DRAFT MINUTES FOR 8 - 11 October 2007
MEETING OF ISO/JTC1/SC22/WG14 AND INCITS J11
WG14/N1270

Meeting Dates & Times

08 October 2007 09:00-12:00 13:30-17:00
09 October 2007 09:00-12:00 13:30-17:00
10 October 2007 09:00-12:00 13:30-17:00
11 October 2007 09:00-12:00 13:30-17:00

Meeting Location:

Royal Kona Resort
75-5852 Alii Drive
Kailua-Kona, Hawaii 96740
T +1-808-329-3111
F +1808-329-9532

Host Contact information:

Thomas Plum
Plum Hall Inc.
Waihona Box 44610
Kamuela HI 96743 USA
T +1-808-882-1255
F +1-808-882-1556

1. Opening Activities

WG14 Convenor, John Benito, called the meeting to order at 9:00 AM, 8 October 2007.

1.1 Opening Comments (Plum, Benito)

Tom Plum welcomed everyone to Kona, and described the facilities, local restaurants.

1.2 Introduction of Participants/Roll Call

John Benito	Blue Pilot	USA	WG14 Convenor
Randy Meyers	Silverhill Systems	USA	J11 Chair
Douglas Walls	Sun Microsystems	USA	US HOD

Fred Tydeman	Tydeman Consulting	USA	J11 Vice Chair
Barry Hedquist	Perennial, Inc.	USA	Secretary
Jeff Muller	Oracle	USA	
P. J. Plauger	Dinkumware, Ltd	USA	
Tana L. Plauger	Dinkumware, Ltd	USA	
Christopher Walker	Dinkumware, Ltd	USA	
Keith Derrick	Plantronics	USA	
Nick Stoughton	Usenix	USA	
John Parks	Intel	USA	
Clark Nelson	Intel	USA	
Bill Seymour	self	USA	
Arjun Bijanki	Microsoft	USA	
Tom Plum	Plum Hall	USA	
Rich Peterson	Hewlett Packard	USA	
David Keaton	self	USA	
Edison Kwok	IBM	USA/CAN	CAN HOD
Francis Glassborow	self/Plum Hall	USA/UK	UK HOD

1.3 Procedures for this Meeting (Benito)

The Chair, Benito, announced the procedures are as per normal. Everyone is encouraged to participate in straw polls. INCITS J11 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.

1.4 Approval of Previous Minutes (N1231) (Hedquist)

Comments: jb sent out some changes, already incorporated – N1267 Final Minutes

Minutes approved as modified.

1.5 Review of Action Items and Resolutions (Hedquist)

ACTION: Ulrich Drepper and a small team to write a liaison report on threads to WG21, in response to Lawrence Crowl's presentation, N1196. JB, Nick, Tom Plum, and Bill Seymour to work on this with Ulrich.
DONE N1257

ACTION: Convenor to change the C99 Rationale as proposed in N1189.
OPEN

ACTION: Convenor to forward DTR 24732, as revised by the Edison Kwok editorial group, to SC22 for DTR ballot.
DONE

ACTION Bill Plauger and Rich Peterson to propose new words for N1216, unsuffixed floating point numbers and TTDT. Editorial Group is PJ, Fred, Jim Thomas, Janis Johnson, and Rich Peterson.
DONE

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.
OPEN

ACTION PJ to produce a rationale for N1182 before the FPDTR ballot.
DONE

ACTION Chris Walker to investigate sensible constraints for the beta function, and make recommendation for disposal of comment 9 in N1185.
DONE

ACTION: Bill, Fred, & JB to review changes made to N1182 and the rationale when the updated document is available.
DONE

ACTION: Convener to forward the updated Special Math TR for FPDTR ballot when ready.
NA

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

ACTION Tom, Nick, Mark, David to review N1209 (CERT Secure Programming Guidelines).

DONE – N1255

ACTION Joseph Myers to submit his words for DR314 as a proposal for C1x in the post-London mailing.
DONE

ACTION Joseph Myers to provide updated TC for DR 340.
DONE

ACTION: Randy Meyers and Joseph Myers to propose new words for a TC for DR 341.
DONE

ACTION Convenor to add change to fwprintf to match the change for fprintf %g made in TC2 to TC3.
DONE

ACTION Fred, Derek, David to review TC3 draft when available.
DONE

ACTION Convenor to forward TC3 to SC22 secretariat for ballot when ready.
DONE

ACTION Convenor to write rationale for changing the SC 22 programme of work to make TR 24747 special math to IS. Bill to review.
DONE

ACTION Nick to prepare draft words for an additional “asprintf” function for TR24731-2.
DONE

ACTION Tom to submit a paper on critical undefined behavior.
OPEN

ACTION John Benito to rewrite the C9x charter as the C1x charter for the post London mailing. All to review.
DONE

ACTION Arjun and JB to prepare a paper based on MSVC extensions that might be appropriate for standardizing.
DONE N1264

ACTION PJ to produce a paper on EDG extensions that might be appropriate for standardizing.
DONE

1.6 Approval of Agenda (WG14/N1251)

Revisions: minor items only.

Agenda approved as modified.

1.7 Information on Future Meetings

April 14-18, 2008 Delft, Netherlands. NEN as host along with ACE.

Sep 8-12, 2008 Cisco as host, likely to be in Santa Clara (needs to be in a Cisco Building).

The April meeting will not be synchronized or co-located with the C++ committee. C++ is planning to meet Sep 14-19, 2008 in Silicon Valley. We might meet the week prior to that. During a revision of both C and C++, it is particularly important to keep the two groups closely aligned.

We have no invitations for hosting after 2008.

Discussion: Should we move to a three times a year schedule for the revision? Teleconferences have proved very effective for the Austin Group. We can spend several hours discussing the width of a single bit ... will one hour teleconferences work? Use a single wiki rather than one per meeting. Discussion deferred to Thursday.

1.7.1 Future Agenda Items

none

1.7.2 Future Mailings

Post meeting, 9 Nov, 2007
Pre Delft, 14 March, 2008

1.8 Identification of National Bodies (Benito)

Three National Bodies attending are US, UK, Canada

1.9 Identification of J11 Voting Members

See J11 minutes following the WG14 minutes.

2. Reports on Liaison Activities

2.1 J11/WG14 (N1245) (Meyers, Benito)

J11: INCITS raised their participation fee to $1200.00

WG14: JB got agreement to republish TR18037 w/o going through a reballot.

TR24732 is now a Preliminary Work Item, gives us 18 months, rather than a year. If no progress then, we could be cancelled. This project is still depending on IEEE 754R, which is going through a reballot at IEEE.

Nick named editor for TR24731-2 at SC22 Plenary.

2.2 J16/WG21 (N1266) (Seymour), N1269 (Plauger), (Hedquist)

The C++ Committee met last week, here in Kona, for six days, Mon – Sat. Future meetings are planned for three meetings per year, six days each session. Considerable progress was made in defining the content of the revision to the C++ Standard.

WG21 adopted two resolutions with respect to content:

1. Constrain the scope of memory management extensions as follows:

- include making some uses of disguised pointers undefined, and providing a small set of functions to exempt specific objects from this restriction and to designate pointer-free regions of memory (where these functions would habe trivial implementations in a non-conforming implementation).

- exclude explicit syntax of functions for garbage collection or related features such as finalization.

2. Constrained the scope of concurrency extensions as follows:

- include a memory model, atomic operations, threads, locks, condition variables, and asynchronous future values,

- exclude thread pools, task launching, and reader-writer locks.

Bill Plauger explained the rationale behind these resolutions. One being to nail down what will be included, and won't be, in the revision. They represent a compromise of what is realistic, and what isn't in the time frame desired – to have an FCD ready in a year. The scope of what's adopted here makes it easier for us to address some of these items for consideration in a C revision.

Bill Seymour walked us through N1266 for containing details of WG21 activities. Lawrence Crowl has asked that we give some of these items consideration for inclusion in the C revision. Discussion of quick_exit, and threads.

ACTION PJ to write paper on guick_exit

ACTION PJ to write paper on threading library.

ACTION Clark Nelson will advance a paper on the memory model.

Tom pointed out that the use of atomics is generally confined to serious guru's, and don't generally have widespread use. This is not simple stuff. Francis asked about interest with constrained overloads. There is 'some', but not a great deal.

Mike Wong discussed WG21 support for attributes (See Agenda Item 10.13, N1262). He presented a slide presentation that described the overall concept of attributes from a C++ perspective. In general, they allow a flexible extensibility of the language without dramatic changes to the language grammar, or the addition of new keywords. Attributes can create some issues with the readability of source code, and depending on the exact syntax chosen, can create conflicts with existing grammar that uses similar syntax. Example: the use of [] as an attribute indicator conflicting with [] used in dimensions. The choices made for C++ are semantically a bit different different than that used today in practice.

JB moved some of the NWIs for WG21 into PNWI. Modules, 24737, TR1, DFP, 24733. Sally (ANSI) does not want us to submit NWIs unless we are really going to work on it NOW.

2.3 Linux Foundation (N1268) (Stoughton)

FSG has merged with OSDL to become the Linux Foundation. LSB TC1 is in preparation.

Nick presented N1268. No questions.

2.4 WG11 (Wakker)

No report.

2.5 OWG: Vulnerabilities (Benito)

OWGV met last week here in Kona. SC22 renewed the OWGV for another year and the PDTR 24772 document was registered at the Plenary.

Initial work in this area has been focused on security, however this last meeting focused on Safety (aka high integrity, etc.) Bjarne discussed the Joint Strike Fighter (JFS)

mission critical items, and there was discussion of MISRA C. There is some discussion about doing a MISRA III (MISRA II was recently published).

Francis pointed out that there is no representation on this committee of computer languages as a whole, such as dynamic programming languages.

JB – OWGV is on track to publish a Type 3 TR in 2009.

2.6 Other Liaison Activities

2.6.1 SC22 (Benito)

Rex Jaeschke will likely become SC22 Chair.

2.6.2 POSIX/Austin Group N1257 (Stoughton)

Nick presented a POSIX/C Liaison report addressing three issues:

1. Null Pointer Issues
POSIX desires that the value 0 be converted to a pointer w/r/t the difference between the definition of 'null pointer' between POSIX and C. POSIX is talking about a number, C is talking about a constant. POSIX wants NULL to be 'void * 0'. PJ says this was brought up in 1985, and there was strong push back them against doing so. i.e. NULL can also be defined as '0'. This discussion is also related to ern3.txt, an Austin Group defect report.

ACTION – Tom Plum to work with Nick to respond to the NULL pointer issue.

2. Thread APIs
Austin Group recommends not adding any APIs for threads, but provide support for multi-threaded programs. Return to discussion on this in Sec 10.

3. Problem with errno.h, and proposes new wording for the C revision. Is it, or isn't it a macro. C++ is requiring that they be macros. It would not be a bad thing if we did likewise. Make it required that errno is a macro, rather than unspecified.

ACTION – Nick to submit an proposal on requiring errno be implemented as a macro.

3. Report of Rationale Editor (Benito)

No Report

4. Report of Project Editor (Jones)

Larry Jones is likely going to be the PE for the revision, but we would like to have a back-up PE. Must know groff, uses the mm macro set, and localized extensions. the back-up is not necessarily a person expected to replace the PE if the PE is unable to carry on. The back-up PE can also expect to get overflow work from the PE.

PJ pointed out that Pete Becker took the existing roff standard for C++ to Latek.

5. Status of Bounds Checking TR 24731-1 (Meyers)

Published, ISO store cost is about 600 E.

6. Status of Decimal floating-point TR 24732 (Kwok)

Now a PNWI. IEEE is in the process of reforming the Ballot Group.

7. Status of Special Math functions TR 24747 (Plauger)

Will become an IS rather than a TR.

8. Status of TR 24731-2 (Stoughton)

Paper in mailing for discussion with Editor's report.

9. Status of TC 3 (Benito)

TC3 has passed Ballot, awaiting publication. N1235 balloted; N1256 is final with editorial changes.

10. Document Review

10.1 N1250, C1X Charter

Three additional principles have been added for C1X:

1) A prior principle, Trust the Programmer, should be regarded as outdated with respect to security and safety programming communities. Programmers need the ability to check their work.

2) No Invention. Only features that represent existing practice, and have gained commercial acceptance in their implementation should be standardized. Further, their specification should be compatible with commercial implementations.

3) Migration of the existing code base must be taken into consideration. We don't want to break existing code.

The proposed milestones for the revision process are:

CD Registration – Dec 2009
FCD Ballot – Dec 2010
FDIS Ballot – Dec 2011
Publication 2012

JB emphasized that the quality of the document is more important than the date, but we should also try to stick with a schedule. The schedule above needs an additional CD ballot period (FCD), but that is workable with the time frame outlined. PJ would like us to come up with a coherent scope of what we are doing, and why we are doing it, within the next two meetings. Should we consider adding the specific TRs to the revision? Possibly delete C99 items that conflict with C++. Look for Bjarne's paper on considerations for C/C++.

10.2 N1248, Draft WDTR 24731-2, Dynamic Allocation Functions. (Stoughton)

Discussion: This extension is still a work in progress. See Also Item 10.3.

Change the __STDC* macros to match those used in the EXT1 version of this TR. (TR24731-1).

Nick will make additional changes to stay compatible with POSIX work.

We decide to add 'wide' versions of the functions that had 'narrow' versions previously included. Those 'wide' versions are invention, i.e. have no prior art or implementation, however general feedback is favorable for including them. They maintain consistency with the functions include in 24732-1. Some discussion, with a general conclusion that extending these functions to cover wide character sets is one of "completeness" rather than "invention", and that doing so is worth while.

Some typos identified – to be corrected.

Discussion some memory allocations functions that were initially brought up in Mont Tremblant and were deferred. Do we want to further pursue those functions? No consensus to do so.

Do we want to 'transition' to the C++ documentation style, i.e. use of the naming convention in addition to paragraph numbers. Yes.

Nick to continue work with the aim of getting to FPDTR ballot in June 2008.

10.3 N1249, Editor's report for WDTR 24731-2 (Stoughton)

Discussion: This document covers the changes made to the document above.

10.4 N1243, PDTR 24747, Extensions to Support Mathematical Special Functions (Plauger)

This Type 2 TR is being moved to an IS. The PDTR ballot was put on hold. If we are happy with this, we can ballot as an IS. Change only the cover page. The document is missing the second macro which would be used to access the functions. i.e. something like:

__STDC_MATH_SPEC_FUNCS__

ACTION: PJ, Chris, and Randy to do a technical editorial review of PDTR 24747, Special Math, then forward to SC22 as an FCD.

10.5 N1244, Rationale for PDTR 24747 (Plauger)

Rationale for the above PDTR. No discussion of note.

10.6 N1252, A finer-grained specification of sequencing (Nelson)

Clark Nelson presented a paper on issues / problems with the specification of 'sequence points.

The essence of this proposal is that the existing definition of sequence points is too vague or insufficient to be useful to multi-threaded programs (parallel programming). A more useful model would be one that allows clear specification of the interactions between threads. An example presented in the paper is the Java memory model, which is described in terms of program actions such as loads and stores. Admittedly, understanding this topic is difficult. Clark is not proposing to change the technical content of sequence points, but to describe them more clearly, and eliminate those areas where the Standard implies undefined behavior where conforming behavior is expected.

Clark would like us to study this paper more, and generate comments, questions, etc. until we can decide whether or not we like this proposal.

STRAW POLL: Is this a correct direction for us to go for C1X? yes – 14; no – 0; abstain – 3.

10.7 N1253, Lifetime of temporaries (Nelson)

Clark Nelson presented a paper that points out an incompatibility between C++ and C99 caused by a statement in C99 that says: If an attempt is made to modify the result of a function call or to access it after the next sequence point, the behavior is undefined. (6.5.2.2;p5).

Discussion:
This is the shallow end of the sequence point pool. In C99, the Standard says the lifetime of a temporary ends on the next sequence point. Accessing that temporary after the next sequence point is undefined behavior. In C++, the lifetime of the temporary lasts until the end of the function, making it accessible. Why does C differ from C++ on this? Randy does not recall why we changed it in C99, nor does he believe that creating undefined behavior was our intent. The rule did not exist pre-C99. Suggestion is to remove the words "next sequence point", and/or make the words compatible with C++.

ACTION: Clark Nelson to propose new words regarding the lifetime of temporaries. See N1253.

10.8 N1263, Comments on N1241 (Kwok)

Edison has submitted comments on DTR 24732, Decimal Floating Point.

Discussion:
Edison comments:

1. No. This is a new feature. Could be an extension.
2. OBE – See Item 10.10.
3. Yes, they should be there – Defect
4. No, this should be well defined.
5. Agree
6. Agree
7. No mixing, should be diagnosable.

Fred comments:

1. Agree
2. No. We decided in London to NOT do this.
3. Agree, text to be added from Sec 9.3 as suggested.

ACTION: Edison to revise DTR24732 per the changes above, and have it reviewed by the DFP review group.

10.9 N1247, specification for a, A output conversion for WDTR 24732 (Peterson)

This paper proposes the specification of the conversion specifiers a and A output conversion specifiers for TR 24732 are incomplete, and suggest changes to make them so.

Discussion:
Rich Peterson presented N1247. The proposal completes the specification to cover explicit as well as implicit precision. Fred asked that an additional example be included. The phrase "current rounding direction" is ambiguous, and should more specific.

10.10 N1258, Unsuffixed floating constants (unresolved problems with TTDT), TR24732 Decimal Floating Point, (Peterson)

This paper refers to N1216, Problems with TTDT (Translation Time Data Types), discussed in London, where there was a lack of consensus to resolve the issue by removing TTDT and adding a pragma. Plauger and Peterson had an action item to come up with a soft resolution. However, before they could do that, JTC1/N8645, TR 24732, Decimal Floating Point, went out for ballot. The paper goes into detail describing the problems with TTDT, and proposes some solutions.

Discussion:
Rich Peterson presented N1258. The basic question: What is the type of an unsuffixed floating point constant in Decimal Floating Point? Three basic options are presented:

1) Eliminate TTDT: unsuffixed floating constants have type double

2) Eliminate TTDT: the type of unsuffixed constants is a translation-time option

3) Specify TTDT with "as if" rules that provide context

Each of these options have several sub-options.

These options have lots or few flowers, depending on the amount of fertilizer applied.

What happens when binary and decimal uses are mixed? Most programmers that want decimal will not want to be using binary. Set a compile flag? Use of Pragma to resolve this issue is doable, but puts considerable burden on the programmer. Pursue a "do what I mean" path? Always use suffixes for decimal ? Tom knows of no language that has actually implemented TTDT.

Removing the requirement for TTDT from the TR would require the use of a suffix for decimal. An unsuffixed constant would be defaulted as binary.

STRAW POLL: Pursue TTDT approach: yes -3, no – 10, abstain -5

ACTION: Rich and Edison to address the possibilities of standardizing a PRAGMA to deal with unsuffixed decimal.

10.11 N1261, Possible Defects in TR 24731-1 (R Meyers)

Randy presented a compiled list of potential DRs against TR 24731-1, with proposed recommendations.

Blanket approval of all changes that are only editorial, typos, etc.

3.1 rsize_t – the Standard is a bit vague about the exact contents of rsize_t. Leave as is, possibly add words to the rationale.

3.2 rsize_t number of elements of a wchar_t array. The wording is intentional. No change. Possibly add words to rationale.

4.0 Programmers lie about bounds. Programmers are able to defeat the intent of bounds checking functions by allocating a buffer well in excess of that needed. For example, allocating RSIZE_MAX as the buffer size allows the programmer to essentially turn off bounds checking. Do we want to do anything about this? NO. But, adding words to the rationale pointing out the problem, or set of problems. General consensus (no opposition) to adding words to the rationale.

10.12 N1259, C1X—Attribute syntax (J Myers)

This paper addresses an earlier paper on support for C++ attributes (N1233). See Also: 10.13 Below.

10.13 N1262, Towards support for attributes in C++ (Kwok)

This paper discusses proposed support for attributes in C++.

Some believe that "attributes" is another word for "pragma". Pragmas are not favorably regarded in C++. Do we have any comments that we want to send back to WG21 on this ?

Clark stated that the C++ proposal is very disappointing because it does not go far enough. We should also take into consideration Joseph's paper (N1259) on attribute syntax in considering this issue. Francis pointed out that there was a concern in C++ about trying to do too much with attributes, and thus reduced the scope of what would be accepted. Nick pointed out that we agreed in London to add attributes in the C

revision. There is existing practice in MSVC (declspec) and GCC (attribute). However, their documentation does not necessarily match implementation, and each has different scope of exactly what can be done. We want to avoid invention.

Implementors that have some form of attributes: Intel – uses EDG, Microsoft, Sun – gcc compatibility. Oracle can't use any one solution because they ship products on Windows, UNIX, and Linux. So, they make very little use of attributes. They would prefer to see a standard approach.

Nick believes we could argue / debate / wordsmith all day about the syntax for attributes. What we should do is settle on what we want attributes to do, then address syntax. General agreement on this approach. Clark believes the design goals of the C++ proposal are in flux, not clear, idiosyncratic.

Tom does not believe that we, WG14, should not tell WG21 what to do. Clark believes a list of design goals would be useful feedback, particularly if that list reflects what we, WG14, want to do.

N1229 includes a list of GCC goals, Arjun has a list in N1264.

A list of "Good Choices" in this paper, N1262:

> align(unsigned int)
> pure (promise that a function always returns the same value)
> probably(unsigned int) (hint for if, switch, ...)
> - if [[probably(true)]] (i == 42) { ... }
> noreturn (the function never returns)
> deprecated (functions)
> noalias (promises no other path to the object)
> unused (parameter name)
> register (if we had a time machine)
> owner (a pointer is owned and it is the owner's duty to delete it)

One of the things Clarks was looking for in Design Goals was saving the cost of adding new keywords. That was not addressed.

It would be good if we could agree on
1) a list of goals based on Nick's compiled list,
2) a list of what we do not want, and
3) a list of a 'to be considered'.

Nick compiled a list from these three papers, removing those items that are specific to only C++. Items with a ~~strikethrough~~ indicate those deleted during the discussion. The list below is broken out by where the attribute is used: FUNCTION, VARIABLE, or TYPE.

N1229 (GCC)	N1262 (C++)	N1264 (MSVC)
FUNCTION		
noreturn	noreturn	noreturn
~~returns_twice~~		
pure	pure	
warn_unused_result		
nonnull		
	deprecated	deprecated
VARIABLE		
aligned	align(unsigned int)	align(alignment)
cleanup		TRY-FINALLY
unused	unused	
	probably(unsigned int)	
	noalias	
	~~register~~	
	~~owner~~	
		thread
TYPE		
aligned		
packed		
~~transparent_union~~		
~~unused~~		

Francis stated that WG21 was only concerned with syntax. Our concern is focused on semantics.

ACTION: Nick, Jeff, JB, Arjun to work up a paper on pulling the candidate attributes together (See N1229 and N1264). DONE – See N1273 (post meeting).

With respect to a liaison statement, WG14 is very keen on preserving exist practice, and opposed to invention in this area, such as an attribute appearing in a place where declspec could go. WG21 allows attributes in places where they are not presently allowed, and has invented new places for them to go. Not allowing attributes in places where existing practice presently allows them is much more bothersome that allowing them in new places.

Further discussion on "cleanup" vs "TRY-FINALLY" will require more work before we are close to deciding which of those, if either, to adopt. TRY-FINALLY is not an attribute, per se. Nick should address both in his paper.

Nick's paper in draft form is on the Wiki as "attribute.doc". What else does this paper need? There is no discussion of "cleanup" and "TRY-FINALLY" as we requested above. Will be addressed in a separate paper.

ACTION: Nick to address 'cleanup' and TRY-FINALLY in a separate paper on attributes.

10.14 N1260, C1X, Bit-fields (J Myers)

DR#315 discusses issues concerning the type of a bit-field. For C99 the conclusion for the DR was to leave most of these matters implementation-defined, with the possibility of revisiting the matter in C1x.
The fundamental question is: In the width of a bit-field part of its type, or is the type that with which it was declared (ignoring the width, and with the possibility that the absence of signed is interpreted as unsigned)?

Discussion:
Joseph is not here. Nobody is really prepared to discuss this paper in detail, but it's likely that the UK C Panel will be able to provide constructive input in the near term. We would like this paper to get turned into a more thorough, complete, proposal. Fred believes that DR335 should be included in this work.

10.15 N1254, Saturation in C1x. (Plum)

Within the safety and security communities, there is an important need to determine that certain computed results have not been contaminated by integer overflow. Languages that compete with C have traditionally met that need using methods that are awkward for C; this category would include the throwing of exceptions (or signals), encoding ranges into the integer types, and the use of arbitrary-precision ("BigInt") libraries. (Call this category the "awkward methods".) However, in C1x we could meet this need by providing saturation semantics for overflow-handling.

Discussion:
Tom led a discussion of N1254. Today, in the C Standard, integer overflow is Undefined Behavior. There is no requirement to produce a result such as saturated "MAX". Pragmas could be used to detect these cases, such as _Pragma(STDCSAT). It could be a reasonable and efficient way to address a way to force detection of these cases. Tom believes that the compiler could easily deliver a solution that would beat any user-programmed solution. The intent is to get this paper on the table for further discussion.

10.16 N1264, MS extensions for consideration for C1X (Bijanki)

Arjian presented a paper of Microsoft extensions to the C language for consideration for C1X. These extensions include:

__declspec – a mechanism of applying an attribute to a declarator, and in general behaves as a storage class specifier.

Structured exception handling, which is also addressed in N1229 (Stoughton)

SAL (Structure Annotation Language), a language to annotate functions in a way that describes how their parameters are related to each other.

Links to Microsoft documentation that provide more details of these items are provided in N1264.

Discussion:
Item 8 __unalign, use to declare data as unaligned. This is an item we did not discuss in our attribute discussion (Agenda Item 10.13) . HP and Intel have implemented this as have several others although not in exactly the same way. This is not the same as align==1 contained in a C++ proposal. Tom believes the only liaison issue is the one of not introducing new types. Nick would like to see a paper describing how to add this.

ACTION: Arjun to write a proposal on adding __unlign as an attribute for C1X (See N1264).

Brief discussion on adding exception handling to C1X, i.e. try_except. No support was expressed for doing so.

Item 7 SAL. Two examples on the Wiki:

```
/* malloc returns a writeable block of _Size bytes (__bcount) which may be
NULL (_opt).  The return value must be checked by the caller (__checkReturn)
*/

__checkReturn __bcount_opt(_Size) void * __cdecl malloc(__in size_t _Size);



/* sprintf_s takes a pointer to an out buffer (__out) that has storage for
_DstSize bytes   (_bcount) and will be left zero-terminated on exit (_z).
It also takes a zero-terminated    (_z) printf-style format string
(__format_string). */

__checkReturn int __cdecl sprintf_s(__out_bcount_z(_DstSize) char * _DstBuf,
        __in size_t _DstSize, __in_z __format_string const char * _Format,
...);
```

These are typically used to give compiler warnings, such as exceeding the buffer size. It is only a compile-time check, not a run-time check. Arjun expects all Microsoft headers to use these in future releases. The actually syntax here is actually macros on top of attributes, and is meant only to represent the concept rather than propose syntax.

General favor for the concept. GCC does not seem to have anything like this. Are there IPR issues? Not clear. Arjun needs to check on this. SAL may be implementable in GCC and we could gain some feedback. Leave it in the macros. If we want to incorporate it, we would likely want it in attributes. We may want to discourage this as a run-time check. More discussion / thought is needed prior to deciding to add this to C1X. A lot of folks like it. A key question is one of having in implemented in GPL software. Doug: Sun has something called 'source code annotations' that is an identical concept, but less extensive in scope. EDG has an extension for a similar concept, compile time only.

ACTION: Convenor to check with EDG for more information on what they've done similar to the concept presented by SAL (See N1264).

10.17 N1246, Trig function cleanup. (Tydeman), Possible DR

This paper implies the Standard is unclear regarding the range of trig functions. Objection to the use of the word "may", replace it with "might". Reduce the comma's as well.

Change to read:

The **Returns** sections are in terms of ideal mathematical functions and mathematical *pi*. But, due to one or more of machine representations of floating-point numbers, rounding and implementation algorithms, the actual returned value might exceed the listed range.

10.18 N1256, Committee Draft: C99+tc1+tc2+tc3 (Benito)

N1256 will serve as the base document for the C1X revision. Proposals, etc., will be considered on the basis of this document.

10.19 N1255, CERT C Programming Language Secure Coding Standard (Secord)

David Keaton presenting on behalf of Robert Secord. Walked through the first several pages, and last two, or until OBE (lunch). Robert is soliciting comments for this document. Tom suggested that we also mine this document for things that might be good to include in C1X.

PRE00-A. Prefer inline functions to macros. PJ says this is a terrible rule. He would trust library implementer. The C Standard has specific rules for macro behavior, and Standard Library functions should be exempted.

Were the rules derived from real-world experience from experts versed in the C Standard or were they derived from some other sources? At least CERT has asked for feedback, unlike some others. If we have recommendations, they should be sent to David.

The links in the document take you to a document on a web site, which contains additions links to text not contained in the original document. This makes the document somewhat fragmented and difficult to follow. The links on the definition pages for some definitions do not go to a page that has the cited reference.

10.20 N1257, POSIX Report, #2. Threads (Stoughton)

POSIX / Austin Group wants to see support for multi threading, but does not want C to develop a threads API. There is a range of opinions on this Committee that go from adopting a low level specification for threads, to doing nothing at all with threads.

Dinkumware has developed an approach that allows a programmer to write threads for either POSIX pthreads, or Microsoft's threads. Windows does offer a pthreads interface, however it is not widely publicized or widely used.

Three realistic approaches:

1. Specify the underlying language features, no library.

2. Specify pthreads.

3. Specify something more portable, such as what Dinkumware has already done.

PJ prefers either #1 or #3. There is no point in simply doing #2.

There is no argument against adding the language hooks needed to write threads, such as semantics for atomics, and a memory model (#1 above). The question comes down to what library functionality, if any, to add (Options #2 or #3). Likely to have agreement that 'freestanding' implementations will not need to support this.

With respect to the hot button of thread cancellation, Dinkumware's implementation defers to POSIX.

STRAW POLL:

Q: #1 ONLY – 5 yes, 10 – no, 3 – abstain
Q: #2 ONLY – 5 yes, 7 – no, 4 – abstain
Q: #3 ONLY – 15 yes, 2 –no, 1 – abstain

ACTION: PJ to write up a proposal for Threads along the line of a thin binding (#3) that allows a portable approach containing language support (atomics, and a memory model), and library interfaces as needed.

10.21 N1265, Number of Exponent Digits with a, A format specifier, Decimal Floating Point (Peterson) (See Also 10.9)

See Also 10.9
No controversy here, add it to the TR. One suggested change to change the wording of an exception for the exponent of the specifier. David also asked that a note be added to the Rationale explaining this.

11. Defect Reports

11.1 Review/Resolve defect reports

DRs discussed in London that were moved to REVIEW

DR 315, Myers (UK). Implementation-defined bit-field types

Proposed Technical Corrigendum

Last sentence of paragraph 2 of 6.3.1.1, add the words *as restricted by the width, for a bit-field* as follows:

If an `int` can represent all values of the original type (as restricted by the width, for a bit-field), the type is converted to an `int`;

Doug Gwynn email: Agree with proposed TC. Suggest around third line of Spring 2006 discussion "inor" => "int or", and in same discussion Question 2 Note "I" => "The editor". In Fall 2006 discussion, responses would be better "{These would all | This would} be determined by the implementation-defined behavior specified in 6.7.2.1#4." The idea is that there is not a separate requirement for documentation.

Moved to CLOSE – No objection.

Note: The numbering scheme, i.e. which paragraphs to change, are based on TC2.

DR 328, Jones (Project Editor), String literals in compound literal initialization.

Proposed Technical Corrigendum

Replace 6.5.2.5 paragraph 2 and 3 to:

All the constraints for initializer lists in 6.7.8 are applicable to compound literals.

Change 6.5.2.5 paragraph 7 to:

All the semantic rules for initializers lists in 6.7.8 are applicable to compound literals.[82)]

Doug Gwynn email: Correction to proposed TC: "semantics" => "semantic".
Agree

Moved to CLOSE – No objection.

DR 330, Tydeman (US), Externally visible exceptional conditions.

Proposed Technical Corrigendum

Change 7.12.1 paragraph 1 last sentence to:

Each function shall execute as if it were a single operation without generating any of the exceptions "invalid", "divide-by-zero", or "overflow" except to reflect the result of the function.

Doug Gwynn email: Suggested TC is blank (probably was lost during editing). Please check that "underflow" was intentionally omitted from the list in the proposed TC. (It seems like it belongs in the list.)
Response: We don't change anything above the line. The omission of "underflow" was intentional.

Moved to CLOSE – No objection.

DR 331, Plum (US), permit FE_DIVBYZERO when errno says EDOM.

Proposed Committee Response

The Standard seems clear, no change is needed.

Moved to CLOSE – No objection.

DR 336, Stoughton (US), What does TMP_MAX actually indicate?

Proposed Technical Corrigendum

Change 7.19.1 para 3 from:

the maximum number of unique file names that can be generated by the tmpnam function".

to:

the minimum number of unique file names that can be generated by the tmpnam function".

Also, at 7.19.4.4 p2, change

"The function is potentially capable of generating TMP_MAX different strings, but any or all of them may already be in use by existing files and thus not be suitable return values."
to
"The function is potentially capable of generating at least TMP_MAX different strings, but any or all of them may already be in use by existing files and thus not be suitable return values. It is implementation defined if `tmpnam` can generate more than TMP_MAX different strings."

NOTE: Delete the last sentence above.
Moved to CLOSE – No objection

DR 337, Stoughton (US), stdio.h macro definition problems.

Proposed Committee Response

All of these constants already have required minimum values that are positive, non-zero. No changes are required.

Delete the empty Proposed Technical Corrigendum
Moved to CLOSE – No objection.

DR 343, Myers, (UK), Initializing qualified wchar_t arrays.

Proposed Technical Corrigendum

Change 6.7.8 paragraph 15:

"`wchar_t`"
to
"a qualified or unqualified version of `wchar_t`".

Moved to CLOSE – No objection.

DRs discussed in London that remained OPEN

DR 314, Myers, (UK), Cross-translation-unit tagged type compatibility

See Also N1237

Doug Gwynn email: DR # 314: Agree with response. Suggest as Editorial, in Summary after first code block changing "that in TU 2 and TU 3 but they" to "those in TU 2 and TU 3 but the latter".

Randy presented N1226 in London, and Joseph agreed with the paper then. Joseph went off to provide word to fix the cases he cared about, and preserve the cases Randy cared about (N1237). Randy developed an additional analysis in session since he found Joseph's paper to be a little vague. Tom: struct s means only one compatible type across the program. After considerable discussion, it seems that this DR needs more work. The Standard, as written, does not seem to specifically allow the compilation/linking of the three translation units in question, but it does not seem to disallow it either.

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.

OPEN

DR 329, Tydeman, (US), Math Functions and Directed Rounding

PJ: There is no rounding for some of these.
ACTION: Bill and Fred should work wording for a proposed TC for DR 329.
OPEN

DR 334, Tydeman, (US), Missing semantics of comparison macros

Should we delete reference to 754R? No.
Typo: "IC" should be "IEC"

Doug Gwynn email: If the Summary is correct, then either a caveat ought to be added to the example, or the example should be removed. I slightly favor requiring the arguments to a comparison macro to have the same real-floating type. The macros could still be the 60559 comparisons, but limited to that subset of the possible type combinations. (We don't *have* to provide access to every microfeature of 60559.)

ACTION: PJ to look at developing an approach / words for DR 334.
OPEN

DR 335, Tydeman (US), _Bool bit-fields

Is this really a Defect?

Doug Gwynn email: I think the answer is that the value stored is *unspecified* (which was not one of the bulleted choices offered). The answer to "is that undefined?" is "No, but it might not compare equal to the value that you thought was being stored."

David Keaton proposed the following Committee Response:

6.2.5p6 states that "The type _Bool and the unsigned integer types that correspond to the standard signed integer types are the standard unsigned integer types." In other words, _Bool is one of the unsigned integer types whether it is used in a bit-field or not. 6.3.1.2p1 explicitly defines the semantics of _Bool, which are different from other unsigned integer types.

A _Bool bit-field has the semantics of a _Bool (and not an unsigned int).

Accept David's response.
Moved to REVIEW.

DR 338, Peterson (US), C99 seems to exclude indeterminate value from being an uninitialized register.

Doug Gwynn email:

Comment on Summary: This describes a weak form of a "tagged" architecture, and we long ago agreed that such an architecture would not be able to fully exploit the utility of tags in a conforming C implementation.

Command on Rationale: "no allowance for padding bits present only in register representation": That's true, and analogous to guard bits for f.p. registers, or sign extension in widened signed integer registers, etc. I disagree strongly that "ia64 NaT values clearly exhibit the properties intended for C99 trap rep." Trap rep. was an unfortunate choice of name, having no necessary connection with trapping; it was only meant to describe any bit configuration that would not be a valid representation for the type. The cited use of memcpy was only one application for unsigned char w.r.t. its not being allowed to have a trap rep. The description of "translations being required to perform run-time initialization" is a distortion; it is always the case that specific architectural problems can require special care in the implementation to avoid the problems; we see many instances of that. (I had to generate an occasional extra instruction to access the top-of-stack upon function entry on the PDP-11, merely in order to guarantee that there would not be a page fault when a floating-point instruction happened to be the one to force a new stack segment to be allocated.) I also disagree with the claim that "it was certainly not the intent ... that all trap reps. ... be representable in memory"; trap reps. were defined *only* in terms of the object (storage) model -- registers are a hardware optimization that may or may not require some special care in the generated code.

Comment on Spring 2007: The second sentence is *true* and *important*. The page 6 change doesn't work; unsigned char could have an indeterminate value, but *cannot* be allowed to behave as if it had a trap representation just because its address isn't taken; that would break many of the intended uses of unsigned char as being able to safely access any byte value taken from any object.

My general feeling on a response is that no changes in the standard wording are needed, and that the NaT implementations may need to take special care in code generation in order to avoid potential problems caused by that architectural feature.
=== end of Gwynn email.

PJ sympathizes with Doug's view, Rich does not.

ACTION: Rich to work with Doug on an approach / wording.
OPEN

DR 339, Myers (UK), Variably modified compound literals

This DR is fixed by DR328. Constraints and semantics are the same as 6.7.8.
Moved to REVIEW

DR 340, Myers (UK), Composite types for variable-length arrays

This DR refers to the following reflector message (SC22.WG14.11145):

```
Consider the declarations:

typedef void (*T1)(int);
typedef void (*T2)();

int x;

In the context of these declarations, what is the composite type of the types
T2[x] and T1[]?  The possibilities are:

T1[x] - the composite type rules are applied recursively to T1 and T2,
yielding a VLA of the same size (x) as T2[x] whose element type is the
composite of T1 and T2 (which is T1).  This is the interpretation followed by
GCC.

T2[x] - in "if one type is a variable length array, the composite type is
that type" (6.2.7#3 first point), "that type" is taken literally, overriding
recursivity, and is distinguished from "an array of that size"
which is the wording used for an array of known constant size.  This is the
interpretation apparently followed by EDG.

Here is a complete testcase.  Under the first interpretation, the code is
invalid; under the second, it is valid.
```

```
typedef void (*T1)(int);
typedef void (*T2)();

int x, y;
void *p, *q;

void
f (void)
{
  T1 (*a)[] = p;
  T2 (*b)[x] = q;
  (y ? a : b)[0][0]();
}
```

Joseph Myers has supplied proposed wording:

Proposed wording for DR340, as per the action item from the London meeting.

6.2.7 paragraph 3, change first bullet to:
– If both types are array types, then:

- the element type of the composite type is the composite type of the two element types;
- if one type is a complete type, the composite type is a complete type;
- if one type is an array of known constant size, the composite type is an array of that size.

The proposed wording above has some problems.

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

DR 341, Myers, (UK), [*] in abstract declarators

London:
Parameter declarations are such even without an identifier. This answers the first question affirmatively.

All agreed that first example is/should be valid. Second is/should be invalid. “Attractive” wording still needed.

Action: Rich to propose new wording. DONE, posted on Wiki, see below.

Fall 2007

Above reference to N1238 is not relevant.

It appears the issue hinges entirely on the point that a type-name is not a declaration and does not declare an identifier, and because of that it has no scope. Instead of adding complex wording to avoid using the term "scope" as suggested in the DR, it seems clearer to modify the definition of

Scope such that it applies to type-name, which is described in 6.7.6 as "syntactically a declaration for a function or an object of that type that omits the identifier".

Suggested Technical Corrigendum
6.2.1, add a new paragraph at the end (following paragraph 7): "As a special case, a *type-name* (which is not a declaration of an identifier) is considered to have a scope that is the scope determined by the place within the *type-name* where the omitted identifier would appear were it not omitted." Also add a forward reference to "type names (6.7.6)".

6.7.5.2 paragraph 4, change "declarations with function prototype scope" to "declarations or type-names with function prototype scope".

Wordsmithed the above somewhat. Now reads:

Suggested Technical Corrigendum
6.2.1, add a new paragraph at the end (following paragraph 7): "As a special case, a *type-name* (which is not a declaration of an identifier) is considered to have a scope that begins just after the place within the *type-name* where the omitted identifier would appear were it not omitted." Also add a forward reference to "type names (6.7.6)".

6.7.5.2 paragraph 4, change "declarations with function prototype scope" to "declarations or type-names with function prototype scope".

Agreed with the words above.
Moved to REVIEW.

DR 342, Myers (UK), VLAs and conditional expressions

London:
There are no rules for specifying a composite type when it depends on an expression and the expression is not evaluated. The proposal makes an implicitly undefined behavior explicitly undefined.

General consensus is that implicitly undefined is satisfactory, and we should issue an RR stating this (therefore making it explicit but only in an external document).

Proposed Committee Response: The standard does not speak to this issue, and as such the behavior is undefined.

Leave OPEN.

Kona:

Doug Gwynn email: While the proposed response is true, it is not a good response. The submitter knew already that there was no spec for this; we need to say more. Try "The standard does not specify the behavior for this situation, so the behavior is undefined. The committee does not wish to encourage such code by imposing arbitrary rules to specify this behavior." (Or some other reason.)

Proposed Committee Response

The Standard does not speak to this issue, and as such the behavior is undefined.

General feeling that the proposed response is too brief. Do we want the Standard to say anything more about this? No. We prefer to leave it as UB, rather than make it either allowed, or a constraint violation.

ACTION: Randy will generate more words for the response to DR 342. DONE – on Wiki.

Moved to REVIEW

DR 344, Nelson, (US), Casts in preprocessor conditional expressions

London:
General agreement that this is non-controversial. Copy Suggested TC to Proposed TC.

Kona:

Doug Gwynn email: Agree with proposed TC, and would also suggest a TC in the footnote: add "casts," after "enumeration constants,". Casts are important enough (to understand as not applicabile in this context) to merit continued explicit mention.

Proposed TC (6.10.1p1, strike "it shall not contain a cast;") (fn reference s/be 144)

Change 6.10.1p1:

The expression that controls conditional inclusion shall be an integer constant expression except that: identifiers (including those lexically identical to keywords) are interpreted as described below;[141)] and it may contain unary operator expressions of the form

Move to REVIEW.

DR 345, Jones (UK), Where does parameter scope start?

London:
Q1: The committee believes that 6.9.1p9 is a comment on the storage duration and does not override the lexical scope described 6.2.1p7.

Proposed Committee Response:

As above.

Q2 needs Proposed TC: not ready yet.

Leave OPEN.

Kona:

Proposed Committee Response

For question 1: The Committee believes that 6.9.1 paragraph 9 is a comment on the storage duration and does not override the lexical scope described in 6.2.1 paragraph 7.

Doug Gwynn email: I suggest a TC changing the relevant sentence of 6.2.1p4 to "If so, the identifier in one declarator (the inner scope) appears after the identifier in the other declarator (the outer scope). {new footnote: The inner scope might not be a strict subset of the outer scope; consider void f(int f/*I*/)/*O*/{} where the inner scope starts at point I and the outer scope starts at point O.]

PJ believes that the wording in the Standard is basically wrong, and need to be reworked.

Leave OPEN

ACTION: PJ to generate a new proposed response to DR 345.

SUMMARY OF DR ACTIONS

[WP] – add to Working Paper
[CR] – committee response only

DR 315 – CLOSED [WP]
DR 328 – CLOSED [WP]
DR 330 – CLOSED [WP]
DR 331 – CLOSED [CR]
DR 336 – CLOSED [WP] (SEE NOTE)
DR 337 – CLOSED [CR]
DR 343 – CLOSED [WP]

DR 314 – OPEN
DR 329 – OPEN
DR 334 – OPEN
DR 335 – REVIEW
DR 338 – OPEN
DR 339 – REVIEW
DR 340 – OPEN
DR 341 – REVIEW
DR 342 – OPEN

DR 344 – REVIEW
DR 345 – OPEN

12. Other Business

12.1 Web Site (Stoughton)

Do we want to continue to use our present web site, or move to another. Doug would like to see an active member controlling our web site. Today, only our maintainer has access to the web site, and that's created problems with accuracy and timeliness in getting information out. USENIX or Dinkumware could do this. Francis believes whether or not we move, we need a shadow site. Move the C reflector as well? Possibly.

Brief discussion of Live-Link. SC22 is moving all documents to Live-Link, however access to the site right now seems limited.

12.2 Liaison Statements

12.2.1 Liaison Statement to POSIX/Austin Group (N1272) (Stoughton)

C - POSIX Liaison
==========
This paper describes the issues arising from the most recent WG 14
C meeting that affect POSIX.

Thread Proposals for C++
==========

There were at least three proposals on the table for how to proceed with threading in the C revision:

1. Just do the underlying mechanisms for multiple threads (memory model, atomic data types, thread local storage, sequence points, etc), but do not introduce any Thread Launching API. [proposal in N1257].

2. Add the pthread* interfaces from POSIX. There is a question on how much of the pthread interface is appropriate, but The Austin Group has agreed to assist in developing the words for C. [proposal in N1257]

3. Add a slightly higher abstraction level, based on the existing practice of Dinkumware, which should also be compatible with whatever C++ does, that would be a thin veneer over any underlying pthread* and windows threading API.

Each of these proposals has strong arguments in favor of it, and somewhat weaker arguments against it. There is widespread existing practice in this space (mainly option 2, but to a lesser extent option 3 (Dinkumware)).

The entire committee was in favor of adding the underlying mechanisms, and it is almost certain that the C++ memory model, atomics etc will be included in the revision of C.

However, when it came to deciding beyond that there was considerably less consensus.

Option 1: 5 in favor, 10 opposed, 3 abstain.

Option 2: 5 in favor, 7 opposed, 4 abstain.

Option 3: 15 in favor, 2 opposed, 1 abstain.

As a result, an Action Item was given to Bill Plauger to develop a paper describing option 3 for consideration at the next meeting. Neither of the other two options is specifically off the table, but the "thin-layer over the OS specific one" seems to have the most traction, providing greater portability for applications that use this layer. It was observed that the Dinkumware library was approximately 1500 lines of code + 1500 lines of header for *both* the pthread and windows implementations (which are ifdefed together).

The committee sees its job as harmonizing competing existing practice, and believes that there is a useful intersection that can be implemented with low overhead, and at this point has requested a more detailed proposal.

CERT Secure Programming Document
==========
The committee spent a short while discussing the CERT C Programming Language Secure Coding Guidelines in N1255. In the past, other groups have written documents similar to this (and often not as good as this), and published them without asking for committee input. This time at least that step has been taken. The document contains a section on POSIX, and the Austin Group has been invited to collectively review this section. If the Austin Group does nothing, it is possible that such silence will be taken as assent, so WG 14 would strongly encourage Austin Group members to spend some time looking over this.

12.2.2 Liaison Statement to WG21, C++

Thread Cancellation

WG 14 notes with satisfaction the progress made towards a thread API in WG 21 reported by its liaisons. WG 14 also notes that much of this progress was made possible by the removal of the cooperative cancellation/interruption part of the API. This aspect had previously caused concern in WG 14, and the working group wishes to thank WG 21 for this significant improvement. However, WG 14 also notes that the "Scope" motions passed by WG 21 do not explicitly remove cooperative cancellation from the scope. WG 14 would have serious liaison concerns should the concept of cooperative cancellation reappear, and urge WG 21 to explicitly remove it from the scope of its current revision for of the C++ Standard.

Discussion: PJ is all for this. Tom doesn't share Bill's pessimism. Francis pointed out there is a body of people in WG21 that don't believe they can't solve the problem of thread cancellation.

STRAW POLL – do we support this Liaison Statement?

yes – 12, no – 0, abstain - 4
Consensus to proceed with the liaison statement, review by UK (Francis), US, and Canada (Edison)

13. Administration

13.1 Future Meetings

See 1.7 above.

13.2 Resolutions

No meeting resolutions adopted.

13.2.1 Review of Decisions Reached

See Action Items, 13.2.3

13.2.3 Review of Action Items

ACTION: Convenor to change the C99 Rationale as proposed in N1189.

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.

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

ACTION: Tom to submit a paper on critical undefined behavior.

ACTION: PJ to write paper on guick_exit

ACTION: PJ to write up a proposal for Threads along the line of a thin binding that allows a portable approach containing language support (atomics, and a memory model), and library interfaces as needed.

ACTION: Clark Nelson will advance a paper on the memory model.

ACTION: Tom Plum to work with Nick to respond to the NULL pointer issue. (N1257)

ACTION: Nick to submit an proposal on requiring errno be implemented as a macro. See N1257.

ACTION: Clark Nelson to propose new words regarding the lifetime of temporaries. See N1253.

ACTION: Everyone review N1256 (C99+TC1+TC2+TC3) for completeness, proper inclusion of TC3 material, etc. Comment to JB

ACTION: PJ, Chris, and Randy to do a technical editorial review of PDTR 24747, Special Math, then forward to SC22 as an FCD.

ACTION: Arjun to write a proposal on adding __unalign as an attribute for C1X (See N1264).

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.

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

ACTION: PJ to generate a new proposed response to DR 345.

ACTION: Nick to address 'cleanup' and TRY-FINALLY in a paper.

ACTION: JB will maintain a table of open action items on the Wiki.

13.2.4 Thanks to Host

Thanks to Plum Hall for hosting the meeting, providing the internet connections, and arranging the great weather.

14. Adjournment

Adjourned at 15:05 local, Thursday, 11 October, 2007

WG14 US TAG, INCITS J11 Meeting, Wednesday, October 10, 2007, 4:00 PM

Attendees

John Benito	Blue Pilot	
Randy Meyers	Silverhill Systems	J11 Chair
Douglas Walls	Sun Microsystems	J11 IR
Fred Tydeman	Tydeman Consulting	J11 Vice Chair
Barry Hedquist	Perennial, Inc.	Secretary
Jeff Muller	Oracle	
P. J. Plauger	Dinkumware, Ltd	
Nick Stoughton	Usenix	
John Parks	Intel	
Bill Seymour	self	
Arjun Bijanki	Microsoft	
Tom Plum	Plum Hall	
Rich Peterson	Hewlett Packard	
David Keaton	self	
Edison Kwok	IBM	

1. Anti Trust

INCITS J11 members are reminded of the requirement to follow the INCITS Anti-Trust Guidelines which can be viewed at http://www.incits.org/inatrust.htm.

2. Patent Information. Randy believes that it may be worthwhile to review http://www.incits.org/call.htm

Tom believes there may be several folks working on Quality of Implementation patentable items. He and David have one.

If there are folks with patent issues please contact Randy.

3. Attendees: 14 voting members present. Total voting members is 20. We have quorum.

4. Select US delegation for the next two meetings.

Douglas Walls HOD
Nick Stoughton
Barry E. Hedquist

Motion to approve the US Delegation (Plauger) (Keaton)
Vote:
Motion is APPROVED 14, 0, 0, 20

4. 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.

5. J11 is soliciting input for what we want in the C1X revision. Convenor has polled all NBs to provide input for this.

- PJ: Market responsive, make it easier to conform, close the gap between C and C++.
- Review of Bjarne's papers on issues between C and C++
- Any show stoppers. i.e. things that MUST be in C1X ? We've not generally approached things that way. We are in general agreement with the principles laid out by the WG14 Convenor, and expect to remain that way.

6. Adjourn at 4:30 PM