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

Meeting Location:
Cisco Systems
Building 54
510 McCarthy Blvd.
Milpitas, CA USA 95035

Meeting Host: ANSI

Host Contact information: Mark Terrel, mterrel@cisco.com

Meeting Dates: 8-12 September, 2008
Meeting Times:
08 September, 2008: 9:30-12:00 lunch 13:30-17:00
09 September, 2008: 9:00-12:00 lunch 13:30-17:00
10 September, 2008: 9:00-12:00 lunch 13:30-17:00
11 September, 2008: 9:00-12:00 lunch 13:30-17:00
12 September, 2008: 9:00-12:00

1. OPENING ACTIVITIES

1.1 Opening Comments (Terrel, Benito*)*

Mark welcomed us to Cisco. The meeting is sponsored by ANSI and Cisco Systems.
Restroom is in lobby.

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	USA	
Mark Terrel	Cisco	USA	
Tana L. Plauger	Dinkumware, Ltd	USA	
P. J. Plauger	Dinkumware, Ltd	USA	
Rich Peterson	Hewlett Packard	USA	
Edison Kwok	IBM	USA/CANADA	HOD - CANADA
John Parks	Intel	USA	
David Keaton	Self	USA	
Arjun Bijanki	Microsoft	USA	
Jeff Muller	Oracle	USA	
Barry Hedquist	Perennial	USA	Secretary
Keith Derrick	Plantronics	USA	
Tom Plum	Plum Hall	USA	
Bill Seymour	Tydeman Consulting	USA	
Clark Nelson	Intel	USA	
Randy Meyers	Silverhill Systems	USA	PL22.11 Chair

Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice chairman
Nick Stoughton	Usenix	USA/UK	HOD - UK
Hans Boehm	Hewlett Packard	USA	
Ulrich Drepper	Red Hat	USA	
Blaine Garst	Apple	USA	
Bill Bumgarner	Apple	USA	
Douglas Walls	Sun Microsystems	USA	HOD - USA
John Hauser	Self	USA	
Andrey Gusev	Oracle	USA	
Jim Thomas	HP	USA	
Robert Seacord	CMU/SEI	USA	
Christopher Walker	Dinkumware	USA	

1.3 Selection of Meeting Secretary

Barry Hedquist returned as 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 informal, and used to assess whether or not consensus exists to proceed to the next step indicated by the poll.

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

1.5 Approval of Previous Minutes (N1300) (Hedquist)

Several comments for typos, etc. Minutes approved as modified. N1343.

1.6 Review of Action Items and Resolutions (Hedquist)

ACTION: Derek Jones to make a pass through the C standard itself to ensure that the terms “format” and “encoding” are not used in any inconsistent manner, especially with respect to IEEE 754r. OBE - dropped

ACTION: Randy Meyers, Robert Seacord and Nick Stoughton to write additional rationale for the issues raised in N1210. CLOSE - DROP

ACTION: PJ to write paper on quick_exit.
DONE N1327

ACTION: Nick to submit a proposal on requiring errno be implemented as a macro. DONE N1338.

ACTION: Randy does a LOT more work with support from Joseph and Rich to look more deeply into DR 314 w/r/t optimization, safety, and security issues. DROP

ACTION: Tom and David will work on proposed wording for DR 340 for next meeting.

DONE N1342

ACTION: Robert Seacord and David Keaton to write a paper on static assert.
DONE N1330 (Plum)

ACTION: Tom to provide list of undefined behavior that should become constraint violations to match C++ usage.
DECIDED TO NOT PURSUE THIS AREA - OBE

ACTION: JB to work with Larry Jones to ensure that Nick has an up to date set of document sources.
DONE

ACTION: Nick Stoughton to set up shared version control system for the document editors. OBE

ACTION: Nick to get revision of TR24731-2 into pre-Santa Clara meeting mailing.
DONE N1337

ACTION: Editor to add the N1282 words to the Working paper.
DONE

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

ACTION: Editor to add the words from N1252 to the WP.
DONE

ACTION: Editor to add factored approach from N1285 to the WP
DONE

ACTION: PJ to talk to compiler writers about type generic issues and overloads for C.
Under way.
DONE N1340

ACTION: Keith to develop attribute placement requirements as a part of a template for attributes.
OPEN – IN PROGRESS

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.
OPEN – IN PROGRESS

ACTION: Nick to forward N1287 to the Austin Group for their prompt review and comment.
DONE N1341

ACTION: PJ to present actual edits for threads for the next meeting in Santa Clara (Sep 2008).
DONE N1325

ACTION: Tom Plum to develop a fully-formed proposal for try-finally.
OPEN

ACTION: PJ to turn TR19769 into editing instructions for inclusion into the WP.
DONE N1326

ACTION: Clark Nelson to develop a proposal for C based on WG 21 N2545 (thread local storage).
DONE N1329

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

ACTION: Edison to apply edits as described in N1295 as modified during the Delft meeting to the Decimal Floating Point TR.
DONE

ACTION: Edison to apply edits as described in N1293 to the Decimal Floating Point TR.
DONE

ACTION: Rich and Edison to work on the %a and %A printf conversion specifiers in relation to decimal FP and propose a solution.
DONE N1312

ACTION: Fred Tydeman to develop a fully-formed proposal to add xxx_SUBNORMAL_MIN and xxx_MAXDIG10 to C1x. In Progress
DONE N1317

ACTION: Edison to remove xxx_SUBNORMAL_MIN and xxx_MAXDIG10 from DTR 24732.
DONE

ACTION: Editor to remove 7.12.3.1 para 4 from the C1x working draft.
DONE

ACTION: John Benito to produce a "Technical Corrigendum" document describing how to apply all as yet unapplied defect resolutions to the C1x working paper.
OPEN

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: Tom to develop a proposal to remove K&R functions (and making VLAs optional?).
DROP ACTION ITEM

ACTION: JB to provide list of DRs that are to be handled in C1X.
DONE N1307

ACTION: Tom and Arjun to make a proposal for new library APIs for EncodePointer() and DecodePointer().
DONE N1332

ACTION: Fred to tidy up the FLTDEN paper and present editing instructions to the committee. Dup of earlier.
DONE N1317

ACTION: Fred to poll the SC22WG14 and Austin Group reflectors to ask for implementation experience with longjmp from a signal handler and refine his paper accordingly.
DONE N1318

ACTION: Fred to propose editing instructions to make conversion between pointer and floating types a constraint violation.
DONE N1316

ACTION: Willem to submit changes to C1x to flood static memory with zero bytes, then patch floats and pointers as needed.
DONE N1311

ACTION: Clark to propose actual text to remove padding bits from signed char types.
DONE N1310

ACTION: Tom to develop fully formed proposal to remove trap representation for signed char types.
OPEN

ACTION: Arjun and Mark to develop a fully formed proposal on alignment issues (align attribute, _Unaligned etc).
DONE N1335

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

1.7 Approval of Agenda (N1334)

Revisions to Agenda:

ADD: ITEM 5.1, DEFECT REPORTS N1342

Agenda approved as modified.

1.8 Information on Future Meetings and Mailings.

1.8.1 Future Meeting Schedule

N1328
Spring 2009: IBM, Markham, 30 March - 03 April, 2009.

Fall 2009 - Santa Cruz, CA, Plantronics / ANSI host
Preliminary date: C: 19-23 October; C++: 26-30 October

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. JB will start working to get a host for April 2010. There will be teleconferences before then.

1.8.2 Future Mailing Schedule

Post Santa Clara mailing: 2008-10-10
Pre Markham mailing: 2009-03-02

1.9 Identification of National Bodies (Benito)

US, Canada, UK

1.10 Identification of PL22.11 voting members (Tydeman)

16 of 17 voting members present.

2. LIAISON ACTIVITIES

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

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

SC22 Plenary in Milan coming up in two weeks.

Special Math IS 24747 is done, ready to publish

Embedded TR 18037 in same black hole.

DFP TR 24732 has been submitted.

Only TR24731-2 is open, along with DRs.

Major work item now is the C1x revision, and two defect reports.

John Benito has volunteered for another term as WG14 Convener.

2.2 WG21 / PL22.16

2.2.1 Rationale for C++ WP Definition of 'memory Location' (N1315) (Boehm)

Outlines the reasoning behind the definition of "memory location" used in the C++ concurrency model. That definition states: "A memory location is either an object of scalar type, or a maximal sequence of adjacent bit field all having non-zero width. " [Notes and examples excluded.] There is concern by members of the C Committee that definition is too restrictive on the implementation.

Discussion:
Latest plan is to put out a CD at the meeting in San Francisco next week. May be able to get an FCD out by the end of 2009. PJ: The biggest single item that affects the two languages together is probably the memory model. The C++ definition for memory location will be in that CD.

Tom has been reviewing the C++ proposals for applicability to C, and has one paper he's bringing forward.

N1315 - Hans Boehm.
Hans walked through his slide presentation describing N1315 as a liaison report from C++.
The slides are on the wiki as: location-for-C.pdf.

One key issue for C is the treatment of bit-fields within structures. If the bit-field is small, it may be hard to guarantee that assignments don't interfere with a thread. In the C++ definition, adjacent bit-fields are in separate memory locations, and therefore can be updated concurrently by two execution threads without interference. The same applies to two bit-fields, if one is declared inside a nested struct declaration and the other is not, or if the two are separated by a zero-length bit-field declaration, or if they are separated by a non-bit-field declaration. It is not safe to concurrently update two bit-fields in the

same struct if all fields between them are also bit-fields, no matter what the intervening bit-fields happen to be.

Randy – we are trying to address the treaty point between the optimizers and the synchronizers. The treaty point needs to be established, and this is a reasonable point to establish that point.

Rich's issue is that, for C, the user community does not want a memory model imposed by default. There are significant changes for the code generators, and there's a lot of C code out there that is not threaded.

PJ – if the code is single threaded there is no assumption the compiler needs to make. There is an agenda item for this topic later.

2.3 Linux Foundation (Stoughton)

Nothing to report.

2.4 WG11 (Wakker)

No Report

2.5 OWG: Vulnerability (Plum)

Nothing new to report.

2.6 Austin Group - Unofficial Comments on N1287, Threads API. (N1341) (Stoughton)

See Also: Agenda 4.7

Discussion: Nick provided feedback from the Austin Group on N1287, Threads API: "Here we go again,…."

tgamma – possible POSIX issue that we may be able to clarify.

3. EDITOR REPORTS

3.1 Report of the Rationale Editor (Benito)

Nothing new to report.

3.2 Report of the Project Editor (N1336) (Jones/Stoughton)

New draft available, incorporates actions from Delft, some editorial corrections, ISO boiler plate updated, along with abstract, patent info, defined 'sequenced-before' change bars inserted.

ACTION: JONES - Update the sequence point annex in the WP.

If Larry makes a change that is NOT regarded as editorial, please speak up.

3.3 Status of TR24731, Bounds Checking (Meyers)

Nothing new to report.

3.4 Status of TR24732, Decimal Floating Point (N1312) (Kwok)

N1312, 2009-05-06, is the latest version of TR 24732, and incorporates changes adopted in Delft. It has been sent to ITTF for publication.

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 (N1337) (Stoughton)

Document is actually identical to the prior, except for heading information, and should be ready for PDTR ballot. Tags were added similar to C++.

ACTION: Convenor to forward, N1337, DTR24731-2, to SC22 for PDTR Ballot.

3.7 Status of TR18037, Embedded Processors (Wakker)

No Report

4. DOCUMENT REVIEW

4.1 Promised Future Action (N1307) (Benito)

This is a list of closed DRs that may be incorporated into the WP.

4.2 Initializing Static or External Variables (N1311) (Wakker)

This paper came out of SC22WG14.11416, and addresses initialization of static or external variables that are not initialized explicitly. In Delft, a decision was made to propose wording to the WP to flood static memory with zero bytes, then patch floats and pointers as needed. The paper provides those words

Proposed change to C1X:
Change the 2nd sentence of 5.1.2p1 to:
Before program startup first the storage area that holds all objects with static storage shall be cleared (all bytes are set to zero), then all objects with static storage duration shall be *initialized* (set to their initial values).

Is there still an issue here? If an object is initialized to zero, which zero is it? Agreement that the words are fine for the abstract machine.

Straw Poll:
Q: Move the proposed words from N1311 into the WP.
17-0-0

4.3 Requiring Signed char to Have No Padding Bits (N1310) (Nelson)

C++ requires that all character types have no padding bits, and the C Standard requires that unsigned char has no padding bits, however it allows padding for signed integers w/o excluding char. This paper adds words to the WP prohibiting signed char from having padding bits.

Straw Poll:
Q: Add N1310 to the WP.
22-0-0

4.4 Changing some Undefined Behavior to ill-formed (N1324) (Plum)

In Delft, N1278 proposed changing some Undefined Behavior to ill-formed (a C++ term). This paper proposes to leave the matter as is, recognizing that compile-time Undefined Behavior already allows for the issuing of a diagnostic.

Tom asked that this paper be withdrawn.

4.5 Critical Undefined Behavior (N1331) (Plum)

Another follow-on to N1278 focused on the concept that any 'bug' in some component can lead to incorrect results in all downstream components.

Newer version, N1344, on the wiki.

Focus on the subset of Undefined Behavior that results in storing out-of-bounds.

Could Critically Undefined Behavior actually be categorized as "bad-non-portable code". For the most part, yes.

John would like to see examples as applied to this proposal.

The proposed definition for Critical Undefined Behavior is really the same as the existing definition for Undefined Behavior.

The new proposed definition for Undefined Behavior adds after "…imposes no requirements" the words: "… except that the behavior shall not perform an out-of-bounds store and that all values produced or stored are unspecified values. NOTE: The behavior might perform a trap."

Much discussion, both in the meeting and during the break (side discussions). It's not clear what conclusions were reached, if any, but Tom will produce a revise version of the paper.

ACTION: Tom to produce revision to this document based on the discussion.

If we establish a "Security Profile" is it a problem? As long as the Standard does not mandate that conformance to 'x' is required. Profiles are constructed as separate documents rather than as part of a 'base' standard. Consider putting a profile in an Annex?

Does adopting an approach of using 'strictly conforming' code help the security issues? Tom does not think so, because the effort required to do so would be onerous. Using only 'strictly conforming' code makes it very difficult to write meaningful programs.

4.6 Adding TR19769 to the C Standard Library (N1326) (Plauger)

Delft: Action to Plauger to turn TR19769 into editing instructions to add to the WP.

PJ presented the paper. It's basically the TR in standardese. Some folks are interested in the types, but not really interested in the functions. This TR provides a minimal level of support for Unicode, and has

been adopted by C++ in their revision. We have elected to do so as well. Some discussion on the use of UTF-16 vs. UTF-32. UTF-32 is more universal, there is limited use of UTF-16.

Rich is concerned there is a possible issue with the c16rtomb ? There seems to be some information missing w/r/t what happens with surrogate pairs.

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

Straw Poll:
Q: Add this material, TR19769, N1326, to the WP?
14-1-7

Q: Cut the functions out, and leave the data types and literals?
4-13-5. NO CONSENSUS

Discussion prior to Straw Poll.
PJ: What is the point of keeping the 16 bit literals and throwing out the functions? Douglas – the literals and data types were adopted by some users, but not the functions.

Ulrich: people who want to write 'new' code in 16 bit, already have their own tools. Most will use 32 bit code.

PJ believes there is a growing need to manipulate UTF-16 text, even if it isn't the right thing to do.

4.7 Adding Threads to the C Standard Library (N1325) (Plauger)

Delft: PJ presented N1287, a paper on the Dinkumware threads library, which is a very thin wrapper on the POSIX pthread interface. That paper was forwarded to the Austin Group for review and comment - See Agenda 2.6, Liaison to Austin Group. From the Introduction in N1325:

> "A thread in this document is a separate flow of execution within an application. On a multi-processor system threads can execute simultaneously on different processors. On a single-processor system and on a multi-processor system with fewer available processors than active threads two or more threads must share a processor. The details of switching a processor from one thread to another are handled by the operating system and are not covered in this document."

Discussion:
The document is missing a header file name.
Ulrich would prefer that this feature be made optional. He does not want to use it, and doesn't want to be required to implement it.

Hans Boehm has a couple of technical issues, N1341.

Straw Polls:
1. Should we continue this work?
13-5-4

2. Should we make this an optional package?
8-8-7
No consensus to change.

N1341 (Stoughton)
Raw individual comments from members of the Austin Group on the Thin Threads proposal, N1325.

Hans reviewed the comments he made. PJ will pass Hans comments on to Pete Becker to make sure his concerns have been addressed.

David Butenhof:
Concern over a lack of an extension mechanism to change any semantic attributes of threads before starting them.

Wants a good memory model.

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

4.8 Thread-locale Storage (N1329) (Nelson)
Parallel memory Sequencing Model, (N1284) (Nelson) Moved from 4.25

Discussion:
Clark presented N1284, which is the paper that added the definition of 'memory location'. There is some concern that the proposed change will impact legacy code, however there does not seem to be a solution that will completely resolve that concern.

Rich would like an option expressed in the source code that affects how this is handled. Need to be able to distinguish between memory models, threaded or non-threaded. We do not have a paper for a source construct. Rich does not disagree with the model presented, but is concerned about the legacy code issue.

PJ – if the 'dusty deck' is a single threaded program, there is nothing that would cause the code to be changed. Single threaded programs can be linked and run with multi-threaded w/o changing them. We know that we cannot fix this issue with attributes.

David wants N1284 to sit until we resolve the issues. The key element in N1284 is the definition of Memory Location, and we have not really examined the rest of the paper. What else is there that we should be concerned about. C++ has been working this issue, and this paper is not really complete.

PJ and Nick believe we should not make this a C++ dependency, but we need to make this as good as possible, and move forward with it.

Randy willing to let the single threaded code run a little slower. Proper granularity of access needs to be resolved, and Randy believes that is byte level.

We formed a subcommittee in Delft to address the issues discussed, but we do not have a 'new' paper in front of us.

Hans is not convinced that 'performance loss is a major issue, and ABI compatibility should not be a real issue – the old code should work.

Keith – if the compiler lays out the struct differently, the ABI could be broken.

Clark – if that happens, it's the compiler vendor that has broken the ABI, not the standard. We are not forcing it. Should not be an issue with existing compilers. The belief that ABI would not be broken really depends on how the code was written.

Hans believes such could not be possible for correctly written code. Certainly, code that depends on a bug could break the ABI.

The library part of this paper needs editorial work, and Clark needs some help in figuring out some if this stuff. There are a lot of in-line notes, and casting them all as footnotes is not optimal. Clark is not prepared to do the editorial work.

Larry will work with Clark to sort out the editorial issues. Clark will produce a revision prior to the next meeting. Everyone needs to give some thought on the granularity issue. Can we back off from saying that int is the smallest object that can be atomic.

Straw Poll

1. sequence of bit fields (C++ has adopted this) In favor -15

2. sequence of bit fields plus narrow integer fields on either end. In favor - 3

3. if a struct has a bit field, then the whole struct is a memory location: in favor -3

Tom believes #2 and #3 are infeasible.

Put the bit field in a sub-structure, create additional boundaries for the compiler to respect?

Straw Poll provides direction for #1.

N1329
Proposal to add thread_local storage as a local storage specifier. (C++ adopted).

Q: Can the address of a thread_local object be accessed from two separate threads? Consensus is that the standard should not make such a guarantee. One option is to have it defined in C++, but undefined in C. Much discussion. Make it implementation-defined? Tom, this does not seem to be a position we have complete knowledge of, so it may be best to make it implementation defined.

Options:

1. Yes - It has to work. In favor: 4

2. It's implementation defined whether it is supported. in favor: 18

Consensus – make it Implementation Defined as to whether or not it is defined.

Spelling of "thread_local" as a storage class specifier. Likely to create a conflict with existing uses of this term. Use and underscore, as in "_Thread_local". Take it for granted that there will be a header for threads that will include a macro for this.

What does the term "object" mean? An object is a region of memory.

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

4.9 Encoding and Decoding Function Pointers (N1332) (Plum, Bijanki)

In Delft, during the discussion of N1279, Attributes Commonly Found in Open Source Applications, regarding constructor/destructor attributes, Tom pointed out that Microsoft had developed a technique to

hide function pointers. The purpose is to prevent a hacker from overwriting a function pointer, thereby hijacking the process. The approach in this paper is to encrypt the pointer, and proposes two new functions: encode_pointer, and decode_pointer.

Discussion:
Do this as a macro? Static inline function? To be resolved.

Nick: There is existing code that uses these names, so if it's a macro it will break existing code.

Clark: This sort of functionality is unprecedented as far as the Standard goes. It's ground that we've never been on before.

PJ shared that concern, but pointed out that Microsoft has gotten a lot of mileage out of doing this.

Clark is comfortable that there is existing practice, but there are number of things about implementing something like this that we do not have expertise in.

Red Hat is doing things like this, but never intended to make the feature an interface. When a function pointer is stored, it is not obvious that it is encrypted or not encrypted.

Microsoft has had this as an interface for about four years, but its use outside of Microsoft is unknown. It is being used internally.

At Red Hat, its use is internal only, since it is not a published interface.

Apple is also doing stack protection, so Blain would like to see a language solution explored.

Tom is concerned that a language solution will be a greater unknown than the solution proposed in N1332, and we could end up doing nothing.

Randy: This could be a case of the better being the enemy of the good, and that a language solution may be the better, and a better solution for C++. Doing this in the language is certainly worth looking at.

Clark, maybe we should consider a TR?

David K: we have an opportunity here to nip this problem in the bud, and would really like to explore doing this in the language.

Straw Poll:
1. Accept N1332, as is, for the WP. 3-12-7. No consensus.

4.10 Adding Alignment Support to C (N1335) (Bijanki)

Arjun presented a paper in Delft on adding _unaligned, and took an Action Item to work on other alignment pieces as well. This paper is based on the C++ proposal contained in WG21 N2341 in a "semantically equivalent manner" .

Two questions are asked in the paper:
1. Should packing and weaker alignment requirements be considered ?
Clark – should not pursue packing. Cisco uses packing extensively, and do other embedded system folks.

2. Does C need aligned allocation (e.g. posix_memalign) ? Nick says yes, but you might want to change the name.

Why not alignof? C does not have a requirement for it. Clark believe that it does.

Are people comfortable with 'fundemental_alignment' and 'extended_alignment" (See N1335, 6.2.8;p2&3). Underspecified ?

6.2.8, para 7, text regarding '..the implementation is also allowed to silently disregard the requested alignment'.

Intention is that you can only 'add' padding, not reduce it.
Arjun, packing sounds like a separate paper

John H: The fact that the Standard treats a pointer to an int differently than a pointer to a char probably scuttles the whole idea of packing, because all pointers would have to be treated as pointers to chars.

Packing is an issue, does not seem to be resolved.

Dynamic allocation: agree that this is a good thing.

4.11 Extensions to the C1X library to enhance security (N1339) (Seacord)

N1339 describes extensions to the C1X library to enhance security in the C language. Most of these are existing functions, while some of these are enhancements or minor modifications to existing library functions. Each of these is stand-alone function.

David Svoboda led us through the discussion of this paper.

#1 fopen() exclusive access with “x”
This is an extension to Linux and POSIX in glibc, existing practice. This differs from fopen_s. PJ may have an issue with this, and sees possible problems with the fail-safe mode for this function. Ulrich believes that are many alternative mechanisms to try.

Straw Poll Q: Continue with developing this: 14-0-6: YES

#2 sigaction()
Existing implementation: POSIX
Ulrich - does not care about this for ISO C. There are many platform extensions that are necessary to implement this function properly. There is a lot of work needed with other supporting functions, not just this function.
John A – the existing facility in the C Standard is insufficient as is, so if the signal utility is to have any real use, more support is needed.

Recommendation is that 'secure coding' not use signals at all. It's not usable in a portable way, and has always been known to have problems. These issues are identified in the C Standard Rationale.

CERT recognizes more work is needed here.

#3 random()
Pseudorandom number generators use mathematical algorithms to produce a sequence of numbers with good statistical properties, but the numbers produced are not genuinely

random. The C Standard function rand() (available in stdlib.h) does not have good random number properties. The numbers generated by rand() have a comparatively short cycle, and the numbers may be predictable.

John A thinks a better solution would be to tighten up the specification for rand(). Robert S thinks rand may be beyond saving. Legacy versions would be a problem. PJ – it really hard to pick a single attribute of a random number generator and say that's the issue.

Mark T: It is really a QOI issue, and 'what are you using it for'.

Newer systems are generating better random number generators in the CPU.

There is no point in trying to fix what we've got, because it is really a market issue.

Suggestion that the Cert Guideline state to NOT use rand(); if you have a POSIX interface, use random() instead; if you have Windows, use blah-blah…

We don't really like the proposal, but have no suggestions on how to make it better. Bad implementations of rand() exist.

No consensus to move forward with this as is.

#4 fremove
Removing a file via its filename is subject to a race condition. The following code example shows a case where a file is removed. After the program closes the open file, but before the program removes the file, an attacker can substitute the file to be removed with a different file, causing the program to remove the wrong file.)Removing a file via its filename is subject to a race condition. In an example in the proposal, a case where a file is removed. After the program closes the open file, but before the program removes the file, an attacker can substitute the file to be removed with a different file, causing the program to remove the wrong file.
Discussion: This requires OS support to work, as well as library support. The text refers to file descriptors, but should be file pointers. It's not clear that this will accomplish its intended function. From an OS perspective, this is new ground. What are the problems we are seeing. CERT really needs to work with somebody that understands libraries to come up with fixes to the problems they see. Ulrich pointed out that this is functionality that has deliberately not been implemented. Suggested talking to the OS people to come up with a approach to accomplish he goal. No consensus to move forward as proposed.

#5 const char getenv()
proposse that getenv() should return not a char, but rather a const char* to prevent programmers from inadvertently modifying the string.*
Ulrich – this is not what we want, and it would make existing code invalid.
Arjun – suggested looking at getenv_s

#6 setenv()
C99 currently provides no mechanism to modify environment variables, although it provides the getenv() method to access environment.variables. The POSIX setenv() function has become the standard means of modifying individual environment variables.

POSIX has this, but discourages its use. It is not thread safe. The time to modify the environment is when it is passed to exec. setenv() is not a secure interface.

#7 clearenv()
David K – most of the time what's needed is resetting the environment to the default environment.

Ulrich – clearenv cannot be modified, but there may be another way to introduce a default environment, which he would be in favor of. Move forward, but in a different form/name.

#8 siglongjmp()
The C99 longjmp() function does not specify its behavior with respect to signals. This may result in undefined behavior for programs that rely on signal handling. The POSIX siglongjmp() and sigsetjmp() functions address this problem by specifying that the signal state is restored. This is particularly significant for programs that use the proposed sigaction() function.

This is useful w/o sigaction. Move forward on this & propose standard words. But, like sigaction, a good deal of more work is needed to moved forward with this.

#9 fmkstemp()
Temporary files are commonly used for auxiliary storage for data that does not need to, or otherwise cannot, reside in memory and also as a means of communicating with other processes by transferring data through the file system. For example, one process will create a temporary file in a shared directory with a well-known name, or a temporary name that is communicated to collaborating processes. The file then can be used to share information among these collaborating processes.

POSIX has a function called fdopen that converts a file descriptor to a function pointer.

As a secure practice, this is not necessary. Implementing this feature would require OS extension support. Its really an OS issue to get right. Also QOI, even with OS support. Can this be a recommended practice? ISO C does not know anything about 'permissions', or 'directories'. Nick believes that a beefed up 'recommended practice' for tmpnam will accomplish the goals.

Not clear what problem this function is trying to solve.

Possible combine some of the function that look reasonable here into a newer, more complete proposal.

4.12 Abandoning a Process (adding quick_exit and at_quick_exit) (N1327) (Plauger)

This paper is an adaptation of WG21 N2440, which has been accepted as a revision to the C++ Standard. It provides a mechanism to cancel a process from the outside when an application is unable to do so internally.

Why do we need this in C? It's not clear. PJ submitted this because he was asked to do this for compatibility to C++. Why? Important to the C++ concurrency group. When the threading package was put forward in C++, there was no provision for terminating a thread. C tried to address the problem. Crowl pointed out that that when things are really ugly, there is no way to clean up the mess w/o something like this. Basically, that's why it's been proposed. Nick would like to see a prohibition in quick_exit handlers from calling exit. PJ agreed. Crowl designed these in C++ in such a way as to make them callable from C.

Straw Poll:10-2-10 YES
Q: Add N1327, modulo changes discussed, to the WP.

4.13 Static Assertions (N1330) (Plum)

This paper defines a specification for static assertions which would achieve maximal compatibility with the C++ definition. It is a word-for-word adaptation of WG21 N1720, and Incorporates feedback from WG14.11495.

This proposal is meant to be compatible with C++ static_assert.
The examples come from the C++ proposal. Change 'ill-formed' to 'diagnostic required'.

Straw Poll
Q: Add N1330 to the WP with the above change: 20-0-2 YES

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

This paper defines a specification for 'u8' (UTF-8), 'u' (char16_t), 'U' (char32_t) and raw string literals which would achieve maximal compatibility with the C++ definition. It is a word-for-word adaptation of SC22/WG21/N2442, with section numbers and standards text adapted to WG14/N1256. It's likely that there are errors in this volume of cut-and-paste; the presentation attempts to follow WG21/N2442 paragraph-by-paragraph.

Straw Poll:
Q: Add N1333 to the WP, modulo the usual.
11-2-10

4.15 Extensible Generic Math Functions (N1340) (Plauger)

There is no portable way to implement the type-generic math functions defined in <tgmath.h>. This paper proposes a mechanism to do so.

Ulrich may have an approach to doing this based on GCC implementation that uses two built-ins. PJ would like to see a written proposal based on that approach. We need a good generic solution to this problem in order to get a solution that does not depend on compiler-magic.

ACTION: Ulrich to put together a paper on an alternate approach to implementing type-generic macros.

We will defer action on PJ's approach until we can review the GCC approach.

4.16 Conversion between pointers and floating types (N1316) (Tydeman)

Currently, the standard is silent on what happens when a conversion is done between a pointer type and a floating type -- which means it is implicitly undefined as per section 4, paragraph 2, in conformance. The Committee decided to make such conversion a Constraint Violation. This paper provides the words to do so.

Makes an attempt at such a conversion requiring a diagnostic.

Straw Poll

Q: Add N1316 to the WP 22-0-0

4.17 New macros for <float.h> (N1317) (Tydeman)

This paper proposed macros for minimum subnormal numbers when they are supported by the implementation.

Possible C++ compatibility issue with C++ names

Straw Poll
Q: Revise N1317 for compatibility with C++
12-0-9

4.18 longjmp() from signal handler (N1318) (Tydeman)

*Currently, there is no way to leave a signal handler for **SIGFPE** and continue execution. **abort()** and **_Exit()** are the only two functions that can be called from a signal handler; both result in program termination. Both **return** (for **SIGFPE**, **SIGILL**, **SIGSEGV**, or implementation-defined values) and **longjmp()** (for any signal) being used by a signal handler results in undefined behavior. Fred conducted a survey and did some research to identify the problems associated with the use of longjmp from signal handler. This paper presents those results, and proposes three possible choices:*

> *1. Drop this idea and do nothing to the standard (but maybe update rationale (about undocumented quiet changes)).*
>
> *2. Pursue the idea of more general signal handlers along the lines of N821, item PC-UK0097 (major writeup on **longjmp** and signal handlers). The previous version of this proposal was a very small part of that.*
>
> *3. Pursue the idea of IEEE-754:2008 and alternate exception handlers. Since there are most likely no implementations of this (IEEE-754:2008 has not been published), there is no prior art in a commercial product, this would be invention.*

Fred got lots of feedback on this. There are two conflicting ways to do this, one of which has probably not been done. Fred recommends doing nothing until the community sorts out a way to do this.

DO NOTHING.

4.19 Adding EPOLE to math library functions (N1319) (Tydeman)

Some error conditions for some of the math functions (for example log(0.0)) have been controversial in the past in that some implementations claimed that they were domain errors, while others claimed that they were range errors. They truly are their own class of error and should be treated as such. In POSIX, for example, atanh() has pole error, domain error and range error.

Note, per the Standard: A 'domain error' is one where an input argument is outside the domain over which the function is defined. A 'range error' is one where the result cannot be represented in an object of the specified type. Disagree with the above. The Standard is clear the log(0.0) may be treated as a range error, i.e. treating it as a domain error is NOT an option.

What is the impact of these requested changes? We do not want to make a change that is going to force implementations to change. POSIX has not asked for this change. Changing the 'may' in the listed functions to 'shall' denies Implementers a choice of what to return. PJ is not sure there is something to fix.

Straw Polls:

1. Add the description of 'pole error' from N1319 to the WP. 14-0-5

2. Accept the changing of 'may occur' as shown in N1319, implying a 'shall' in its place. (5-2-12) no consensus for change.

4.20 Integrating C89 Defect Report 25 into C1x (N1320) (Tydeman)

Not clear why this was not done in C99. Perhaps no one brought a paper forward?

The issue is whether or not a program containing these numbers can be compiled.

The material proposed as "changes to the Rationale (or Standard)" are acceptable for the Rationale, but not the Standard.

Straw Poll
Q: Adopt N1338 to the WP, modulo words to insure that errno is reserved in the external namespace. 22-0-1

Straw Poll:

Q: Accept proposed changes to the Standard contained in N1320. 8-0-7 YES

4.21 Split FLT_EVAL_METHOD into operations and constants (N1321) (Tydeman)

Proposes to add FLT_CONST_METHOD

Straw Poll:
1. Accept N1321 to the WP, removing const from FLT_EVAL_METHOD 8-1-8 YES

2. Accept FLT_CONST_METHOD to replace const 4-4-10 - no consensus

Nick has concerns that this may have a POSIX impact.

4.22 Translation-time expression evaluation (N1322) (Tydeman)

DR 300. We initially decided that was a new requirement, so now it is being brought forward as a new requirement to add to the WP.

Straw Poll
Accept N1322 for the WP
3-9-8 – no consensus.

4.23 Pure imaginary types with classification macros (N1323) (Tydeman)

When complex and imaginary were added to C99, most people were only thinking in terms of complex. Some restrictions were added which makes sense for complex (as they have no meaning). Unfortunately,

that wording also outlawed the use of those operators with pure imaginary types (where they do have a meaning).

The committee rejected the idea of allowing pure imaginary with increment/decrement operators, with relational operators and/or with comparison macros.

This proposal relaxes constraints to allow imaginary types (along with real types) to some operators.

There is no 'constraint' per se, it's a matter of semantics.

PJ concerned about the impact on type-generic functions, and deos not want to rush into anything. He does not think that EDG could handle this.
Fred offered to take this off the table until the type-generic issues can be resolved.

4.24 More Thoughts on Implementing errno as a Macro (N1338) (Stoughton)

N1257 describes an inconsistency in the wording for <errno.h>. N1308 followed up proposing changes in wording to both C and C++ to bring them into alignment with POSIX, allowing errno to be implemented as either a macro or an identifier with external linkage.

*Following discussions with both the C and C++ committees, it is clear that the cost of **requiring** it to be defined as a macro is infinitesimally small, since in the worst case it simply requires*
#define errno errno

In the vast majority of known implementations, errno is already a macro.

To that end, this proposal seeks to clarify the wording in the C standard to force errno to be defined as a macro.

Those who do not want to define errno as a macro can do:

```
#define errno errno
```

C++ already requires that errno be defined as a macro.

4.25 Parallel memory Sequencing Model, (N1284) (Nelson)

See 4.8

4.26 namespace.html (Stoughton) from wiki

Several recent proposals for the C1x revision have introduced new keywords, functions, or macros that require the implementation to "do the dance": the keyword (function or macro) is actually named _[UpperCaseChar]name, and a header is required to define the natural spelling of the keyword. For example, _Thread_local, which can be spelled thread_local if the correct header (<thread.h>) is included.

An alternative approach to "doing the dance" is proposed here. It is not intended that necessarily every occurence of _[UpperCaseChar]name need be replaced by this mechanism. It simply is intended to increase our arsenal if methods when we need to introduce such new keywords etc.

Proposes to reserve stdc_, STDC_, and __STDC__ for future standardization.

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

4.27 errno.txt (Tydeman) from wiki

Draft proposal from Fred to require that errno not be altered if there is no error.

No. Functions are allowed to alter the value of errno, even w/o an error.

5. DEFECT REPORT REVIEW

5.1 Two OPEN Defect Reports

DR334
Clause 7.12.14, Comparison Macros (and subsections) are missing 'Semantics', and likely need further additional words for clarification. A resolution may be to add this material to the WP.

We do not have an N paper, or proposed text, yet. The type-generic work being done by PJ/Nick should fix this. PJ agreed, and make this behave the same as any other type-generic function. This will then be proposed as an addition to the WP, and reviewed as such.

Add some words that for C99 these macros are fragile.

Moved to REVIEW

DR340

The definition of composite types in 6.2.7#3 says "If one type is an array of known constant size, the composite type is an array of that size; otherwise, if one type is a variable length array, the composite type is that type." and also "These rules apply recursively to the types from which the two types are derived.". Which of these wins for variable length array types? Are the element types composed recursively, or is the element type of the variable length array type taken even though it may have less information than the other element type? (That loss of information in the composite type would mean some sequences of three or more declarations of the same function are constraint violations for some orderings of the declarations and undefined behavior for other orderings.)

See reflector messages 11145-11147 for discussion.

Delft: It appears that current implementations differ in this area. Some compilers work one way, as described above, while others do not. Consideration to make this undefined behavior.

Clarification of Composite Type N1324 (Keaton)
Address DR 340, and provides wording that can be used for the WP for DR 340. Editorial issue with the proposed words? Yes, some minor edits needed, and agreed to.

ACTION: David K to make a run at final words containing the editorial changes to DR340 and post to the wiki by Thursday morning.

David provided the words they were accepted for addition to DR340.

Moved DR340 to REVIEW

5.2 REVIEW / RESOLVE DEFECT REPORTS

5.2.1 DRs in REVIEW

DR345 – Moved to **CLOSED**

DR342 – Sticks to **DR340**

DR338 – Moved to **CLOSED**. Will be added to WP, and the internal TC.

DR329 – Moved to **CLOSED**. Mostly changes to Annex F.

DR314 – See Also N1226. The words in N1226 differ from the words in DR314. Randy wants to use the response for #1 contained in N1226. All declarations of the same thing have to be compatible. Expressed support that N1226 response #1 is better than the proposed response in the DR. Response #3, Randy believes the Standard actually leads the reader down the garden path making the reader believe it works. We don't really want it to do what a literal translation of the words say.
Question #2, we want the answer to be brief, although the answer in N1226 in really better.
Question #3, take up as a revision. If you go through the current rules, it could be interpreted to mean that the Standard does require such behavior, but it's not what we really want the Standard to say. It was never our intention that the standard be interpreted that way. Change the response to reflect that.

Leave in **REVIEW**, reworded as above.

6. REVIEW

6.0.1 Mid-meetings on memory model

We want to get a memory model added to the revision, so a small group is being put together that can meet via phone conference to work on this. Meetings will be announced via the reflector.

6.0.2 Web Hosting

SC22 is moving the SC22 web site from Keld's site to Live Link. We want to maintain control over the WG14 web site, and Live Link does not give us that. Dinkumware has been doing a backup of Keld's site. Consider keeping WG21 on the same system if possible. Nick already has obtained a URL for sc22wg14.org, and will apply to get one for sc22wg21.org.

6.0.3 Email Reflector

If we move our web site, we'll also want to move the email reflector. We're not sure who owns or runs the machines we are using for the reflector. No decisions yet, nor is there any rush, but something we should all be thinking about

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:

N1311, Initializing Static or External Variables
N1310, Requiring signed char to have no padding bits.
N1326, TR19769
N1327, Abandoning a Process
N1330, Static Assertions
N1333, Unicode and Raw String Literals
N1316, Conversion between pointers and floating types
N1319, Adding EPOLE to the WP (the description of 'pole error' only)
N1338, More thoughts on Implementing errno (modulo adding words to insure that errno is reserved in the external namespace)
N1320, Integrating C89 DR25 into the WP
N1321, Split FLT_EVAL_METHOD into operations and constant, modulo removing const from FLT_EVAL_METHOD.

6.2 Review of Action Items

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

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

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.

ACTION: Tom Plum to develop a fully-formed proposal for try-finally.

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

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

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

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

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

ACTION: Convenor to forward, N1337 DTR24731-2, to SC22 for PDTR Ballot.

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

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

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

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".

ACTION: Ulrich and Nick 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 (N1345).

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

ACTION: Clark to revise N1284 for a review committee.

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

ACTION: Tom and Arjun revise N1332, proposal for new library APIs for EncodePointer() and DecodePointer().

ACTION: Arjun to revise N1335 to add alignof

7. CLOSING

7.1 Thanks to Host

Thanks to Mark Terrel for a great job.

8. ADJOURNMENT

Meeting adjourned at 10:15, Thursday, 11 Sept 2008

PL22.11 Meeting
9 September 2008

Meeting convened at 1630 hours by Char, Randy Meyers

Attendees:

John Benito	Blue Pilot	
Larry Jones	Siemens PLM Software	Project Editor
David Svoboda	CMU/SEI	
Mark Terrel	Cisco	
Tana L. Plauger	Dinkumware, Ltd	
P. J. Plauger	Dinkumware, Ltd	
Rich Peterson	Hewlett Packard	
Edison Kwok	IBM	
John Parks	Intel	
David Keaton	Self	
Arjun Bijanki	Microsoft	
Jeff Muller	Oracle	
Barry Hedquist	Perennial	Secretary
Keith Derrick	Plantronics	
Tom Plum	Plum Hall	
Bill Seymour	Tydeman Consulting	
Clark Nelson	Intel	
Randy Meyers	Silverhill Systems	PL22.11 Chairman
Fred Tydeman	Tydeman Consulting	PL22.11 Vice chairman
Nick Stoughton	Usenix	
Hans Boehm	Hewlett Packard	
Ulrich Drepper	Red Hat	
Blaine Garst	Apple	
Bill Bumgarner	Apple	
Douglas Walls	Sun Microsystems	PL22.11 IR
John Hauser	Self	

Agenda Approved as modified.

1. Select US delegation for the next two meetings.

Motion to name the following to the US Delegation: Douglas Walls, David Keaton, Barry Hedquist, and Keith Derrick. (Benito, Walls)
15-0-1 MOTION PASSES

2. Report on INCITS Officers Symposium (Plum)

We do most of our business as a WG, and will likely continue in this mode. SC22 guidance no longer places a restriction on delegation size. At the symposium, the emphasis was that there was no intent to

changing the way we, PL22.11, operate w/r/t our operation as J11. So, it's not likely that we need to make any changes to our delegation size.

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 that criteria.

5. Anti-Trust

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.

6. Adjournment

Meeting adjourned at 17:03.