent on the atomic_flag.
Clark: It is the one and only type that is guaranteed to be lock free.

Blaine: Imposing requirements on new hardware?
Hans: Yes – byte addressability for multi-core systems.

The features provide as much performance portability as we can provide. A cycle or two may be lost. Whether or not that matters depends on the application. W/o this, we can lose several hundred cycles. We've tried to accommodate different architectures as best as we know how.

Are all atomic types volatile? Yes. Allows you to declare a volatile atomic object, and do what you want with it. What's the implication? Some believe it important to allow atomic operations to be done in a non-volatile manner. Volatile semantics are not required. An implementation is allowed, via type generic, provide different operations.

Clark: Volatile is a yawning chasm that this discussion can fall into – maybe it already has.

Hugh: So, whatever you think you need volatile for, you probably already have? Yes.

Clark: The implementation is allowed to treat it as volatile, it is not required. It is not the intent that every access be treated as an volatile operation. It is not necessary to make the function work. See:

> "NOTE Many operations are volatile-qualified. The "volatile as device register" semantics have not changed in the standard. This qualification means that volatility is preserved when applying these operations to volatile objects. (<<ATM>> 5p3."

PJ: We are not sure whether you need to be volatile or not, but being volatile cannot hurt, so we've taken the least harm approach. There's really not a lot more to be said.

Straw Poll: Adopt N1349 into the WP.

David K –wants a rationale for this paper that explains how and why we got here. Hans is thinking about doing this work. David wants the words in the Straw Poll. Many said no.
Walter – there's not a whole lot of experience dealing with this. Clark: the approach here is not new.
Tom: we will be sending a wrong message to C++ if we do not vote this in. If we found bugs we should fix them, but we did not. We ought to move forward.
PJ: If we want atomics in C1X, we should vote this in, and fix it up, if needed. If we don't want atomics, then we should not adopt this. Blaine still has specific technical concerns, but likes the proposal in general.

Yes – 17
No – 1
Abs – 3

DECISION: Adopt N1349 into the WP.

4.3 Thread-local Storage (<u>N1364</u>) (Nelson)

N1364 proposes a specification for thread-local storage. This feature is provided by most C compilers today in one form or another. This paper was reviewed in Milpitas, N1329, and minor changes were made to it at the committees request.

The changes to the standard necessary to add this feature to C are fairly limited and straightforward. The terminology adopted into the C++ standard is retained for consistency.

Note: The changes to C++ are more extensive mainly because they address dynamic initialization and destruction. It should be noted that allowing dynamic initialization and destruction for thread-local objects in C++ is a fairly significant extension of existing practice.

[discussion]
Should it be ID to access another thread's auto variable? YES
Hans: Major downside , impossible to have a pointer to a local variable, then call a function to it (?) Tom: Yes, that's true.
PJ: Making support for sharing auto variables ID accommodates those who are unable to do so. Does not see that a giant change.
Does everyone agree? No objections (default yes).
Tom: add to the Standard the words: "It is implementation-defined whether …", i.e. whether or not a particular feature is supported.

Should there be a feature test macro? No. If we start down this path, we'll need several. A potential slippery slope. Tom would like a #error available if it's not going to work. Something that the code can call (has to be a macro). This aspect of the approach can remain an open issue, to be addressed in a separate paper. Blaine points out that this is a general problem, and can lead to 30-40 macros.

Nothing is said about auto in the paper, so it has to work. Do we want to say that. Yes for auto, nothing about macros.

Straw Poll: Adopt N1363 with the addition of auto into the WP.
Yes – 19
No – 0
Abs - 2

DECISION: Adopt N1363 with the addition of auto into the WP.

[Thurs]
ACTION: Clark write a paper to clarify the behavior that values are computed before main starts, and thread local storage get the same values as before program start up.

4.4 Support for Conditional Inclusion of Headers (N1348) (Seymour)

N1348 proposes adding a mechanism for determining whether standard headers are available in the implementation. This can help users whose code needs to be portable to implementations of different versions of the Standard.

PJ: The Standard is not the position of telling people what to do if the implementation does not conform.

General discussions of the impact a feature like this would affect a preprocessor, is such a features transitional (intended for being between versions of the standard), is it intended for features of the compiler, or simply headers.

Bill would like compilers to be able to tell his code which features they have. Tom's example is freestanding, being able to identify what additional features / headers exist above that required by 'freestanding'.

Bill wants a C99 compiler, for example, to tell him what C1X features exist.

Do we have any consensus to move forward on this, Up or down.
Straw poll: Proceed to a complete proposal ?
Yes – 4
No – 10
Abs – 5
No consensus to proceed with this.

4.5 C Language support for multiprocessor application environments (N1351) (Banks)

N1351 is a general introduction to language issues in supporting multiprocessor execution environments. Two basic questions are asked:

1) Should the mainstream language support a multiprocessor environment? **(Currently N1256 Programming languages — C ISO/IEC 9899:TC3)**

2) A separate but related issue is should the embedded systems technical report be enhanced to support named execution spaces to reflect current embedded systems practices? **(ISO/IEC TR 18037 Programming languages - C - Extensions to support embedded processors)**

Key: Multiprocessor *applications*, not simply multiprocessor systems. Sees quite a bit of this in the automotive industry where several things are going on at once. Broader questions of whether or not C will support functions of across processors?
What's different from multithreaded libraries? It’s a different approach. Things happen asynchronously.
What's needed?

Clark does not see a difference between what we've been doing with memory model, and what Walter is looking for. We have not done work using heterogeneous processors, that's one area that applies. Do we want to go down that road?
Inter-threaded communication libraries? That one approach that has not been used.
Threads communicating through memory. Threads were rejected because they do not have a view of parallel control processing.
It's difficult to wrap our heads around this because we need to bridge a terminology gap that exists between what Walter is looking for, and what we are already doing.

In general, the answer to the first question is yes, but we need to better understand the terminology used on both sides to determine what's needed.

The second question really involves a mechanism to identify a named execution space so that two or more execution spaces can be called directly by a function. This could be an Amendment to the existing TR.

ACTION: Convenor to find out from SC22 what we can do / not do with the existing TR, then look at how to proceed from there w/r/t N1351 and possible changes to TR18037.

4.6 Proposal for an Annex to C1X (N1350) (Plum)

N1350 is a proposal to add an annex to C1X that addresses issues that have come out of the work of SC22/WG23, Programming Language Vulnerabilities. There are four proposals in the paper:

1. Add two definitions to the main body of C1X (trap representation, to perform a trap).
2. Add an informative Annex to C1X for solutions to various C vulnerabilities.
3. Add four definitions to the new annex
    - to perform a trap
    - out-of-bounds-store
    - bounded undefined behavior
    - critical undefined behavior
4. Add the contents of TR24731-1 to the new annex.

[Discussion]
Proposal #4
Focus initially is #4, as an informative new annex. Should we do at least this much? Tom would like to see this as a conditionally normative annex, i.e. if you support this, do it this way, however support is optional. There has been some NB feedback to not impose a lot of new requirements in C1X. David K believes it should be merged in-line.

TR 24731-1 was put together as an approach to retrofit existing code, and considered by some as not the optimal approach for new code. Hence we started TR24734-2, which has been through DTR ballot, and for which we are waiting final text.

If we put too much in the Standard. we dilute the entire Standard. Complex is a likely example.

Lots of discussion, but general support for a conditionally-normative annex. Tom considers such a friendly amendment.

Straw Poll: Support for adding TR24731-1 as a conditionally normative Annex to C1X.
Yes – 16
No – 2
Abs – 0

DECISION: Add TR 24731-1 as a conditionally normative Annex to C1X.

Proposal #1
Adds the following new definitions for terms that are not defined in the definitions, or for terms that are defined in-line in the Standard, but not in the definitions.

trap representation

to perform a trap

Do these definitions conflict with the ISO Vocabulary standard?
ACTION: Tom Plum to check that the proposed definitions in N1350 do not conflict with the ISO Vocabulary standard.

We do not want to constrain what implementations do when a trap occurs.

Straw Poll: Adopt definitions for '*trap representation'* and *' to perform a trap'* in N1350 to C1X.
Yes - 15
No - 1
Abs – 3
DECISION: Adopt definitions for '*trap representation'* and *' to perform a trap'* in Proposal #1, N1350 to C1X.

[Tues – PM]
Proposal #2:
Add an informative annex to C1x for solutions to various C vulnerabilities (called "Annex K" in this paper), with a tentative title such as "Analyzability".

Proposal #3:
Provides a number of definitions, and a table of undefined behavior reclassified (elevated) as critical undefined behavior:

N.1 *to perform a trap*
As defined at 3.17.5, "to perform a trap" means to interrupt execution of the program such that no further operations are performed. However, in this Annex K, this interruption is permitted to take place by invoking a runtime-constraint handler. Any such semantics are implementation-defined.

N.2 *out-of-bounds store*
an (attempted) access (3.4.1) which, at run-time, for a given computational state, would modify one or more bytes (or for an object declared volatile, would fetch one or more bytes) that lie outside the bounds permitted by the standard

N.3 *bounded undefined behavior*
behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this International Standard imposes no requirements, except that the behavior shall not perform an out-of-bounds store, and that all values produced or stored are indeterminate values.
NOTE The behavior might perform a trap.

N.4 *critical undefined behavior*
behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this International Standard imposes no requirements
NOTE the behavior might perform an out-of-bounds store, or might perform a trap

All undefined behaviors defined in clauses 1 through 7 of this International Standard are (in this Annex K) restricted to the semantics of bounded undefined behavior, except for the following critical undefined behaviors:

6.2.4	An object is referred to outside of its lifetime
6.3.2.1	An lvalue does not designate an object when evaluated
6.5.3.2	The operand of the unary * operator has an invalid value
6.5.6	Addition or subtraction of a pointer into, or just beyond, an array object and an integer type produces a result that points just beyond the array object and is used as the operand of a unary * operator that is evaluated
7.20.3	The value of a pointer that refers to space deallocated by a call to the free or realloc function is used
7.21.1, 7.24.4	A string or wide string utility function is instructed to access an array beyond the end of an object

PJ: Trouble enumerating between things that are bad with everything that is really bad. Believes there should be additional Undefined Behavior on this list.
Clark: This is really a basis for work to improve the reliability of software.
Tom: This work came from the work OWGV / WG23, and is directly applicable to C. The belief is that WG14 is the correct group to vet this material.

PJ: Does not believe it belongs in the C Standard, even as an informative annex. It's an interesting work in progress.

Straw Poll: Adopt proposals #2 and #3 in N1350 to C1X.
Yes - 9
No - 10
Abs – 3
No consensus to add this to the Standard at this time

Tom plans to bring this to the next meeting, and try to convince the committee to adopt it. There are other forms this work can take, TR, position paper, but additional work is needed.
Randy: This is still in a fairly rough state, but in some sense there is still a lot of work to be done. The problem should continue to be addressed.

4.7 Extensions to the C1X Library (N1358) (Svoboda)

N1358 proposes a number of extensions to the C1X library to enhance security in the C language.

1. **fopen() exclusive access with "x"** - adds a mode 'x' that will cause **fopen()** to fail rather than open a file that already exists. This is necessary to eliminate a time-of-creation to time-of-use (TOCTOU) race condition vulnerability.

   The ISO/IEC 9899-1999 C standard function **fopen()** is typically used to open an existing file or create a new one. However, **fopen()** does not indicate if an existing file has been opened for writing or a new file has been created. This may lead to a program overwriting or accessing an unintended file.

The GNU C library has implemented this feature.

Proposed Changes:

Section 7.19.5.3 (The **fopen** function), paragraph 3 would receive the following additions:

wx	create text file for writing with exclusive access
wbx	create binary file for writing with exclusive access
w+x	create text file for update with exclusive access
w+bx *or* **wb+x**	create binary file for update with exclusive access

The following paragraph should be inserted before paragraph 5:

> Opening a file with exclusive mode (**'x'** as the last character in the **mode** argument) fails if the file already exists or cannot be created. Otherwise, the file endeavors to create the file with exclusive (also known as nonshared) access to the extent that the underlying system supports exclusive access.

'w' also means truncating existing, does that get lost? Yes, with wx you are creating a new file.

Straw Poll: Adopt #1 to the WP

Yes – 17

No – 0

Abs – 2

DECISION: Adopt proposal #1 in N1358, as above to the WP.

2. **fopen_s() exclusive access with "x"** - similar to above for TR24731-1 function fopen_s().

We chose to defer discussion on #2 until we get to Toms paper on TR24731-1, N 1350.

Straw Poll: Using 'endeavors to' add fopen_s to TR24731-1 as Annex to C1X.

Yes – 16

No – 0

Abs – 3

DECISION: Using 'endeavors to' add fopen_s to TR24731-1 as Annex to C1X.

3. **rand() disclaimer** - random number generators are not genuinely random, and rand produces numbers that may be predictable. At the last WG14 meeting in Milpitas, we proposed adding the **random()** function, as specified by POSIX, to the C standard. The WG14 committee was unfavorable to this proposal. They agreed that **rand()** was a poor random-number generator (RNG), but experts did not agree on what constitutes a 'good' RNG. They suggested that CERT should draft a disclaimer for **rand()** for inclusion in the standard. We hereby provide such a disclaimer:

Add a footnote to 7.20.2.1, paragraph 4, which reads:

These specifications for a pseudo-random sequence generator do not guarantee that the numbers generated are sufficiently random for applications with strict randomness requirements, such as cryptographic applications. There are several implementations of

rand() which are known to produce insufficient results. For instance, the low-order bits may follow a short cycle. Specifications for proper pseudo-random number generation are beyond the scope of this document.

This is a potential slippery slope, and implies that all implementations of rand are wrong, or insufficient to use. Do we want to go down that slope? Some believe we should, others don't. There is no requirement that rand() be of high quality. This is really a QOI item. Should CERT go back and define what is really needed to meet certain users?

Straw Poll
Adopt something along the line of Proposal #3, rand() disclaimer, N1358, to the WP.
Yes – 10
No – 4
Abs – 5
DECISION: Adopt something along the lines of Proposal #3, rand() disclaimer, in N1358, to the WP.

4. **resetenv() to clean the environment** - in C99 the method for altering the environment list is implementation defined. Because environment variables are inherited from the parent process when a program is executed, an attacker can easily sabotage variables, causing a program to behave in an unexpected and insecure manner. The purpose of this function is to 'sanitize' the environment by resetting it to a certain default state, making it less vulnerable to attack.

Proposed Change:
Add this section after 7.20.4.5 (The **getenv** Function) *7.20.4.6 The **resetenv** function*

> *Synopsis*
> #include <stdlib.h>
> void resetenv();
>
> *Description*
> The **resetenv** function endeavors to replace the current host environment with a minimal host environment. This minimal environment is independent of the environment state before the call to **resetenv**, and is a suitable environment for a subsequent call to **system**. The set of names and values in the minimal environment and the method for altering the environment list are implementation-defined.
> The implementation shall behave as if no library function calls the **resetenv** function.
>
> *Returns*
> The **resetenv** function returns no value.

Discussion
Add 'endeavors' to the description, i.e. …resetenv endeavors to replace … Not clear how this can be implemented, and it's not clear whether it belongs in the Standard.
PJ: This does not have functional cohesion.
Larry: We don't give you a way to set the environment, why do we want to provide a way to reset it?

The call to 'system' referred to above, is not a C call. It's specific to Windows.
Change 'minimal environment' to 'default' ?

PJ: Better approach is to not trust environment variables, and clean them up as needed. Not Standard material. The promise of the function is too vague.

Straw Poll
Adopt Proposal #4, as modified, resetenv, N1358, to the WP.
Yes – 3
No – 12
Abs – 4
No Consensus - Stop

5. **memset_s() to clear memory, without fear of removal** - Use of the C99 function memset() is often used to zero out a series of bytes. However it's use can be insufficient in cases where an optimizing compiler employs 'dead store removal' whereby the call to memset() is optimized away, leaving critical data exposed to discovery. A new function, memset_s, would guarantee that such a call to it would not be optimized away.

Not in TR24731.

Proposed Change:

Add the following section after section 7.21.6.1 *The* ***memset()*** *function*:

7.21.6.2 The **memset_s** function

Synopsis

```
#include <string.h>
void *memset_s(void *s, int c, size_t n);
```

Description

> The **memset_s** function copies the value of **c** (converted to an **unsigned char**) into each of the first **n** characters of the object pointed to by **s**. Unlike **memset**, any call to **memset_s** shall be evaluated strictly according to the rules of the abstract machine, as described in 5.1.2.3. That is to say, any call to **memset_s** shall assume that the memory indicated by **s** and **n** may be accessible in the future and therefore must contain the values indicated by **c**.

Returns

> The **memset_s** function returns the value of **s**.

The paper notes that this function may not be sufficient when working with sensitive information.
[Discussion – Tue AM]
Are we in the business of providing secure memory? Why not add a footnote similar to the one for rand?

Existing implementations? CERT provides a solution via a code example in the paper freely available for anyone to use.

Blaine believes there are provisions in the Standard that will allow a user to this.
Randy in favor of example 5 function.
David would like an argument added in congruence with TR24741. Actually reordering the arguments works.
Clark would prefer to see something like this added to POSIC rather than the C Standard.
Tom believes the commercial impact is greater in Windows that POSIX, but is willing to wait to see where TR24731-1 goes. May want to add this function to the TR.
David S is OK with that suggestion.
David K: reorder the argument, switch the second and third arguments for other _s consistency.
Blaine: Now it differs from memset. It should be consistent.

ACTION: David Svoboda to come up with words we can agree with for memset_s.
DONE

[Thurs]
Reviewed new words from David. The synopsis is consistent with other materials in 24731-1. New paper – N1381

Straw Poll: Adopt the new words in N1381 to WP.
Yes – 17
No – 0
Abs – 3
DECISION: Adopt N1381 to the WP.

4.8 Benign typedef redefinition (N1360) (Thomas)

C++ allows a typedef redefinition with the same name as a previous typedef to appear in the same scope, as long as it names the same type. Some C compilers allow similar typedef redefinition as an extension, though C99 does not allow it. Adding benign typedef redefinition to C1x would enhance consistency with C++, standardize some existing practice, and safely eliminate a constraint that is unhelpful and an occasional nuisance to users.

Recommended change (to C1x draft N1336): Change 6.7 #3 from:

If an identifier has no linkage, there shall be no more than one declaration of the identifier (in a declarator or type specifier) with the same scope and in the same name space, except for tags as specified in 6.7.2.3.

to:

If an identifier has no linkage, there shall be no more than one declaration of the identifier (in a declarator or type specifier) with the same scope and in the same name space, except a typedef specifier can be used to redefine the name of any type declared in that scope to refer to the type to which it already refers, and except for tags as specified in 6.7.2.3. (added portion is underlined).

Tom: Thought C99 already had this. C++ has it, and he's in favor of it.

Straw Poll:
Adopt the proposed change in N1360, as shown above, to the WP.

Yes – 17
No – 1
Abs – 0
DECISION: Adopt the proposed change in N1360 to the WP.

4.9 Contractions and expression evaluation methods (N1367) (Thomas, Tydeman)

This paper notes an ambiguity in the specification of contractions in N1336 (C1X) and the effect of expression evaluation methods on them, and it proposes text changes for resolution.

Proposed Changes:
Footnote 79: (Clause 6.5#8) Insert at the beginning: “The intermediate operators in the contraction are evaluated as if to infinite precision and range, while the final operator is rounded to the format determined by the expression evaluation method.”

F.6: change to read “A contracted expression is correctly rounded (once) and treats infinities, NaNs, signed zeros, subnormals, and the rounding directions in a manner consistent with the basic arithmetic operations covered in IEC 60559.”

F.6, Recommended practice: remove the second sentence.

Straw Poll:
Adopt words to the effect of the proposed changes in N1367 to the WP.
Yes – 13
No – 0
Abs – 6
DECISION: Adopt words to the effect of the proposed changes in N1367 to the WP.

4.10 FLT_EVAL_METHOD and constants (N1361) (Thomas)

In September, 2008, the committee indicated consensus to remove const from FLT_EVAL_METHOD. This paper argues that decision should be reconsidered because doing so would change the behavior of existing C99 conforming programs.

Example:

```
#include <float.h>
#include <stdio.h>
Int main(void)
{
#if defined(__STDC_IEC_559__) && (FLT_EVAL_METHOD == 1)
        If (0.1F != (1.F / 10.F)) printf(“non-conforming implementation\n”);
#endif
}
```

The above code would print “non-conforming implementation” if the #if conditions were satisfied but the constant 0.1F were not widened to double like the divide operation.

See 4.17, N1365, for related discussion.

Proposal: Rescind the change proposed by N1321 adopted in Sept 2008. i.e. Don't remove constants from the current standard evaluation methods, namely those for FLT_EVAL_METHOD values 0, 1, or 2.

Straw Poll
Yes – 10
No – 2
Abstain – 8
DECISION: Go back to C99: Rescind the change proposed by N1321 adopted in Sept 2008. i.e. Don't remove constants from the current standard evaluation methods, namely those for FLT_EVAL_METHOD values 0, 1, or 2.

NOTE: Larry did not make exactly the correct change to the WP. Larry will move back to what is in C99.

4.11 New macros for <float.h> (N1352) (N1377) (N1378) (Tydeman)

N1352 proposes several new macros to better define the values of the smallest subnormal floating-point numbers, if they are supported.

Initially came from C++, but these names differ. Real words are needed. Intent is to close the gap between C and C++.

ACTION: Fred to proposed new words for *_MAXDIG10, etc. re: N1352.
DONE

Jim T objects to term 'subnormalization'.
Larry: Use the terms we've already defined in our floating point model.

[Wed PM]

Fred presented a response to his action item above as N1377.
Add words to the rationale saying how the reverse trip is made.

Straw Poll: Adopt N1377, replacing N1352, for incorporation into the C1X WP.
Yes – 16
No – 0
Abs -4
DECISION: Adopt N1377 for incorporation into the C1X WP.

Fred presented N1378, a reworded version of the remainder on N1352.
Why do we need this. If we look at the subnormal bits, don't they go away. Let QOI determine.
Jim thinks some rewording is necessary w/r/t what -1 and +1 really mean.

What do we do, if anything for implementations that don't know whether or not they have subnormal numbers.

ACTION: Fred and Jim will rework the words in N1378.

4.12 FLT_EVAL_METHOD issues (N1353) (Tydeman)

This paper notes problems related to sections of text, in C1x draft N1336,
as recently modified by N1321, and proposes ways to fix them.

Problem 1: The term "operations" is potentially misleading. The standard uses "operations" as a more general term than "operators", for example, in "Operations on files". One might wonder if library functions like sqrt() could be affected by wide evaluation. But only operators are subject to the usual arithmetic conversions, so "operators" could replace "operations" in the text to clarify the matter, without substantive change.

Proposed Change #1:
5.2.4.2.2 [8]: change "operations" to "operators".

Straw Poll: Adopt changes #1 from N1353 to the WP.
yes – 18
no – 0
abs – 0

DECISION: N1353, Adopt Proposed Change #1: 5.2.4.2.2 [8]: change "operations" to "operators". to the WP.

Problem 2: The new text in 5.2.4.2.2 and footnote 142 now implies that function returns are widened by widening expression evaluation methods. This specification is not tenable from an ABI perspective because it would introduce inconsistencies between the format expected by the callers and the format returned by the callee, depending on the evaluation method used for their translation. The following recommended changes revert to the original intention, namely that function returns, like assignments and casts, are not affected by wide expression evaluation methods.

Proposed change #2:
5.2.4.2.2 [8]: change "Except for assignment and cast" to "Except for assignment, cast, and return".
6.8.6.4 #3: change the second sentence to "If the expression is evaluated to a format different from the return type of the function in which it appears …"
Footnote 142: remove the last two sentences (thereby reverting to the C99 text).

Adopting these changes would indo what we did in DR290 (TC3), which we talked about at length. General discussion of FLT_EVAL_METHOD, and maybe we should simply leave it alone, i.e. leave it in the C99 state. Concern about breaking implementations. Fred decided to withdraw #2

[Thurs]
N1382 (NEW)
This paper notes a problem related to the following sections of text, in C1x draft N1362, as recently modified by N1321, and proposes a way to fix it.

Straw Poll: Adopt N1382 to the WP
Yes – 18
No – 0
Abs – 2

DECISION: N1382 Adopted to WP.

4.13 Treatment of math error conditions (N1354) (Tydeman)

N1353 proposes text to define what happens when math_errhandling is set to certain values. A poll taken by the author yielded a variety of answers, indicating the existing text is unclear.

Proposes the following changes:
In 7.12.1, Treatment of error conditions, 1st paragraph: Add:
If no error occurs, errno shall be unaltered and none of "invalid", "div-by-zero" and "overflow" shall be raised.
If an error occurs and if the integer expression math_errhandling & MATH_ERRNO is zero, either errno is unaltered or errno acquires the value corresponding to the error condition.
If an error occurs and if the integer expression math_errhandling & MATH_ERREXCEPT is zero, either no floating-point exception is raised or the floating-point exception corresponding to the error condition is raised.

In F.9.1.* thru F.9.10.*: Add 'is a domain error and' before 'returns a NaN and raises the "invalid" floating-point exception'.

In F.9.1.* thru F.9.10.*: Add 'is a pole error and' before 'returns +/-INF and raises the "divide-by-zero" floating-point exception'.

In F.9.1.* thru F.9.10.*: Add 'is a range error and' before 'raises the "overflow" and "inexact" floating-point exceptions'.

In F.9.1.* thru F.9.10.*: Add 'is a range error and' before 'raises the "underflow" and "inexact" floating-point exceptions'.

What real world problem is being solved here?
Tom: This is too low level for an application programmer to have to understand.

Straw Poll: Adopt the proposed changes to 7.12.1 in N1354 to the WP.
Yes – 7
No – 7
Abs – 7
No Consensus to Adopt the proposed changes in N1354 to the WP

Straw Poll: Adopt the proposed changes to Annex F in N1354 to the WP.
Yes – 3
No – 1
Abs – 16
No consensus to adopt the proposed changes in N1354 to the WP

4.14 Preferred quantum exponent for DFP math functions (N1355) (Tydeman)

Neither IEEE-754: 2008, nor TR24732. Decimal Floating Point, defines the preferred quantum exponent for transcendental DFP math functions with special values. Several implementations have taken different approaches.

Should we attempt to standardize this?
As anybody complained about this?
Is this a defect in the TR?
If IEEE did not specify it, why do we want to tackle it. They deferred it to implementations.
A programmer can detect this, but would not have a use for it.

No consensus to move forward with this at this time.

4.15 _Bool bit-fields (N1356) (Tydeman), See Also Ref: DR 262, DR 335

N1356 proposes to clarify whether or not implementations are required to support widths in the range of 0 to CHAR_BIT (inclusive) for _Bool bit-fields. The question revolves around what is meant by 'width'. C99 has a very specific definition of width which implies that the width of a _Bool bit field can be 1.

Requested changes to C1x:

Add a footnote to 6.7.2.1, paragraph 3:

While the number of bits in a _Bool object is at least CHAR_BIT, the width (number of sign and value bits) of a _Bool may be 1 bit.

Add to 6.7.2.1, paragraph 9:

A _Bool bit-field has the semantics of a _Bool.

Tom: Shouldn't the 'may' above be 'shall'. (re: 1 bit)? That's normative. DR335 was not a TC, it was a clarification (committee response). David: We answered wrong. It must be 1. Larry: Who says it has to be 1? The Standard says Implementation Defined.

Straw Poll: Adopt requested changes of N1356 into the WP.
Yes – 12
No – 2
A – 5
DECISION: Adopt proposed changes of N1356 into the WP.

4.16 tgamma range error (N1357) (Tydeman)

N1357 proposes a clarification of the behavior of tgamma when 'x' is too large or too small. In effect, for non-integers, tgamma(+/-large) is the same as exp(+/-large). So, tgamma should have: A range error occurs if the magnitude of x is too large.

Why? Fred wants required range errors for large magnitudes of X.

Two methods are proposed:
1) In 7.12.8.4 The tgamma functions, paragraph 2, change "A range error may occur if the magnitude of _x_ is too large or too small" to "A range error occurs if the magnitude of x is too large. A range error may occur if the magnitude of _x_ is too small".
or
2) Add a footnote to 7.12.8.4 The tgamma functions, paragraph 2, the sentence about range error:
For large positive _x_ values, overflow happens.

For signed _x_ values very close to zero, overflow may happen [depends if 1/max is underflow or 1/min is overflow; that is, if negative exponents are larger in magnitude than positive exponents].
For large negative _x_ non-integer values, underflow happens.

Jim Thomas believes the added text in #2 is too much information. PJ agrees

Any objection to #1 – none.

DECISION: Adopt Option 1 in N1357 to the C1X WP.

4.17 Constant expressions (N1365) (Tydeman)

REF: N1321, N1353, DR 290

N1365 proposes a clarification as to whether or not FLT_EVAL_METHOD is applicable when a floating expression is evaluated during translation time. It is unclear that it is supposed to be covered by FLT_EVAL_METHOD [5.2.4.2.2, paragraph 8] or not.

The issues come down to two points:

1) If FLT_EVAL_METHOD covers too much, then many implementations will end up with the value of -1 (meaning no information).

2) If FLT_EVAL_METHOD covers too little, we might end up with quiet changes from C99.

What is the usefulness of FLT_EVAL_METHOD, and how much information should it convey?

Proposed Changes:

Change 5.2.4.2.2, paragraph 8, "operations" to "execution-time operators".

Add to 6.6 Constant expressions, paragraph 5, These translation time constant expressions are not covered by FLT_EVAL_METHOD.

An alternative: Add to 6.6 Constant expressions, paragraph 5, When the translation-time environment differs from the execution time environment, these translation time constant expressions are not covered by FLT_EVAL_METHOD; otherwise they are covered.

Straw Poll: Adopt the proposed changes in N1365 to the WP.

Yes – 4
No – 2
Abstain – 14

No consensus to adopt this proposal. Now look at N1361.

Alternate Straw Poll

Adopt the following in N1361: Add to 6.6 Constant expressions, paragraph 5, These translation time constant expressions are covered by FLT_EVAL_METHOD.

Yes – 13
No – 0
Abs – 7

DECISION: N1361 - Adopt in the WP the following: Add to 6.6 Constant expressions, paragraph 5, These translation time constant expressions are covered by FLT_EVAL_METHOD.

4.18 TR 24732 (DFP): Problems with FLOAT_CONST_DECIMAL64 pragma (N1369) (Kwok)

N1369 proposes to clarify the intended behavior of the FLOAT_CONST_DECIMAL64 pragma, and possibly to rename it to a more meaningful name in TR 24732. The current wording in section 7.1.1 of the TR implies that the FLOAT_CONST_DECIMAL64 pragma applies to unsuffixed hexadecimal floating-point constants as well, but clearly the intent of the TR is not to support hexadecimal floating-constants for decimal types (see constraints violation in section 7.)

Suggested changes to TR

> In 7.1.1 of the TR, change "unsuffixed floating-point constants" to "unsuffixed decimal floating-point constants".
>
> Change the name of the pragma to DOUBLE_CONST_DECIMAL64 (or some other appropriate name.)

Start a defect log? Words to rationale?

ACTION – Convenor to work with TR 24732 PE to develop a lite-weight DR log and Rationale update mechanism.

4.19 Apple's extensions to C (N1370) (Garst)

N1370 proposes extensions to C implemented by Apple.

Garbage Collection

Apple has implemented garbage collection for C. The use of garbage collection is optional for any process that walks up to it. They've been able to make lots of comparisons between programs running with and w/o GC. Some programs are faster, some are not. The key property is that they never have to stop the world, i.e. stop all threads at once. Need write barriers to make this happen. The work is open source, running on an Apple platform, and is available. None of this is to say they have a perfect system, or that there are not issues. Do not take over malloc, allocate out of a special allocator. Not fully automatic, and does require a level of programmer cooperation. Uses strong and weak pointers.

Apple believes that C is the software to use to built system software, and using GC has been found to be beneficial, fast and efficient.

Clark: The state is C++ is that one cannot assume the presence or lack of GC.

Blocks

A closure is a function expression that closes over and preserves its lexical scope. A closure that is a first class object can be stored and any variables referenced from that lexical scope must be preserved beyond the lifetime of the declaring activation frame.

Apple has introduced a new compound type called a Block reference much like but distinct from a function pointer, a Block literal expression value syntax, and a new storage duration class for variables referenced from closures that live beyond the declaring activation frame.

Blocks have parameters like functions, which are checked by the compiler for type checking.

No decision reached on this material.

4.20 Thread Unsafe Standard Functions (N1371) (Crowl)

N1371 is a revision of SC22/WG21 N2827, and provides the following changes / clarifications:

- It is a joint paper with WG21/WG14 as liaison information
- Clarification that function call may race, not functions,
- Clarification that groups of functions may race with each other,
- Removal of functions that are not part of C/C++ Standards.
- Addition of requirements on floating-point environments.

With the introduction of concurrency into the C++ standard, some functions adopted from the C standard need explicit exemption from the general prohibition on data races.

The references in this document are for C++. They need to be converted to C.

Tom: How do locales get set? Dinkumware provides thread specific locales, POSIX does not. The paper uses the POSIX method, set locales once.

Errno needs to be discussed in this paper, or at least needs to be added through another paper

Straw Poll

Adopt N1371 for incorporation into the C1X WP, modulo the paragraph numbering and terminology changes needed for the C Standard vice C++ (WG21 N2800).

Yes – 20
No – 0
Abs – 0

DECISION: Adopt N1371 for incorporation into the C1X WP, modulo editorial changes as needed.

ACTION: PJ to write a rationale for N1371.

ACTION: PJ to write a proposal for incorporating errno, as applicable w/r/t N1371.

4.21 Threads for the C Standard Library (N1372) (Plauger)

N1372 is a proposal to add threads to the C Standard Library. It is a revision of an earlier document presented in Milpitas. At that meeting the committee approved this proposal for inclusion in the next revision of the C Standard, subject to improvements in wording. This document has been revised in response to those suggested improvements. It proposes that all declarations and definitions described here shall be placed in the new header <threads.h>.

This paper includes a Library only solution for thread local storage, which has been adopted in the WP. We could leave it in because it makes a stepping stone for Implementors. The semantics and machinery of the memory model and atomics are needed to implement this. Both have been adopted.

Is this related to pthreads? Yes, as the lowest common denominator, but is not the complete package provided by pthreads. Has been implemented in Linux and Windows.

Namespaces? There is precedence, exists in Dinkumware implementation.

Hans may have one or two minor changes, which PJ will be happy to incorporate.

Straw Poll

Adopt N1372 for incorporation into the C1X WP.

Yes – 17
No – 0
Abs – 4

DECISION: Adopt N1372 for incorporation into the C1X WP

ACTION: PJ to write a rationale for Threads, N1372.

4.22 Wording improvements for mblen, mbtowc, and c16rtomb. (N1373) (Plauger)

N1373 proposes improvements in the wording for three existing functions.

mblen:

> it is equivalent to
> mbtowc((wchar_t *)0, (const char *)0, 0); *// ADDED*
> mbtowc((wchar_t *)0, s, n);

mbtowc:

> ADD to description:
> If multibyte characters have a state-dependent encoding, the function stores the initial conversion state in an internal static-duration object at program startup. It updates this object on each call to reflect the current conversion state. If s is a null pointer, the resulting conversion state is the initial conversion state.

c16rtomb:

> ADD after "(including any shift sequences)" in Description:
> using the conversion state stored at *ps,
>
> ADD after "(including any shift sequences)" in Returns:
> and stores the resulting conversion state at *ps

These changes came about as a result of discussions in Milpitas, and an action PJ had.

Straw Poll

Adopt N1373 to the WP.

Yes – 19
No – 0
Abs – 2

DECISION: Adopt the proposed changes in N1373 (mblen, etc) to the WP

4.23 Unified Parallel C Overview (N1374) (Mak)

N1374 is a slide presentation on extending the C language to parallel programming.

[Thurs AM]

The link to the specification is http://upc.lbl.gov/docs/user/upc_spec_1.2.pdf

This presentation presents a view of what is needed for the C language to support parallel programming. UPC is an extension to C, and a PGAS language (See Slide # 28).

The number of threads needed is established prior to entering main(), and once set will not change while the program is running. There is a compiler generated variable named "THREADS". It's possible to make this dynamic. In a shared array, all threads can access all elements. Elements are distributed with affinity to threads. Controlling the parallelism is done through a upc_forall() statement. [chart 25 is missing a parameter in the upc_forall statement]. The distribution of the array can be controlled by the 'shared' qualifier, i.e. 'shared int'or 'shared[3] int' .

There are several implementations, commercially available, incorporating the UPC features.

Primary use today is the scientific programming community, high performance applications, academic research, shared memory and multi-core platforms.

There are some high level pieces that need to be worked on. This proposes some solutions to a subset of the problems associated with parallel programming. A final solution may be only a part of UPC, or none of it. Possibly consider a TR for this material, we don't have enough experience to try to standardize this material.

There are changes in processor architecture coming that could impact how approaches to parallelism can be done.

It's not clear whether there is a conflict with what we are doing with threads and memory model and the UPC approach. The UPC committee is following what we are doing, and will try to avoid such conflicts.

Although there is a lot of interest in this area, the committee took no action on pursuing UPC.

4.24 Adding EPOLE to Math Library Functions (N1319) (Tydeman)

[Fred – make sure it matches POSIX]
Add the concept of a pole error, Remove ilogb as well.
PE mistakenly added EPOLE to the WP.

Proposed Changes:

7.12.5.3 **atanh** change "A range error may occur if the argument equals -1 or +1." to "A pole error may occur if the argument equals -1 or +1."

7.12.6.7 **log** change "A range error may occur if the argument is zero." to "A pole error may occur if the argument is zero."

7.12.6.8 **log10** change "A range error may occur if the argument is zero." to "A pole error may occur if the argument is zero."

7.12.6.9 **log1p** change "A range error may occur if the argument equals -1." to "A pole error may occur if the argument equals -1."

7.12.6.10 **log2** change "A range error may occur if the argument is zero." to "A pole error may occur if the argument is zero."

7.12.6.11 **logb** change "A domain error or range error may occur if the argument is zero." to "A domain error or pole error may occur if the argument is zero."

7.12.7.4 **pow** change "A domain error or range error may occur if x is zero and y is less than zero." to "A domain error or pole error may occur if x is zero and y is less than zero."

7.12.8.3 **lgamma** change "A range error may occur if x is a negative integer or zero." to "A pole error may occur if x is a negative integer or zero."

7.12.8.4 **tgamma** change "A domain error or range error may occur if x is a negative integer or zero." to "A domain error or pole error may occur if x is a negative integer or zero."

Straw Poll: Adopt the above changes to the C1X WP (N1376)
Yes – 17
No – 0
Abs - 2

DECISION: Adopt the proposed changes in N1376 to the C1X WP.

4.25 Namespace Reservation for the C Standard (N1345) (Stoughton)

The C Standard does not have a mechanism to reserve namespace for it's own use. Clark pointed out that this addition is being proposed for the language portion, not the library, and would this be applicable to keywords.
Straw Poll: Adopt the changes proposed on N1345.
Yes – 6
No - 8
Abs – 4
No Consensus to adopt the proposed change

4.26 Encoding and Decoding Function Pointers (N1332) (Plum)

Proposes to add two functions, encode_pointer and decode_pointer, that would then be use to store an encrypted version of a pointers (vulnerability issue). They are similar to Microsoft Windows EncodePointer, DecodePointer. Discussed in Milpitas, but we found no way to move forward on this. Concern over the name attracting attention. Implementing a viable approach takes a systemic approach. Blain sees this as a typing issue. Could be addressed w/Attributes, but the status of those papers is unknown.

David S thinks this is a worthwhile first step.
Does this only make it slightly more difficult for the attacker? Several think so.
Also needs a failure more so that it not jumped into, creating a denial of service.
How to move forward. First we need a volunteer. David Svoboda.
ACTION – David Svoboda to explore an approach to code and decode pointers with Ulrich.

4.27 Unicode and Raw String Literals (N1333) (Plum)

This paper was approved in Milpitas, but the PE has problems incorporating it into the WP. It is C++ centric, and there are issues with implementing it. There is a semantic issue that needs to be resolved so that C and C++ will read / interpret a raw sting literal the same way that C does. Should we 'wait' to see what C++ finally decides? Yes. Thus this paper should be withdrawn

Straw Poll: Withdraw N1333, w/o prejudice, and wait on C++ to finalize their work.
Yes – 19
No – 0
Abs - 0

DECISION: Withdraw N1333 w/o prejudice, and wait on C++ to finalize their work.

4.28 WG14 Mail 11688 – character string literal (NEW)

The C standard uses the term "character string literal" to refer to an unprefixed string literal, i.e. a sequence of char as opposed to a sequence of wchar_t.

WG21 (or at least the core working group thereof) thinks that's not a very usefully descriptive term; after all, a wide string literal is also composed of characters. They would prefer the term "ordinary string literal".

The concern arises from the profusion of string literal types now in C++ -- which are also (as of the last meeting) also in C. In addition to the plain-old-strings and wide strings, there are UTF-8, UTF-16 and UTF-32 strings (five in all), each in raw and non-raw variants. Are any of these new forms of string literal supported in preprocessing directives such as #include and #line? Is an implementation permitted to define __DATE__ or __LINE__ as, say, a raw UTF-32 string literal? The answer to these questions is currently no, but the prohibition is expressed using the term "character string literal", which, again, is not a very usefully descriptive term.

So the CWG would like to define an unprefixed string literal to be an "ordinary string literal", and to keep the preprocessor description synchronized with C, would like to know whether WG14 would like to make the same terminology change.

Doug – this is bicycle shed. We should not do anything.
Clark – if we could come up with a couple of alternatives with a preference, or a firm commitment to not make a change.
Impact of doing nothing? Preprocessor will be out of sync.

Straw Poll: Change the name of 'character string literal' to something else.
Yes – 2
No - 14
Abs – 4
No consensus to make this change.

4.29 WG14 Mail 11572

What does "INT_MIN % -1" evaluate to? Tom proposes that we make it undefined behavior.
Randy: It's hard to get this right. Value may not be representable. Implementations should be allowed to get it wrong.

Add words "…otherwise the value is undefined." to 6.5.5;p6
ACTION: PE to adopt these words into the WP.

5. DEFECT REPORT REVIEW

5.1 TC for C1X (N1359) (Benito)

N1359 is the Technical Corrigenda for C1X. It contains the proposed changes for approved DRs, and the disposition of DRs not covered by this report.

Two DRs are not included: DR334, and DR314. They will not be addressed unless somebody writes a paper to do so.

ACTION – Fred to write up compatible changes to the LIA annex w/r/t DR330.

Straw Poll: Adopt N1359 to the WP.
Yes – 18
No – 0
Abs - 2
DECISION: Adopt N1359 to the WP.

6. REVIEW

6.1 Review of Decisions Reached

The following papers have attained consensus to be added to the WP, modulo the usual edits by the project editor as indicated or appropriate:

TR 24731-1 - as a conditionally normative Annex. (N1350, Proposal #4)

N1349

N1364 - with the addition of auto.

N1350 - Proposal #1 - Definitions for '*trap representation*' and ' *to perform a trap*' .

N1358 – Proposal #1, Proposal #2, Proposal #3 (something along the lines of)

N1360

N1367

N1377 (replaced N1352)

N1353 - Proposed Change #1: 5.2.4.2.2 [8]: change “operations” to “operators”.

N1356

N1357 - Option 1

N1361 - RESCIND N1321. Add to 6.6 Constant expressions, paragraph 5, These translation time constant expressions are covered by FLT_EVAL_METHOD.

N1371 - modulo editorial changes to convert C++ WP references to C WP.

N1372

N1373

N1376 (replaced N1319)

N1381

N1359

N1382

No Paper - Add words "…otherwise the value is undefined." to 6.5.5;paragraph 6 to the WP
See Also WG15 11572.

The following documents, previously approved, are to be redacted from the WP.

N1333 – Withdrawn w/o prejudice, and wait on C++ to finalize their work.

N1321 - Rescind the change proposed by N1321 adopted in Sept 2008. i.e. Go back to C99. Don't remove constants from the current standard evaluation methods, namely those for FLT_EVAL_METHOD values 0, 1, or 2. NOTE: Larry did not make exactly the correct change to the WP.

[Thurs AM]
When do we want to shut off new papers for the revision? There are a number of existing practice items, such as dynamic linkages, that we've not addressed. Do we expect a proposal along these or other lines. Add an agenda for Santa Cruz to put together a schedule for he WP.

Lots of discussion about work on DLLs, which got dropped in C++. If someone brings it, we will listen. The door is open for DLL proposals.

6.2 Review of Action Items

ACTION: Clark, Mark and Nick to write detailed attribute syntax proposal, focused on existing practice and the current list of attributes intended to be standardized.
REASSIGN – David Svoboda

ACTION: David Keaton to rework his previous proposal on anonymous unions, adding anonymous structures as appropriate. OPEN – in progress

ACTION: Randy to write examples for both DR 340 and 342 to allow vendors to evaluate what their implementations actually do in these cases. OPEN

ACTION - PJ to put together a paper on an alternate approach to implementing type-generic macros. (Picked up from Ulrich and Nick action)

ACTION: Arjun to revise N1335 to add alignof. JB wil ping.

ACTION: Larry Jones: Review use of headers required for the function declaration shown in the synopsis, and any others required to make the declaration valid.

ACTION: Convenor to find out from SC22 what we can do / not do with the existing TR, then look at how to proceed from there w/r/t N1351 and possible changes to TR18037.

ACTION: Tom Plum to check that the proposed definitions in N1350 do not conflict with the ISO Vocabulary standard.

ACTION: Convenor to work with TR 24732 Project Editor to develop a lite-weight DR log and Rationale update mechanism.

ACTION – David Svoboda to explore an approach to encode and decode pointers with Ulrich.

ACTION: Fred and Jim will rework the words in N1378.

ACTION: PJ to write a rationale for N1371.

ACTION: PJ to write a proposal for incorporating errno, as applicable w/r/t N1371

ACTION: PJ to write a rationale for Threads, N1372.

ACTION: Fred to write up compatible changes to the LIA annex w/r/t DR330.

ACTION: Clark write a paper to clarify the behavior that values are computed before main starts, and thread local storage get the same values as before program start up.

7. CLOSING

7.1 Thanks to Host

Thanks to Edison Kwok and IBM Canada for a great job hosting this meeting.

8. ADJOURNMENT

Meeting adjourned at 1:25 PM EDT, 2 April 2009

PL22.11 Meeting
31 March 2009

Meeting convened at 1630 hours by Chair, Randy Meyers

Attendees:

John Benito	Blue Pilot	
Larry Jones	Siemens PLM Software	Project Editor
David Svoboda	CMU/SEI (CERT)	
Robert Secord (phone)	CMU/SEI (CERT)	
Tana L. Plauger	Dinkumware, Ltd	
P. J. Plauger	Dinkumware, Ltd	
Edison Kwok	IBM	
Raymond Mak	IBM	
David Keaton	Self	
Arjun Bijanki (phone)	Microsoft	
Andrey Gusev	Oracle	
Barry Hedquist	Perennial	Secretary
Tom Plum	Plum Hall	
Bill Seymour	Self	
Clark Nelson	Intel	
Randy Meyers	Silverhill Systems	PL22.11 Chairman
Fred Tydeman	Tydeman Consulting	PL22.11 Vice-Chairman
Hans Boehm (phone)	Hewlett Packard	
Blaine Garst	Apple	
Douglas Walls	Sun Microsystems	PL22.11 IR
Jim Thomas (phone)	Hewlett Packard	
Hugh Redelmeier	Self	
Rajan Bhakta	IBM	

Agenda Approved as modified.

1. **Select US delegation for the next two meetings**. Not Applicable for this meeting.

2. **INCITS Anti-Trust Guidelines**

   We viewed the slides located on the INCITS web site.

3. **INCITS official designated member/alternate information.**

   Be sure to let INCITS know if your designated member or alternate changes, or if their email address changes. Send contact info to Lynn Barra at ITI, lbarra@itic.org.

4. **Emeritus Members**

ACTION: Chair to review the requirements for Emeritus Members, and determine if we have anyone who meets those criteria. Contacted Doug Gwynn about this, he is in the process of retiring, and is interested in being an Emeritus Member.

MOTION: Nominate Doug Gwynn as an Emeritus Member of PL22.11 (Seymore, Keaton)
Roll Call Vote:

Blue Pilot	Yes
Dinkumware	Yes
IBM	Yes
Intel	Yes
Keaton, David	Yes
Oracle	Yes
Perennial	Yes
Plum Hall	Yes
Sun Microsystems	Yes
Seymour, Bill	Yes
Tydeman Consulting	Yes
CMU/SEI (CERT)	Yes

Motion Passes 12-0-0

5. Bill Plauger has been selected as one of four winners of the 2009 INCITS Merit Award, which will be presented at the INCITS Officers Symposium dinner on April 20, 2009. This award is presented to no more than four INCITS participants who have demonstrated continuous support for the work of INCITS.
Criteria for nomination:
Long-standing (10+ years) member of INCITS and/or INCITS Subgroup;
demonstrated continuous support for the work of INCITS;
provided support to the Secretariat in progressing the work of INCITS;
served on at least two or more international committees.
Congratulations Bill.

6. J11 Reflector email has changed. Everyone should have received an email to that effect. If not, see Randy.

7. Adjournment

Meeting adjourned at 16:55