WG14 N1604
INCITS PL22.11/12-001
Reply To The Attention Of: Barry Hedquist
PL22.11 Secretary
Email: beh@peren.com

ISO/IEC JTC1 SC22/WG14 AND INCITS PL22.11
MEETING MINUTES (DRAFT), Feb 13-15, 2012

Meeting Location:
Royal Kona Resort
75-5852 Alii Drive
Kailua-Kona HI 96740
United States
Meeting information:
N 1574

Local contact information:
Thomas Plum
Plum Hall Inc.
3 Waihona Box 44610
Kamuela HI 96743 USA
telephone: +1-808-882-1255
Fax: +1-808-882-1556
Email: tplum@plumhall.com

Scheduled Meeting Dates: 13-17 Feb, 2012

Scheduled Meeting Times:
13 Feb 2012 09:00 am to 4:30 pm
14 Feb 2012 09:00 am to 4:30 pm
15 Feb 2012 09:00 am to 4:30 pm
16 Feb 2012 09:00 am to 4:30 pm
17 Feb 2012 09:00 am to 12:00 pm

Teleconference information:
Topic: WG 14 Feb 2012
Date: Every 1 day, from Monday, February 13, 2012 to Friday, February 17, 2012
Time: 11:00 am, Pacific Standard Time (San Francisco, GMT-08:00)
Meeting Number: 956 445 926
Meeting Password: wg14

To start or join the online meeting, go to iso-meetings

Audio conference information:
To receive a call back, provide your phone number when you join the meeting, or call the number below and enter the access code.

- Switzerland toll free: 0800-894627
- USA/Canada toll free: 1-855-299-5224

Having trouble dialing in? Try these backup numbers:

- Call-in toll-free number (UK): 0800-051-3810
- Call-in toll number (UK): +44-20-310-64804
- Global call-in numbers: call-in numbers
- Toll-free dialing restrictions: tollfree restrictions

Access code: *956 445 926*

For assistance:

1. go to ios meeting support
2. On the left navigation bar, click "Support".

To add this meeting to your calendar program (for example Microsoft Outlook), click this link: iso meeting to calendar

To check whether you have the appropriate players installed for UCF (Universal Communications Format) rich media files, go to ios meeting diagnostics

http://www.webex.com

1. OPENING ACTIVITIES

1.1 Opening Comments (N1572) (Plum, Benito)

John Benito and Tom Plum welcomed us to the Royal Kona Resort, and described the meeting facilities. Several local restaurants are within walking distance of the hotel. Lunch break will be from 11:30 - 13:00. We are connected on WebEx to allow folks to call in. This meeting is hosted by ANSI, Bloomberg and Plum Hall. There is a restroom key for the restrooms. Refreshments are available in the back of the room.

1.2 Introduction of Participants/Roll Call

Name	***Organization***	***NB***	***Comments***
John Benito	Blue Pilot		WG14 Convener
Blaine Garst	Self	USA	
David Keaton	CMU/SEI/CERT	USA	PL22.11 Chair
Robert Secord	CMU/SEI/CERT	USA	
David Svoboda	CMU/SEI/CERT	USA	
Tana L. Plauger	Dinkumware, Ltd	USA	
P. J. Plauger	Dinkumware, Ltd	USA	
Jim Thomas	HP	USA	
Rajan Bhakta	IBM	Canada	HoD - Canada
Mike Hennell	LDRA	USA	
Douglas Walls	Oracle	USA	HoD – USA, PL22.11 IR
Barry Hedquist	Perennial	USA	PL22.11 Secretary

Tom Plum	Plum Hall, Inc.	USA	
Fred Tydeman	Tydeman Consulting	USA	PL22.11 ViceChair
Larry Jones	Siemens	USA	WG14 Project Editor
Bill Seymour	Seymour	USA	
Douglas Gwyn	Self	USA	
Nick Stoughton	IRDETO	USA	POSIX Liaison
Roger Scott	Coverity	USA	
Martin Sebor	Cisco	USA	
Clark Nelson	Intel	USA	
Martin Sebor	Cisco	USA	
Dave Prosser	Bloomberg	USA	
Nevin Liber	DRW	USA	
Herb Sutter	Microsoft	USA	
Willem Wakker	ACE	Netherlands	HoD - Netherlands

1.3 Procedures for this Meeting (Benito)

The Meeting Chair, John Benito, announced the procedures are as per normal. Everyone is encouraged to participate in straw polls.

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 by each National Body.

INCITS PL22.11 members reviewed the INCITS Anti-Trust Guidelines at:

http://www.incits.org/inatrust.htm.

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

Emphasis for this meeting is to consider defect reports and future Technical Specifications for WG14.

John Benito, WG14 Convener, is the meeting Chair. Barry Hedquist, PL22.11 Secretary, is the Recording Secretary for the meeting.

1.4 Approval of Previous Minutes - Washington (N1588) (Hedquist)

Several comments for typos, etc.

Minutes were modified per editorial changes and approved.

Final Washington Minutes are **N1603**

1.5 Review of Action Items and Resolutions (Hedquist)

ACTION: Convener to determine National Body interest in a security focused static analyzer, and if sufficient, to forward a NWI for a Technical Specification to SC22.
DONE - will vote this meeting.

ACTION: Robert Secord to make sure the Draft Technical Specification for C Secure Coding Rules is available to WG14 committee members prior to the pre-Kona mailing.
CLOSED - used prior version for the SC22 NP request.

ACTION: Blain Garth to work with Mark Batty to clarify DR 408.
DONE - Standard is OK as is.

1.6 Approval of Agenda (N1597) (Benito)

Revisions to Agenda: None

Added Items: None

Deleted Items: None

Agenda approved by unanimous consent.

1.7 Identification of National Body Delegations (Benito)

US, Canada, Netherlands.

1.8 Identification of PL22.11 Voting Members (Tydeman)

See PL22.11 Minutes, following these minutes. 15 of 15 members present.

2. LIAISON ACTIVITIES

2.1 SC22/WG14 & INCITSPL22.11 C Language (Benito, Walls, Keaton)

The revision to the C Standard has been published as ISO/IEC 9899:2011. PL22.11 will hold a meeting, Tuesday, Feb 14 at 4PM, to establish US Positions for 1) an NP for Secure Coding, and 2) Technical Report on Numerical C Extension. There is an SC22/WG14 NP in process, Secure Coding.

2.2 SC22/WG21 & INCITS/PL22.16, C++ Language (Plum, Benito)

The ISO/IEC 14882, the revision to C++, was approved last year, and has now been published. The new revision of C++ is known as C++ 2011. SC22/WG21 met last week in Kona to address future directions of the C++ Standard, and process Defect Reports.

Two items from WG21: Cilk presentation, and transactions. They are included in the document review portion of the agenda. Both introduce new keywords.

Digit separators also coming back. Possible solutions under discussion in C++, could cause possible divergence.

Three DR's in CWG that break C. Plain int bitfields are required to be signed.
CWG Issue numbers: 739, 1438, 1457. See "Of Potential Wider Interest" in WG21 Core Report.

2.3 SC22 & INCITS/PL22, Programming Languages (Plum)

PL22 meets twice a year via teleconference. Next teleconference: June 20, 2012. See Tom Plum, PL22 Chair, for details.

2.4 WG23 (Benito)

A revision to the TR is underway. WG23 met in Madrid, Spain in March 2011.
For further information, contact John Benito. WG 23 will meet next in March 2012, in Ottawa.

2.5 MISRA C (Montgomery) (N1600)

N1600 is a Liaison Report updating us on the latest activities of MISRA. The third version of the MISRA C Guidelines has been finalized, and will soon be released for public comment. The comment period will close at the end of April, 2012, and the final document should be published sometime in Q4, 2012.

2.7 Floating-point Study Group (Thomas)

A Working Draft Technical Specification, N1591, has been submitted and is on the agenda (See 6.1)

2.8 Secure Coding Study Group (Keaton)

The document will be distributed after the NP Ballot has closed for a comment period OR we could have a WG14 meeting shortly after that to allow for face to face meeting to discuss comments internal to WG14. After WG14 has reviewed and drafted the document to it's satisfaction, it will go out as a PDTS SC22 Ballot. Douglas (US) wants at least 60 days. JB - that won't work for his schedule. Rajan (Canada) expressed the same concern. Wakker (Netherlands) likewise. Plan for 90 days.

2.9 Other Liaison Reports - none

3. EDITOR REPORTS

3.1 Report of the Project Editor (Jones)
ISO/IEC 9899:2011 has been published.

4. Teleconference Meeting Reports

4.1 Report on any teleconference meetings held (Benito)
None held.

5. FUTURE

5.1 Future Meeting Schedule

Portland, OR, USA – 22, October 2012, see N1592.
Spring 2013 – OPEN (WG21 will meet in Bristol, 15 April) WG14 does not have a host, yet. We will try to meet on one side or another of the WG21 meeting. ACCU conference will precede the C++ meeting (Thurs, Fri, Sat).

Fall 2013 – Chicago, IL, USA – Dates not set. LDRA will host, first 2 weeks in October. (C First??)

Feb or Jun 2014 WG21 will meet at Rapperswil.

5.3 Future Mailings

Post Kona – 16 March 2012
Pre Portland – 24 September 2012
Post Portland – 19 November 2012

6. Document Review

6.1 Floating-point extensions for C — Part 1: Binary floating-point arithmetic (N 1591).

Rajan also taking notes

N1591 is draft Technical Specification for binary floating-point arithmetic.
Eventually will consist of five parts.
1. Binary FP - suggested changes to C11
2. Decimal FP - revision to existing DFP TR
3. NEW INTERCHANGE AND TYPES IN THE NEW FP STANDARD
4. SUPPLEMENTAL FUNCTIONS
5. SUPPLEMENTAL ATTRIBUTES

All of which are C Bindings to C11. The current draft addresses Part 1. The draft is functionally complete.

Some issues to discuss.

1. Should we define conformance for freestanding implementations too? Note that IEC 60559 requirements for conversions between floating-point formats and decimal character sequences are met in <stdio.h>. We could define strfrom functions, which would meet the requirements of IEC 60559 without requiring <stdio.h> support.

If adopted, it should be conditional (optional). There will be a macro for conformance to Part 1, Part 2, etc. Freestanding implementations do not require <stdio.h>, but some of the functions here do, so an allowance of some sort needs to be made. MISRA supports the concept. Consensus is to allow for freestanding implementations.

2. Do we need a WANT macro to guard the new interfaces? Should each part of the TS have its own WANT macro, e.g., __STDC_WANT_IEC_60559_BFP__? Yes, add WANT macros as appropriate.

3. F.3 - Replace with an improved table? (Table in draft). The table seems to lack some of the clarity in the original text. Where will that information go? In the details for the functions concerned. Add a third column with a reference to the clause containing the details.

4. Clause 7.6.2, Rounding Control Pragmas ISSUE: While 754 doesn't make requirements for flags beyond the global flags in C, it does allow for what might be called "local" flags, instead of, or in addition to, global flags. The motivation behind constant rounding direction attributes is that accesses to dynamic modes become synchronization points - which inhibits auto parallelization. This problem seems to exist for global flags too. Do we need to add support for local flags? Or should this issue be deferred to the alternate exception handling specification in Part 5.

RESOLUTION: Defer to Part 5 (Alternate exception handling). (7.6.2, Rounding Control Pragmas)

5. Calls made to function pointers. All function calls are function pointer calls. Needs to be reconsidered. No consensus.

6. Integer width macros - Is this where it belongs? Could be outside the Floating Point TS. Consensus to keep it as part of the floating point TS.

7. Withdrawn - names need work, further effort needed here.

8. Clause 7.12.6.7 ISSUE: Should the 7.12 specification of llogb require a domain error for finite out-of-range cases? The C committee didn't want this for ilogb. Defer this until we resolve Fred's paper on errors for llogb. Defer this to discussion of N1595. Leave as is for now.

Are we (the SG and WG14) ready to process an NP based on this document? Jim would like to move forward with this document. We have 2 NBs that will vote NO if a document does not accompany the NP.

ACTION: Convener to process an NP (NWI) through SC22 for a Technical Specification: Floating-Point Extensions for C: Part 1, Binary floating-point arithmetic.

Aside: Keaton asked that the FP folks take a look at N1579 to make sure they did nothing wrong in their floating point material.

6.2 Possible defect in <math.h> – f(inf) is inf being a range error (Tydeman) (N 1593).

N1593 suggests that 'infinity' could be regarded as 'too large', and thus considered a 'range error' for certain math functions in C11. Specifically, those that state "A range error occurs if the magnitude of X is too large." Wording is proposed for each of the affected functions to clarify the intent.

Different implementations take different approaches. Nailing this down will 'cost' somebody.

Should N1593 be added to the Defect Report List?
Straw Poll: 7-5-0 : YES - DR 409.

Discussion of DR 409
Is infinity 'too large'? It is representable, but it is larger than things that are 'too large'. The Standard allows the current use. Doug suggests we may need to redefine 'range error'. There does not seem to be any consensus to make any changes. Fred would like to add 'finite', but Doug pointed out that could cause other issues. POSIX says, "If the correct value would cause

overflow, a range error shall occur." There does not seem to be consensus on adopting the proposed words.

STRAW POLL: Adopt the proposed words in DR 409 (1 - 12 - 0).

Leave OPEN.

Fred presented new words.
Bill does not think they are needed. The words already exist in the Standard to cover what Fred has proposed. DAVE AGREES WITH BILL!! Bill is OK with a clarification, but in only one place, rather than peppered throughout the math library. Dave points out that the statement in the Standard (7.12.1) "...except as specified otherwise.." does require that unique situations be specified. A footnote to state that infinity (inf) is not considered a range error? May of may not be treated as a range error? The preferred interpretation is that they NOT be treated as range error. Recommended practice? There is an existing footnote, work off that? No consensus for action at this time.

Yet newer words from Fred.

The committee does not like the author's suggested change.

The committee considered the following, but rejected it (as just being a restatement of 7.12.1 paragraphs 4 and 5).

If the result overflows, a range error shall occur.

A question arose as to why these range error cases are listed in the individual functions (instead of just being covered by the blanket 7.12.1 paragraphs 4, 5, and 6)

7.12.1 paragraph 1 has the answer:
The behavior of each of the functions in <math.h> is specified for all representable values of its input arguments, except where stated otherwise.

One idea: Add a footnote to 7.12.1 paragraph 5, first sentence:

In an implementation that supports infinities, a range error may happen for functions that map an infinity argument into an exact infinity or exact zero result.

Another idea: Add to end of 7.12.1 paragraph 4:

Recommended practice:

In an implementation that supports infinities, a range error should not happen for functions that map an infinity argument into an exact infinity or exact zero result.

Yet, another idea: Add to 7.12.1 paragraph 4:

An implementation may define additional range errors, provided that such errors are consistent with the mathematical definition of the function.

This DR will remain OPEN for now.

6.3 Possible defect in <math.h> – Missing domain errors. (Tydeman) (N1594)

N1594 suggests that a number of math functions in C11 fail to specify conditions for which a domain error occurs. Wording is proposed for each function.

Should N1594 be added to the Defect Report List?
Straw Poll: 2-10-0: NO

6.4 Possible defect in <math.h> – ilogb inconsistent with lrint and lround (N1595).

N1595 suggests an inconsistency exists in C11 between the requirements for ilogb and those for lrint and lround. The function ilogb does not allow for a range or domain error, while both lrint and lround do. The paper proposes to allow for a range or domain error for ilogb.

Should N1595 be added to the Defect Report List?
Straw Poll: 11-4-0 YES - DR 410

Discussion of DR 410
The effect is no change, so this is OK or NOT. However, the rationale presented as a reason for doing this is NOT regarded as a valid rationale for the change. No change?

STRAW POLL: Add these new proposed words? (8-4-0)

6.5 Preliminary schedule for SC TS (Keaton) (N1596).

N1596 proposes a schedule for the processing of a Technical Specification for Secure Coding. Key dates in the schedule are:

NP and Registration Ballot Submission (SC22) (this date will NOT happen)	March 20, 2012
Draft Technical Specification Ballot Submission (SC22)	Nov 13, 2012
Final Technical Specification Ballot Submission (JTC1)	April 2013
Publication (ITTF)	End of 2013

6.6 Predefined macro values (Jones) (N1598).

The SC22/WG21 Project Editor for C11 points out the C11 macros __STDC_VERSION__ and __STDC_LIB_EXT1__ do not have values assigned to them, and proposes integer constant 201112L for both.
Should N1598 be added to the Defect Report List?

Is this a defect? It is just wrong, and should be errata.

ACTION: Convener to request N1598 be processed as an “errata”.
Also make it a Defect Report, DR 411, and post process as needed.

Discussion of DR 411
See above.

6.7 #elif issue (Jones) (N1599).

The SC22/WG21 Project Editor for C11 points out the description for #elif does not seem to match the intent of that macro, and proposes words to correct the description.

Brought it up a long time ago. We always meant it to be the same as #if, i.e. cannot be ignored. Dave says this was an oversight of the editor (him) at that time.

Should N1599 be added to the Defect Report List?
Straw Poll: 15 - 0 - 0: YES DR 412

Discussion of DR 412

Suggested Technical Corrigendum
In 6.10.1p6, change:
Only the first group whose control condition evaluates to true (nonzero) is processed.
to:
Only the first group whose control condition evaluates to true (nonzero) is processed; any following groups are skipped and their controlling directives are processed as if they were in a group that is skipped.

STRAW POLL: No Objection to adopting words for a future TC, remains OPEN

6.8 initialization (Wakker) (N1601).

N1601 asks for clarification in determining the value of an element in a struct when that element may have been affected by an incomplete initialization.

Summary
Consider the following code:

```
typedef struct {
int k;
int l;
int a[2];
} T;

typedef struct {
int i;
T t;
} S;

T x = {.l = 43, .k = 42, .a[1] = 19, .a[0] = 18 };

void f(void)
{
S l = { 1, .t = x, .t.l = 41, .t.a[1] = 17};
}
```

The question is: what is now the value of l.t.k? Is it 42 (due to the initialization of .t = x) or is it 0 (due to the fact that .t.l starts an incomplete initialization of .t?

The relevant clause from the standard is 6.7.9 clause 19:

19 The initialization shall occur in initializer list order, each initializer provided for a particular subobject overriding any previously listed initializer for the same subobject;[151]) all subobjects that are not initialized explicitly shall be initialized implicitly the same as objects that have static storage duration.

Should N1601 be added to the Defect Report List?
Straw Poll: 9-0-0: YES - DR 413

Discussion of DR 413

Add a clarifying example to the Standard? There are no proposed words at this time. There are ragged edges here. Doug likes the idea of an example to clarify what we intended. What should the answer be? Dave has no problem with adding examples to make the Standard clearer, but existing words do not to be changed. Some disagree, and see two ways to interpret the same sentence. 6.7.9#19. What constitutes 'override'? Clark is less than convinced the Standard is clear about that. We need some proposed words/examples. Clark has an example, but Dave sees it as a separate question that could lead to future DRs. Doug believes that Clark's point has more to do with footnote 151 within 6.7.9#19, and could actually be another DR.

ACTION: Doug Gwyn and Willem Wakker to propose words and examples for DR 413.

Willem Wakker proposed words: Replace the text in 6.7.9#19 with the following:

"The initialization shall occur in initializer list order, each initializer provided for a particular subobject overriding any previously listed initializer for the same subobject [fn151)].

Subsequently, all subobjects that are not initialized explicitly previously shall be initialized implicitly the same as objects that have static storage duration."

The phrase "Particular subobject" needs clarification.

DR 253 looks to be closely related to this DR. Same clause and paragraph. We did not make a change to the Standard.

ACTION: Willem and Blain to work on further proposed words for DR 413.

6.9 Constant Problems (N1602)

Blain: This paper essentially questions the lexablility of the language. Doug believes the grammar is correct, but there is one item presented that we could use - hexadecimal constant.

6.10 Parallelism (Nelson) (WG21/N3361)

Clark Nelson provided a presentation made to WG21 (C++) designed to introduce parallel programming constructs to C++. The PDF document is available on the WG14 Wiki, and the SC22/WG21 web site. On the WG14 Wiki it is named "N3361-Parallelism_Talk.pdf". Additional information can be obtained at www.cilk.org. WG14's interest in this material would for similar kinds of programming constructs for C. Doing so would be important if WG21 decides to move in this direction and adopt these features for standardization. All the capabilities in the presentation are implemented.

6.11 Transactional Memory (Nelson) (SC22WG14_12599)

Material on transactional memory, like the information above, is under consideration for adoption by WG21. For C/C++ compatibility reasons, we should track the progress of discussions in this area and give consideration to adopting compatible mechanisms.

Blain expressed some concern about whether we were treading into an area of invention here, but for these technologies that does not seem to be the case. They have existing implementations, and have been in existence for the last 10 - 15 years. Clark sees the priority as solving and simplifying the parallel programming problem. JB sees this activity (parallelism) as a separate Standard, rather than part of the C Standard. Clark thinks it's too early to think about standardizing transactional memory. Why didn't we think about this for C11? Although the Cilk material represents existing practice, it was not in the state it is in today while we were working on C11.

7. Defect Reports

DR 400

Summary:

There are at least three existing realloc behaviors when NULL is returned; the differences only occur for a size of 0 (for non-zero size, all three implementations set errno to ENOMEM when returning NULL, even though that is not required by C99).

Suggested Change

At 7.22.3, Para 1, change:

If the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned, or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.

to

If the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned and errno set to indicate the error, or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.

Add a footnote to this sentence stating:

Note Memory allocated by these functions should be freed via a call to free, and not by means of a realloc(p, 0).

Feb 2012 Discussion.

Tom believes it is too late to make this change. Nick believes there is a need to know what is going to happen in this case. One of the problems is that people use realloc (0) instead of free(). The problem is in not knowing whether or not memory has been freed when NULL is returned. Doug believes we should focus on what a failure means. Dave: realloc (0) has never meant "free up the memory", i.e. it never means free(). David suggests: If a null pointer is returned, it is an error. Larry likes that suggestion. If we make it 'implementation-defined' we can possibly preclude breaking anyone's implementation, and require it to be documented. However, that means any behavior could be considered conforming.

ACTION: Nick to work with David and Martin to explore new words. DONE

Suggested Change

Alternative A: Implicit implementation defined

Change 7.22.3 Para 1

If the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned, or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.
to
For the aligned_alloc, calloc, and malloc functions, if the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned (indicating that the space cannot be allocated), or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.

At 7.22.3.5 insert a new paragraph after Para 3:

If the size of the space requested is zero, the behavior is implementation-defined: either a null pointer is returned, or the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object. If a null pointer is returned and errno is set to an implementation defined non-zero value, the old object is not deallocated and its value is unchanged.

Alternative B: Explicit implementation defined
As above for 7.22.3 Para 1. At 7.22.3.5 insert a new paragraph after Para 3:
If the size of the space requested is zero, the behavior is implementation-defined from the following list:

- a null pointer is returned and the old object is deallocated

- a null pointer is returned and the old object is not deallocated and its value is unchanged

- the behavior is as if the size were some nonzero value, except that the returned pointer shall not be used to access an object.

Alternative C: Free for all
As above for 7.22.3 Para 1. At 7.22.3.5 insert a new paragraph after Para 3:
If the size of the space requested is zero, the behavior is implementation-defined.

Three basic alternatives. Doug sees additional alternatives, and those presented favor implementers rather than programmers. Dave P wants to see consistency across all memory allocation functions. The alternate words do not address that. Larry points out that all of these alternates bless behavior that we have always said was wrong to begin with. Doug believes we need to address the misunderstanding and clarify our original intent. However, enforcing that rule will break many implementations. Doug believes that not enforcing the rule will also break implementations. Will implementations change to follow our intent? Not likely. Clark likes making the result I-D, because it shines a light on what implementations actually do. The real goal is to get people to write portable code particularly w.r.t using zero (0) as an argument.

Submission by David Keaton

7.22.3p1:

Change

"If the size of the space requested is zero, the behavior is
implementation-defined: either a null pointer is returned, . . ."

to
"If the size of the space requested is zero, the behavior is
implementation-defined: either a null pointer is ~~always~~ returned to
indicate an error, . . ."

7.22.3.5p3:

Change:
"If memory for the new object cannot be allocated, the old object is
not deallocated and its value is unchanged."

to
"If memory for the new object cannot be allocated, the old object is
not deallocated and its value is unchanged, except that if the size of
the space requested is zero, it is implementation-defined whether the
old object is deallocated."

Tom points out that memory leaks can occur if we 'fix' the problem, i.e. disallow the use of realloc to free memory, i.e. realloc(0). glibc is the implementation most affected. Consider merging David and Doug's proposed wording.

Submission by Doug Gwynn

Doug Gwyn's proposed technical corrigendum to the C standard to address DR400:

In subsection 7.22.3.5 (The {realloc} function), change the final sentence of paragraph 3 from:

> If memory for the new object cannot be allocated,
> the old object is not deallocated and its value is unchanged.

to:

> If {size} is non-zero and memory for the new object is not allocated,
> the old object is not deallocated.
>
> If {size} is zero and memory for the new object is not allocated, it is implementation-defined whether the old object is deallocated.
>
> If the old object is not deallocated, its value shall be unchanged.

In subsection 7.22.3.5 (The {realloc} function), change paragraph 4 from:

> The realloc function returns a pointer to the new object
> (which may have the same value as a pointer to the old object),
> or a null pointer if the new object could not be allocated.

to:

> The realloc function returns a pointer to the new object

(which may have the same value as a pointer to the old object),
or a null pointer if the new object has not been allocated.

Doug likes David's proposal, and David likes parts of Doug's. Consider merging the two. First part of David's with all of Doug's. Martin likes the errno approach, but others see that as 'endorsing' the existing wrong practice. Add an example showing what NOT to do. "This is NOT portable!" Making such use I-D at least makes the implementer document the misuse.

ACTION: David and Doug to merge their two proposals for DR 400. DONE

New merged words:

In subsection 7.22.3 paragraph 1, change

"If the size of the space requested is zero, the behavior is implementation-defined:
either a null pointer is returned, ..."

to

"If the size of the space requested is zero, the behavior is implementation-defined:
either a null pointer is returned to indicate an error, ..."

In subsection 7.22.3.5 (The {realloc} function), change the final sentence of paragraph 3 from

"If memory for the new object cannot be allocated, the old object is not deallocated and its value is unchanged."

to

"If {size} is non-zero and memory for the new object is not allocated, the old object is not deallocated.

If {size} is zero and memory for the new object is not allocated, it is implementation-defined whether the old object is deallocated.

If the old object is not deallocated, its value shall be unchanged."

In subsection 7.22.3.5 (The {realloc} function), change paragraph 4 from

"The realloc function returns a pointer to the new object (which may have the same value as a pointer to the old object), or a null pointer if the new object could not be allocated."

to

"The realloc function returns a pointer to the new object (which may have the same value as a pointer to the old object), or a null pointer if the new object has not been allocated."

Add to subsection 7.31.12 a new paragraph (paragraph 2):

"Invoking realloc with a size argument equal to zero is an obsolescent feature."

STRAW POLL: Add the new proposed merged words to DR 400 (13-2-0)
DR 400 will remain OPEN, and will be reviewed in October.

DR 401

C++11 forbids *happens before* from being cyclic, but this change has not made its way into C11. In order to fix this, the following sentence (taken from C++ N3291, 1.10p12) should be added to 5.1.2.4p18:

The implementation shall ensure that no program execution demonstrates a cycle in the "happens before" relation.

NOTE: This cycle would otherwise be possible only through the use of consume operations.

Oct 2011 meeting:
Seems as if C++ made this change at the last minute and WG 14 had already voted a document for ballot. There seems to be consensus to make a change along this line.

Feb 2012 meeting:
Douglas: What's the definition of a cycle? Not defined, per se, but 'happens before' is defined in C11. Defining what a cycle is seems to be elusive. Clark: this applies to thread operations and the sequencing/synchronization of a multithreaded process.

Suggested Technical Corrigenda: Add to 5.1.2.4p18:

The implementation shall ensure that no program execution demonstrates a cycle in the "*happens before*" relation.

STRAW POLL: Move DR 401 to REVIEW. (13-0-0)

DR 402

This is another edit that made it into C++ 11, but not C11.

Subject: memory model coherence is not aligned with C++11

Summary
The memory model described in N1569 matches an older version of the C++0x memory model, one that allowed executions that were not intended by the designers. The recommendation is to match the C++11 text by removing the sentence starting 'Furthermore' in 5.1.2.4p22, and including the following paragraphs in section 5.1.2.4 (Taken from C++ N3291, 1.10p15 through 18):

If an operation A that modifies an atomic object M happens before an operation B that modifies M , then A shall be earlier than B in the modification order of M .

NOTE: The requirement above is known as write-write coherence.

If a value computation A of an atomic object M happens before a value computation B of M , and A takes its value from a side effect X on M, then the value computed by B shall either be the value stored by X or the value stored by a side effect Y on M, where Y follows X in the modification order of M.

NOTE: The requirement above is known as read-read coherence.

If a value computation A of an atomic object M happens before an operation B on M, then A shall take its value from a side effect X on M, where X precedes B in the modification order of M.

NOTE: The requirement above is known as read-write coherence.

If a side effect X on an atomic object M happens before a value computation B of M, then the evaluation B shall take its value from X or from a side effect Y that follows X in the modification order of M.

NOTE: The requirement above is known as write-read coherence.

Suggested Technical Corrigendum
See above.

Feb 2012 Meeting:
STRAW POLL: Move DR402 to REVIEW (11-0-0)

DR 403

Summary
The synchronization afforded to malloc and free is missing some vital ordering, and as the definition stands it makes little sense and creates cycles in happens before. C++11 includes a total order over the allocation and deallocation calls, which fixes the problem and seems to give a sensible semantics. From 18.6.1.4p1 in N3291:

Calls to these functions that allocate or deallocate a particular unit of storage shall occur in a single total order, and each such deallocation call shall happen before the next allocation (if any) in this order.

Unfortunately, there is a typo here. Happens before edges are not transitively closed in to the happens before relation, but the edges here are meant to be. Instead the sentence above should create a synchronizes with edge. In light of this, we suggest removing the last two sentences from 7.22.3p2 and replacing them with:

Calls to these functions that allocate or deallocate a particular region of memory shall occur in a single total order, and each such deallocation call shall synchronize with the next allocation (if any) in this order.

Suggested Technical Corrigendum
See above.

Oct 2011 meeting
The consensus was that this is an oversight, and should be changed along the lines that are recommended.

Feb 2012 meeting
STRAW POLL: Move DR 403 to REVIEW. (12-0-0)

DR 404

Subject: joke fragment remains in a footnote

Summary
C11 seems to have inherited part of a joke from C++, which ought to be removed or made whole and annotated as such. Originally, C++0x had the footnotes:

"Atomic objects are neither active nor radioactive" and "Among other implications, atomic variables shall not decay".

The first is pretty clearly a joke, but it's not obvious that the second doesn't have some technical meaning, and that is the one that remains in C11 in 7.17.3p13.

Suggested Technical Corrigendum
See above.

Oct 2011 meeting
It is not clear that rewording will make the footnote useful.

Feb 2012 meeting
Better to remove the footnote.
STRAW POLL: Remove the footnote, leave OPEN to next meeting.
(12-0-0)

DR 405

Subject: The mutex specification

Summary
The C11 specification of mutexes is missing the total order over all the calls on a particular mutex. This is present in C++11. The following is from 30.4.1.2p5 in N3291:

For purposes of determining the existence of a data race, these behave as atomic operations (1.10). The lock and unlock operations on a single mutex shall appear to occur in a single total order. [Note: this can be viewed as the modification order (1.10) of the mutex. — end note]

The synchronisation in 7.26.4 is defined in terms of some order over these calls, even though none is specified, for instance 7.26.4.4p2 reads:

Prior calls to mtx_unlock on the same mutex shall synchronize with this operation.
This seems like simple omission. We suggest adding a new paragraph to 7.26.4 that matches C++11:

For purposes of determining the existence of a data race, mutex calls behave as atomic operations. The lock and unlock operations on a single mutex shall appear to occur in a single total order.

NOTE: This can be viewed as the modification order of the mutex.

Suggested Technical Corrigendum
See above.

Oct 2011 meeting
The quoted text was added to C++11 after WG 14 voted out the FDIS in London.

The consensus was that this is probably an oversight, and should be changed along the lines that are recommended above.

Feb 2012 meeting:
The propose resolution needs some tweaking. Should read “shall occur” rather that “shall appear to occur.”

ACTION: Clark to write up some new words for DR 405

STRAW POLL:

DR 406

Subject: Visible sequences of side effects are redundant

Summary
It has been mathematically proved that a simplification can be made to the memory model as it is specified in the final draft of the C++11 standard. Essentially, the restriction defining *visible sequence of side effects* (vsse) is redundant and can be removed with no ill effects. The main motivation for doing this is that the current restriction is misleading. 5.1.2.4p22 defines vsse's:

The visible sequence of side effects on an atomic object M, with respect to a value computation B of M, is a maximal contiguous sub-sequence of side effects in the modification order of M, where the first side effect is visible with respect to B, and for every subsequent side effect, it is not the case that B happens before it. The value of an atomic object M, as determined by evaluation B, shall be the value stored by some operation in the visible sequence of M with respect to B.

The wording of this paragraph makes it seem as if the vsse identifies the writes that an atomic read is allowed to read from, but this is not the case. There can be writes in the vsse that cannot be read due to the coherence requirements (to be included in C, 1.10p15 through 1.10p18 in C++ N3291). Consequently this is even more confusing than it at first appears.
Also propose changing 5.1.2.4p22 to the following:

The value of an atomic object M, as determined by evaluation B, shall be the value stored by some side effect A that modifies M, where B does not happen before A.

With a note to remind the reader of the coherence requirements:

NOTE: The set of side effects that a given evaluation might take its value from is also restricted by the rest of the rules described here, and in particular, by the coherence requirements below
If the committee is concerned about allowing a differing text from C++11, then a note could be added to assure the reader:
NOTE: Although the rules for multi-threaded executions differ here from those of C++11, the executions they allow are precisely the same. Visible sequences of side effects are a redundant restriction.
Suggested Technical Corrigendum
See above.

Committee discussion
Oct 2011 meeting
The changes seem reasonable, there is a concern about having C and C++ differ.

Should be contingent on Defect 402.

Feb 2012 meeting
This is more complicated than it initially appeared. Clark is not aware of any discussion in C++ on this item. We do not have a coherent proposal here.

ACTION: Clark to check with Hans Boehm to see it there was any discussion in WG21 (C++) on DR 406, and what, if any, resolution was reached.

We will not proceed with DR 406 until WG21 has taken action on it.

DR 407

Summary

C11 seems to omit the restriction imposed in C++11 in 29.3p7 (from N3291):

For atomic operations A and B on an atomic object M, if there are memory_order_seq_cst fences X and Y such that A is sequenced before X, Y is sequenced before B, and X precedes Y in S, then B occurs later than A in the modification order of M.

Furthermore, it seems that both C11 and C++11 are missing the following two derivatives of this rule:

For atomic modifications A and B of an atomic object M, if there is a memory_order_seq_cst fence X such that A is sequenced before X, and X precedes B in S, then B occurs later than A in the modification order of M.

For atomic modifications A and B of an atomic object M, if there is a memory_order_seq_cst fence Y such that Y is sequenced before B, and A precedes Y in S, then B occurs later than A in the modification order of M.

Suggested Technical Corrigendum
See above.

Committee discussion
Oct 2011 meeting
The changes are difficult to fully understand - a diagram might help. A paper for the next meeting would help the Committee make progress on this. Some concern about having C and C++ differ.

Feb 2012 meeting
No additional information has been provided. Clark believes there are two issues here, and this should likely be two DRs rather than one. Is there an existing WG21 issue here? Unknown. The proposed words apply to C++, rather than C, so we need proposed words for C. Clark believes there may be editorial issues with the proposed words for C++. There is general consensus to adopt something along the lines of the first proposed change, but we want to be in sync with what WG21 actually does.

ACTION: Clark to check into the status of DR 407 w/in WG21.

DR 408
Subject: Should locks provide intra-thread synchronization?

Summary
Most of the C++ standard, synchronisation is used exclusively inter-thread, so in particular, synchronisation can't be used to avoid undefined behavior arising from conflicting un-sequenced memory accesses, e.g.:

(x = 1)==(x = 2)

Firstly, C does not define this sort of thing as undefined behavior. Is this intentional? Secondly in C++ locks can currently be used to fix up such programs and avoid undefined behavior, e.g.:
(lock; x = 1; unlock; x)==(lock; x = 2; unlock; x)

The reason not to allow this sort of synchronisation in general is, because it disallows some single threaded compiler optimizations. Is intra-thread locking intended to be defined and usable?

Suggested Technical Corrigendum

Committee discussion
Oct 2011 meeting
The changes seem reasonable, but without actual text no position can be formed.
A paper for the next meeting is probably the best way to make progress.

Feb 2012 meeting
Blain and the submitter no longer believe this is a problem, so no change is necessary. JB prefers this remain OPEN unless we see some additional text as discussed in Oct. NAD, but leave OPEN for now. (NAD = Not A Defect). The C11 standard does define the semantics of a lock within a single thread.

DR 409 + See Items discussed in Section 6.

8. RESOLUTIONS

8.1 Review of Decisions Reached
NONE

8.2 Review of Action Items

Carry Over: NONE

New Action Items

ACTION: Convener to process an NP (NWI) through SC22 for a Technical Specification: Floating-Point Extensions for C: Part 1, Binary floating-point arithmetic.

ACTION: Convener to request N1598 be processed an “errata”. Also make it a Defect Report, DR 411, and post process as needed.
Note: In the new procedures for ISO, there is no errata listed in ways to change an IS. See 2.10.1 of the ISO/IEC Directives, Part 1.

ACTION: Clark Nelson to write up some new words for DR 405.

ACTION: Clark Nelson to check with Hans Boehm to see it there was any discussion in WG21 (C++) on DR 406, and what, if any, resolution was reached.

ACTION: Clark Nelson to check into the status of the concepts in DR 407 within WG21.

ACTION: Willem, Blain and Doug to work on further proposed words for DR 413.

9. THANKS TO HOST

The Committee expresses its great appreciation and thanks to Dinkumware for the years of hosting the WG14 Wiki since it was created.

The Committee also expresses its great appreciation and thanks to Keld Simolsen hosting the WG14 Wiki.

Thanks also to our hosts Bloomberg and Plum Hall for making the meeting arrangements and outstanding weather.

9. ADJOURNMENT

Meeting adjourned at 1:35 pm local time, Wednesday, Feb 15, 2012

INCITS PL22.11/12-00001
Reply To The Attention Of: Barry Hedquist
PL22.11 Secretary
Email: beh@peren.com

PL22.11 Meeting Minutes (Draft)
14 February 2012
Royal Kona Resort
Kona, Hawaii

Meeting convened at 4:00 PM, Feb 14, 2012, by PL22.11 Chair, David Keaton.

Attendees:

Voting Members:		
Name:	**Organization:** **P – Primary, A - Alternate**	**Comments**
John Benito	Blue Pilot - P	
Martin Sebor	Cisco - P	
David Keaton	CMU/SEI/CERT - P	PL22.11 Chair
Robert Seacord	CMU/SEI/CERT - A	
David Swvoda	CMU/SEI/CERT - A	
P. J. Plauger	Dinkumware, Ltd – P	
Tana L. Plauger	Dinkumware, Ltd – A	
Jim Thomas	HP – P	
Rajan Bhakta	IBM - P	
Clive Pygott	LDRA - P	
Douglas Walls	Oracle - P	PL22.11 IR
Sheldon Lobo	Oracle - A	
Barry Hedquist	Perennial – P	PL22.11 Secretary
Tom Plum	Plum Hall – P	
Bill Seymour	Seymour - P	
Fred Tydeman	Tydeman Consulting – P	PL22.11 Vice Chair
Douglas Gwyn	Self	Member Emeritus
Blaine Garst	Self	
Roger Scott	Coverity	
Prospective New Members		
Dave Prosser	Bloomberg	
Nevin Liber	DRW	
Nick Stoughton	IRDETO	

1. Approval of Agenda

Revisions to Agenda: None

Added Items: None

Deleted Items: None

Agenda approved by unanimous consent. (Benito/Hedquist)

2. Approval of Previous Minutes (PL22.11/11-0013)

Minutes were modified per editorial changes and approved by unanimous consent. (Hedquist/Benito)

3. Selection and Review of US Delegation.
Done at the Washington DC meeting, in Oct 2011.

4. INCITS Anti-Trust Guidelines
We viewed the slides located on the INCITS web site.
http://www.incits.org/inatrust.htm

5. 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.

6. Identification of PL22.11 Voting Members (Tydeman)
See attendance list above.
15 PL22.11 voting members participated out of 15.

6.1 PL22.11 Members Attaining Voting Rights at this Meeting
Coverity
Blain Garst

6.2 Prospective PL22.11 Members Attending Their First Meeting
Bloomberg
DRW
IRDETO

7. Members in Jeopardy

7.1 Members in jeopardy due to failure to return Letter Ballots.
None

7.2 Members in jeopardy due to failure to attend Meetings.
Intel (Attended this meeting)

7.3 Members who lost voting rights attending this meeting.
Microsoft

8. New Business

8.1 US Position on SC22/N4699 Secure Coding NP

Douglas is concerned about the state of the working draft with respect to the proposed schedule, which he believes is aggressive. We are in NP Ballot now. We cannot make changes to the document at this meeting because we are in NP Ballot. NP Ballot closes on or about March 20, 2012. We will not be pushing out a PDTS as scheduled (March 2012).

Motion: Blue Pilot / Blaine
PL22.11 recommends the U.S. position on SC 22/N 4699 Secure Coding NP be as given in PL22.11-2012-00001.

Document PL22.11-2012-00001

Suggested responses to the Seven Questions for the C Secure Coding Rules New Work Item Proposal:

1. Do you accept the proposal in the attached NWI Proposal document as a sufficient definition of the new work item?

Yes

2. Do you support the addition of the new work item to the programme of work of the joint technical committee?

Yes

3. Do you commit yourself to participate in the development of this new work item?

Yes

4. Are you able to offer a project editor who will dedicate his/her efforts to the advancement and maintenance of this project?

Yes * (comment required)
Comment: Robert Seacord of CERT has agreed to be the project editor.

5. Do you have a major contribution or a reference document ready for submittal?

Yes
Comment: The document is submitted as part of SC 22 N 4699.

6. Will you have such a contribution in ninety days?

Yes

7. Which standard development track is proposed?

Default Timeframe
------ end of recommended answers to questions ------

STRAW POLL: Any objection to unanimous consent of the motion- NONE

Roll Call Vote: (15-0-0)

Blue Pilot	YES
CMU/SEI/CERT	YES
Cisco	YES
Dinkumware	YES
Hewlett-Packard	YES
IBM	YES
Intel	YES
LDRA	YES
Oracle	YES
Perennial	YES
Plum Hall	YES
Seymour	YES
Tydeman Consulting	YES
Coverity	YES
Garst	YES

Voting YES	15
Voting NO	0
Voting ABSTAIN	0

8.2 US Position on INCITS eb-2012-00009, entry: *INCITS/TR-17-1997 [R2007], Numerical C Extension*. (5 year Review)

Keaton recommends withdrawal, since the material is overcome by events, and much of it has been added to revisions of the C Standard.

Jim Thomas, Hewlett-Packard, points out there is a portion of the TR that is still used as a reference, so it still has some validity.

We discussed Stabilizing the TR. Benito noted that WG14 adopted much of this TR in the C Standard ISO/IEC 9899:1999, changing a number of items in that adoption. Much of the TR no longer maps to what the current Standard specifies. Thus the content of the TR differs from the existing C Standard by quite a lot. We would be misleading the community if we stabilized the TR, rather than withdrawing it.

STRAW POLL: To Withdraw INCITS/TR-17-1997 (R2007), Numerical C Extensions. (12-1-2)

MOTION: Oracle/CMU
Move that we recommend to the INCITS Executive Board that the INCITS/TR-17-1997[R2007] Numerical C Extensions Technical Report be withdrawn.

STRAW POLL: 12-1-2

ROLL CALL VOTE: (12-1-2)

Blue Pilot	YES
CMU/SEI/CERT	YES
Cisco	ABSTAIN
Dinkumware	YES

Hewlett-Packard	NO
IBM	YES
Intel	YES
LDRA	ABSTAIN
Oracle	YES
Perennial	YES
Plum Hall	YES
Seymour	YES
Tydeman Consulting	YES
Coverity	YES
Garst	YES

Voting YES	12
Voting NO	1
Voting ABSTAIN	2

8.3 **Next Meeting:** Portland, Oregon, 22- 26 October, 2012. Intel is the host. The meeting venue is:

DoubleTree by Hilton Hotel Portland
1000 NE Multnomah St.
Portland OR 97232
www.portlandlloydcenter.doubletree.com

Please do not make reservations by contacting the hotel directly. Instead make your reservations at: http://www.seeuthere.com/CStandardCommitteeMeetings.
The room rate will be USD 134.00 per night; this will include no meals. To obtain that rate, **reservations must be made through the above web site:**

9. **Adjournment**

There being no further business, the meeting was adjourned at 5:00 PM local, Feb 14, 2012. (Benito/Hedquist) - Unanimous Consent.