# MINUTES FOR APRIL 2008 (DRAFT) MEETING OF ISO/JTC1/SC22/WG14 AND INCITS J11 WG 14/N1300 - PL22.11/08-0001

Meeting Location:

*April 14-18 th:*

NEN
Vlinderweg 6, 2623 AX
Delft, The Netherlands.
Tel.nr.: (015) 2690390 (general number)

Host:

NEN

Host Contact information:
Willem WakkerACE – Associated Computer Experts

**1. Opening activities**

## *1.1 Opening Comments (Wakker, Benito)*

Willem welcomed us to NEN. The meeting is sponsored by NEN, ACE and AT Computing. Wireless access passwords will be renewed daily. We must wear ID badges at all times in the building. There is a breakout room available on Tuesday and Wednesday if required. Lunch is available in the canteen (at cost).

The building is open 8.30am-7.30pm.

## *1.2 Introduction of Participants/Roll Call*

| | | | |
|---|---|---|---|
| Willem Wakker | ACE | Netherlands | HOD |
| John Benito | Blue Pilot | USA | WG14 Convener |
| Robert Seacord | CERT-CC | USA | |
| Mark Terrel | Cisco | USA | |
| Tana L. Plauger | Dinkumware, Ltd | USA | |
| P. J. Plauger | Dinkumware, Ltd | USA | |
| Christopher Walker | Dinkumware, Ltd | USA | |
| Cecilia Galvin | Freescale | USA | |
| Rich Peterson | Hewlett Packard | USA | |

| | | | |
|---|---|---|---|
| Edison Kwok | IBM | CANADA | HOD |
| Clark Nelson | Intel | USA | |
| John Parks | Intel | USA | |
| Arjun Bijanki | Microsoft | USA | |
| Keith Derrick | Plantronics | USA | |
| Tom Plum | Plum Hall | USA | |
| Rex Jaeschke | SC 22 | SC 22 | |
| David Keaton | Self | USA | |
| Bill Seymour | Tydeman Consulting | USA | |
| Randy Meyers | Silverhill Systems | USA | J11 Chair |
| Douglas Walls | Sun Microsystems | USA | HOD |
| Fred Tydeman | Tydeman Consulting | USA | |
| Nick Stoughton | Usenix | USA | |

## *1.3 Selection of Meeting Secretary*

Nick Stoughton agreed to act as secretary until Thursday (he will not be present on Friday).

## *1.4 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.

### 1.5 Approval of Previous Minutes (N1270) (Hedquist)

Comments from Fred Tydeman (typos only).

Minutes approved as modified.

## *1.6 Review of Action Items and Resolutions (Stoughton)*

ACTION: Convenor to change the C99 Rationale as proposed in N1189.
CLOSED
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 Randy Meyers, Robert Seacord and Nick Stoughton to write additional rationale for the issues raised in N1210.
OPEN

ACTION: Tom to submit a paper on critical undefined behavior. CLOSED
ACTION: PJ to write paper on quick_exit. OPEN.
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. CLOSED

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

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

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

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

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

ACTION: PJ, Chris, and Randy to do a technical editorial review of PDTR 24747, Special Math, then forward to SC22 as an FCD.
CLOSED.
ACTION: Arjun to write a proposal on adding __unalign as an attribute for C1X (See N1264).
CLOSED

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.

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

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

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

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

## 1.7 Approval of Agenda (N1291)

Additional liaison reports added to agenda. Added N1271 from post-Kona. Added N1252 from pre-Kona. Add TR18037 Status report. Revised agenda approved.

### *1.8 Information on Next Meeting. (Terrel)*

Santa Clara in September, preceding the C++ meeting. On the Cisco campus. Invite expected to go out this week. September 8-12 (Monday-Friday).

There are no currently planned meetings for 2009. Where and when should we meet? How to coordinate with WG 21? Should we try some sort of virtual meeting ... teleconference / wiki? General feeling in favor of 2 physical meetings with regular teleconferences, wiki etc. Cisco prepared to host teleconference.

JB will start working to get a host for April 2009. There will be teleconferences before then.

Post Delft mailing: 2008-05-16

Pre Santa Clara: 2008-08-11

### *1.9 Identification of National Bodies (Benito)*

US, Canada, Netherlands.

### *1.10 Identification of J11 voting members (Tydeman)*

15 voting members present.

## 2. Liaison Activities

### *2.1 WG14 / J11 (Benito, Walls, Meyers)*

FCD 24747 Special Math is in ballot. Aiming for publication early next year. JB three year term is up this year. He is prepared to stand again if nobody else steps up.

This is the last meeting of J11 ... on 4/28 it will become PL22.11. The INCITS EB decided to do away with CT22 and replace it with a structure unlike any other. J11 is no longer to be a Technical Committee (TC), it becomes a Task Group (TG). All officers remain the same. TAG remains with PL22.11.

### *2.2 WG21/J16*

#### 2.2.1 Deprecate "auto"

General agreement to deprecate the "auto" keyword. Should we ask WG21 to go back to the previous use of "register" (no adress). No, this will not fly with WG 21.

#### 2.2.2 Extended Identifiers and TR 10176

There are two Cyrillic characters that are incorrectly classified. This is not a problem for C; the list is a "minimum" list, not an absolute.

#### 2.2.3 Static Assert and General Criteria for C++ Compatibility

We should not consider liaison for things out of scope such as templates, are overly complex or inefficient, or are simply experimental and untested. Those are things that are exclusively C++.

- In C scope (e.g. not templates)
- zero to low performance impact
- definition is fully baked (in shipping implementations)
- low complexity

The static assert is a new feature in C++, and has been subject of a discussion of the WG 14 reflector. It fits the above criteria. ACTION Robert Seacord to write a paper on static assert.

#### 2.2.4 Recategorizing some Undefined Behaviors

C++ has moved some things from undefined behavior to what is effectively a constraint violation.

Every single compiler has extensions. Undefined behavior allows for such extensions. Not all undefined behavior is a problem.

We should review all uses of "undefined" to see if there is benefit to moving some of these to constraint violations.

ACTION: Tom to provide list of undefined behavior that should become contraint violations to match C++ usage.

#### 2.2.5 Timed_mutex

There exists a problem with separating the types of mutex and timed_mutex in C++. This is still a major problem in WG 21, and POSIX is strongly opposed to there being two distinct types.

Context: C++ is adding a threading API. They have two distinct types for mutex. POSIX has one. WG 14 is considering a paper with threading APIs, and currently has timed_mutex.

Should defer discussion of this until we discuss the C threads paper. We should make any decision based on the technical merits only, not the politics.

### *2.3 Linux Foundation (Stoughton)*

Nothing to report.

### *2.4 WG11 (Wakker)*

Have not met since the last WG14 meeting. Language Independent Data Types is now published, and is freely available. LIA is also under development.

### *2.5 OWG:Vulnerability (Plum)*

Meeting last week in Amsterdam. Document progressing well. Schedule for PDTR by end of this year. There are some "Implications for Standardization" that we should consider.

### *2.6 Other Liaisons*

None

## 3. Report of Rationale Editor

There have been a few changes, but the document has not been republished recently. Will probably wait for a few more changes.

## 4. Report of Project Editor (Jones)

The current project editor is not able to attend meetings, is not a member of J11. Nick Stoughton has indicated that (with assistance from David Keaton) he is prepared to act as PE.

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

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

## 5. Bounds Checking TR

Nothing to report

## 6. Decimal Floating Point

Still on hold for IEEE 754r. Edison reported that Mike Cowlishaw says that they are progressing with ballot resolution.

## 7. Special Math IS

FCD is still in progress. Fred has found a typo; let the editor know.

## 8. TR24731-2

Slow progress. PDTR ballot should occur in next six months.

ACTION Nick to get revision TR24731-2 into pre-meeting mailing.

## TR18037 Embedded C

Willem: publication is expected shortly (according to ISO website it is available). Although we have provided justification for free availability it has been rejected, and will be available for fee only.

## 9 Document Review

### *9.1 N1278 Distinguishing Criticality of Undefined Behavior*

This paper divides undefined behavior into two groups"critical UB, and "ordinary" UB. Often the critical undef behavior leads to security exploits. On the other hand, things like shifting by a value too large does not in general lead to a program crash or other unbounded side effect.

The paper defines critical undefined behavior as "Define a *critical* undefined behavior as one which causes either an unbounded side-effect or an improper control-flow."

Lengthy discussion on the merits of undefined for allowing free-form implementation defined behavior. All undefined behavior is an opportunity for vulnerability. Vulnerabilities come from writing out of bounds (and occasionally reading out of bounds).

General sentiment is for Tom to develop next version of the paper.

## 9.2 N 1282 **Clarification of Expressions**

This proposal specifies a wording addition to the C standard to clarify existing intent regarding the ordering of subexpressions within a full expression. DR087 was the initial motivation.
This overlaps (slightly) with the new sequence point stuff that will be discussed in N1252.
Straw Poll: Should we change the C working draft along the lines proposed in N1282:
16 in favor. Opposed: 1. Abstain: 2.
ACTION Nick to add the N1282 words to the Working paper.

### *N1252 A finer-grained specification of sequencing*

This was reviewed in Kona. However, particularly in the light on N1282, we looked at it again. There was no decision made in Kona with this paper. C++ no longer uses the term "sequence point", but still retains the same concepts as described in this paper.

Should we also remove the term "sequence point"? That sounds like a lot of work. Couldn't we copy the C++ wording?

C++ uses the "sequenced before" relationship, rather than "sequence point".

David Keaton noted that it is difficult to prove equivalence between C and C++. Our current words are not well understood, and we should try to improve them. However, it is also critical we don't break optimizers. Need to prove rigorously that we haven't changed anything we didn't mean to!

Bill Seymour noted that few people outside this committee know the term, or even less what it means. How can you prove the semantics haven't changed if we change the words.

Randy pointed out that it is only the real experts who will care.

PJ pointed out that Tom Plum invented the term many years ago. It is a good and important concept. The important thing happening now is multi-threading, and multi-core processors. Sequence points become the optimizer and the sequencer and synchronization mechanism.

Tom responded: concurrency has led to new language. Old language was around discrete points in time. New language is about ordering only. At some point, the program is converted into a linear sequence of operations.

If Annex C could be rewritten in terms of the sequenced before relation, it would go a long way toward showing the equivalence of the two ideas.

PJ proposed "going down the garden path": implement N1252 as written, then fix it later.

Straw Poll: Should we implement N1252 as written, without prejudicing any future discussion on changing to C++ terminology?

In favor: 17

Against: 0

Abstain: 2

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

ACTION: Nick to add the words from N1252 to the WP.

9.3 lifetime of temporaries (WG14/N1285) (Nelson)

Paper provides two alternate approaches. General feeling in favor of paper.

Straw poll: Factored approach?

For: 15

Against: 1

Abstain: 3

Distributed Approach

For: 1

Against: 1

Abstain: lots

ACTION: Nick to add factored approach from N1285 to the WP

9.4 C++ parallel memory model (WG14/N1284) (Nelson)

Clark presented N1284. David K commented on the definition of "memory location", which might invalidate existing ABIs. In the case of a structure containing a char, a 16 bit bit field, and another char, it is proposed to be invalid to access this as a single 32 bit value (e.g. a load-modify-write). So a new compiler must not lay out this struct in a single 32-bit value if it wants to conform ... but old ABI allowed it to be one 32-bit val.

In the world of parallel programming, this is the way of life.

Rich asked about the use of volatile in this case ... basic rule being anywhere one thread can "see" a variable that another thread might be able to modify, that variable should be volatile.

This means most things need to be marked as volatile!

The volatile type qualifier has a long history within C and C++, and changing its meaning is both risky and unnecessary. In addition, the existing meaning of volatile, "may be modified by external agents", is somewhat orthogonal to "may be modified concurrently by the program".

There is some concern about any ABI breakage. Where there is a data structure that would need to change, it already has undefined behavior in a multi-threaded environment. But you can't link an old library with a new app.

Can we come up with terminology that does not break ABI?

The definition is the way it is so that we can understand and convey the concept of data race. Combining loads (or stores) of multiple locations (e.g. loading all 32 bits of the structure above) is perfectly allowable in the absence of any synchronization issues (e.g. use of atomic data types or mutex calls). Only important if multiple threads are using this struct to synchronize.

There needs to be more work on the paper anyway, especially the library portions.

Put discussion on hold for overnight consideration.

*Tuesday morning update:* The most important feature of the memory model is the concept of "sequenced before", a specialization of "happens before". Atomic data types allow for this ... store-release in one thread sequenced before a load-acquire in another thread. Acquisition clears the read cache.

Non-terminating loops break common compiler optimizations. New language forbids some corner case infinite loops, allowing an optimizer to move code from after the loop to be executed before the loop if it feels it appropriate. A "while(1);" loop can be assumed to terminate. PJ suggested a possible new attribute for such loops to stop the optimizer trying such tricks for real, intentional, infinite loops. Very similar to "noreturn".

Mark suggested that this was surprising for programmers, and we seems to be making rules for the compiler writers rather than the users. Embedded applications often use infinite loops intentionally.

Prototypical example is

```
int i = 0;

while(i != 2)

      ;

x = i;
```

Optimizer can remove the loop and set x to 2 anywhere.

Tom pointed out that the entire proposal has been very carefully considered by the C++ WG.

Return to discussion of memory location definition. Only important in the consideration of data races. An atomic char need not be the same size as a plain char.

Rich pointed out that the alpha compilers used an attribute or compiler option to select width of access and alignment.

Process question...should we set up a subgroup to discuss this memory model? Tom thinks we do need such a group. Issues to be dealt with are definition of memory location for word-oriented machine architectures (e.g. super computers, some embedded processors, DSPs); and to explore idea of attribute on non-terminating loop.

Subgroup is Clark, Nick, Tom, PJ, David Keaton, Edison, Randy, Rich.

Clark will also coordinate this with WG21.

One other issue came up: atomic types use type-generic macros. This requires some compiler magic. Some link to Fred's paper N1271. PJ believes we will need to spend some time here. Some sort of function overloading may be needed. There is differing existing practice here, but mostly this is going to be invention.

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

9.5 Attributes Commonly Found in Open Source Applications (WG14/N1279) (Walls)

This paper reviews a number of additional commonly used attributes.

Keith volunteered to take the lead for the attributes section. Each attribute should have its own paper.

Should take a pass through this list giving a yes or no to each.

Straw polls:

alias: 2-4-9

weak: 5-7-5

always_inline: 2-10-5

noinline: 10-3-5

constructor: 1-6-11. Tom pointed out that Microsoft have developed a technique for hiding function pointers that might influence peoples decision. ACTION Tom and Arjun to write a paper on hiding function pointers in regard to N1279 constructor/destructor attributes.

destructor: 1-10-8

const: 3-5-10

malloc: 5-0-10

visibility: 1-4-10

9.6 Attribute Names, Use Existing Practice (WG14/N1280) (Walls)

This paper discusses using the original gcc/MSVC attribute names, rather than moving into a reserved namespace. Feeling is unclear here. Douglas will bring this back up each time we bang on the attribute nail harder!

9.7 Adding ***_Unaligned*** to C1x (WG14/N1283) (Bijanki)

Proposal adds new keyword as type qualifier. General agreement such a concept is useful. Should it be a keyword or an attribute? There is already a proposal for align as an attribute. Main use would probably be in casts of pointers. Clark: Whether keyword or attribute is purely syntactic. Semantically, it effects the type system. C++ hasn't made up its mind yet on this. Adding stuff to the type system is a big step (not as big for C as C++, but still significant). Need a good motivation for such a step. Rich: really a type qualifier.

There is more than one implementation (Microsoft and Tru64).

Needs to be in the type system. It is a property of the operand not hte operator. Keith asked if there is any potential library impact (on existing library funcs). Randy thinks not ... everything is already unaligned, except possibly wide-char stuff, which is typically not an issue.

PJ points out that this is a significant price for some compiler writers. Some people already have it, but many do not. There is also an “outer-product” issue ... combinations of unaligned types.

Tom agrees that this needs to be a new type qualifier. No new APIs needed.

Rich argues with PJ: may not involve in code generator. He believes that this is not guaranteed to actually load an unaligned object, just not to throw a fault if you try.

Willem argues that you probably only need this on a few types ... particular integer types, but not others.

Randy noted that several compilers have done it, and have managed somehow to do it completely and correctly. The feature is over ten years old, and did not have a huge impact on the compilers that implemented it. The feature was designed with the type system in mind. Especially important in packed structures and parsing network packets.

Tom is sympathetic to PJ's point on code generation. This is a platform dependent problem. Agrees that this really impacts the packed and aligned attribute handling. They come as a package.

Nick pointed out that this overlaps with the memory location discussion for the memory model.

Arjun agrees that this is part of the entire package with packed and aligned attributes.

Mark thanks Randy for his points.

Clark notes that there is a package, and the alignof operator should probably be a part of this package. He also notes that C++ has alignment and alignof, but not _Unaligned.

Robert asked if there is any I/O impact. Randy says that despite its many problems, programmers try to fread() a whole structure and assume that it is laid out correctly.

There seems to be a consensus that this is a wanted thing, but should be considered along with other alignment pieces (align, alignof, packed, etc). Arjun should recruit assistance to write the rest of the package.

9.8 Attribute Syntax (WG14/N1297) (Stoughton)

C++ are also developing attributes. Their work is as yet incomplete, but it is anticipated that they will complete it this June. Therefore, we will go from having two individual compiler specific attributes (i.e. __attribute__((...)) and __declspec(...)) to three (adding [[...]]).

By the time C1x is published, the C++ standard will be implemented.

Tom notes that _Pragma really does go a long way ... probably all the way, and spelling it [[...]] would be acceptablt.

Clark notes that _Pragma is not integrated into the phrase structure of the language, so _Pragma does not go all the way.

Using macros to hide the spelling is important. If we equate _Pragma with attributes, do we restrict the placement if all pragmas? Rich notes that the placement of attributes depends on what the attribute is. Similar for pragmas. So it isn't an issue.

Bill Seymour notes that [[]] notation is fine, and placement is important.

Because _Pragma is allowed anywhere, it allows unspecified implementation extensions. Attributes effect declarations. Because pragmas are part of the preprocessor chapter, they are not suitable for C++.

Each attribute has different placement requirements. Keith will develop a template for describing attributes. ACTION Keith to develop attribute placement requirements as a part of a template for attributes.

Clark wants to be able to place an attribute wherever you can put a decl specifier. This is what WG21 finally agreed to (it was the one place you couldn't put them in the original paper).

There are several categories of attributes, and their placement will vary depending on which class.

There is progress to be made on attributes, independent of C++.

The “existing practice” criteria do not meet C's idea of it!

While we are editing documents, we should use [[xx]] for its economy of typing only, without prejudicing the eventual decision.

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

What do we want, where does it go, and how do we specify it; all unanswered questions. Not our job to design on the fly. Base on existing practice.

Clark: there is a difference of approach in attributes between the two committees. C++ is trying to invent a new pragma syntax. This is a place they expect implementers will provide extensions. We are looking much more at what we want to standardize (the attributes themselves) and using that to drive our decisions. We already have _Pragma for other extensions, and vendors already have extended __attribute__ anyway.

We have a list of the attributes we are currently interested in. Focus on these.

9.9 Threads proposal (WG14/N1287) (Plauger)

Bill presented his paper on the Dinkumware threads library.

It is a very thin wrapper over the POSIX pthread interface.

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

Some time spent discussing the relationship between the xtime structure in this paper and the timespec structure in POSIX.

ACTION PJ to present actual edits for threads for the next meeting.

9.10 Cleanup and try finally for 'C' (WG14/N1298) (Stoughton)

Both these constructs have value, and recommendation is to proceed with both. PJ believes we should also add the constructor and destructor attributes to this pair.

Nick does not have the bandwidth to proceed here, and would like to hand this off to someone else!

Nick: The constructor/destructor attributes require linker/loader and other runtime support and do not belong in a language standard.

General support for try-finally. Should there be exception handling too? try-except-finally? Nick said this was never something he planned.

Structured exception handling is widely used and understood, and a part of the MSVC compiler. It is also imperfect, and has known flaws. Try-finally is a well understood metaphor. Exception adds problems. C++ compatibility make it worse.

Adding exception handling needs words about how an exception is thrown. If exceptions were limited to purely synchronous throws, it might be palatable.

Bill Seymour agrees we should talk about exceptions, but it is a separate topic.  Tom answered Robert's desire to see exceptions in C ... from certain points of view they close security vulnerabilities (people not testing return values).

Nick points out that adding exceptions will break the existing library, and will need a whole new parallel library that is exception based. Still will be problems with old code.

PJ feels try-finally is a better solution than cleanup, and Nick agrees he's never been in the position where he has had a choice.

Straw polls:

Who wants to see a fully-formed proposal for try-finally:

For: 11

Against: 0

Abstain: 7

Similar for cleanup attribute:

For: 2

Against: 9

Abstain: 7

Add some kind of exception handling for C:

For: 2

Against: 9

Abstain: 8

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

9.11 On Support for TR-19769 and New Character Types (WG14/N1286) (Stoughton)

Nick and Rich presented the paper. PJ explained the history of why the TR happened in the first place. Nick concerned about the explosion of APIs. HP (Larry Dwyer) concerned about ABI breakage issues.

Unicode still adding characters, though rate slowing. 16-bit chars not really enough any more...over 100,000 chars at this point. Tom pointed out that the ability to use Unicode strings was an important part of this TR. The TR does not really call for library support. C++ has bought the concept lock-stock-and-barrel. How to prevent the library API explosion? Move to UTF-8. Sorting and collating is easier with UTF-8, which follows the same collating sequence as UTF-32. UTF-16 does not.

*Wednesday update:*

PJ: Fundamental way to prevent library explosion is to just say no, and not do it. Vendors will still feel the need to extend here. There is also a lot of problems with the ctype mechanisms ... sometime toupper(), for example, will produce more or less characters on the output than input.

UTF-16 is not a proper wide character encoding. Everyone knows this. What we did in the TR was enough to keep those who wanted some UTF-16 support happy.

Rich: as soon as we have char16_t in the standard, vendors will feel pressure to add all the interfaces everywhere. HP implementation has wchar_t with 32 bit support, and everything works.

Tom: 16 bit support was the important thing for the TR, 32 bit was not the preferred type. Sometimes you have to go to 32 bit (surrogate pairs), but 16 bit is the preferred form.

Clark: We can't wish away UTF-16. Even if we don't provide any support, it is still out there. Just ignoring it won't make it go away, so it seems harmless to add it in a limited fashion.

Rich: the Chinese feel like second-class citizens in the UTF-16 world. They don't have representatives here, but we shouldn't ignore them.

Tom: Our relationship with the character set people; we aren't in the character set business, SC2 is. We haven't heard any change of heart with respect to UTF-16 from them. They still want it. If we don't include char16_t we will be a laughing stock.

Randy: support for adding UTF-16 support. The dust hasn't settled on the issue at this point. The TR is incomplete, but useful.

PJ: C has tried since the beginning to be character set neutral. Now we are being asked to recognize that one encoding (ISO 10646) is better than others. Just as we recognize that POSIX is special, we should acknowledge that 10646 is special.

Rich: we already recognize 10646. There are feature test macros with the name in them.

JB: We need to something with the TR according to the rules.

Nick: UTF-8 is a better solution, and we already have support for it.

Straw poll: Should TR19769 be incorporated into C1x?

For: 11

Against: 2

Abstain: 4

ACTION: PJ to turn TR19769 into editing instructions.

9.12 Adding ***_Pragma()*** operator to C1X (WG14/N1281)

Paper was submitted by DIN. Currently the standard actually forbids string concatenation in the form suggested by this paper, but the concept does appear to be useful (and wanted).

Is it implemented anywhere? Not to our knowledge. Looks like a good idea, though.

We should ask the author to come back with detailed words (Randy would be prepared to help), and to solicit implementation experience (e.g. gcc).

There is existing practice in MSVC for __pragma(sequence of pp-tokens).

9.13 ANSI C and the "task" or "thread" storage class (WG14/N1288)

Thread local storage is covered in N1287. Perhaps it should be further integrated into the language. Clark described the C++ thread_local proposal.

We should try to be compatible with C++ here.

ACTION: Clark Nelson to develop a proposal for C based on WG 21 N2545.

Nick asked to try and use the same keyword as C++. Tom pointed out that C++ would be likely to pick a name that was from the user namespace.

9.14 anonymous struct and union in C and C++ (WG14/N1289)

We agree that anonymous structs and unions are useful. David Keaton said he had been trying to add them to C99, but ran out of time to get them in at that point.

Straw poll: should we have a proposal on anonymous unions:

For 16

Against 0

Abstain 1

Should we have anonymous structs

For: 11

Against: 0

Abstain: 6

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

Clark pointed out that WG 21 N2544 may have some useful terminology to use.

9.15 ***FLT_RADIX*** value applies only to generic floating-point types (WG14/N1294) (Peterson)

*Tuesday*: Equivalent changes have already been made, so Rich withdrew the paper.

*Wednesday update:* Rich withdrew his withdrawal ... Edison would like to add the preamble suggested by this paper. However, the functional problems have been resolved.

9.16 ***#pragma __STDC__ UNSUFFIXED_FLOAT_CONSTANT_IS_DECIMAL64*** (WG14/N1295) (Peterson)

Addresses some fall out from the removal of TTDT following the last meeting. This adds a pragma to give users more control, which seems to be a good thing.

Consensus is to add this.

Should the default be implementation defined? Rich strongly opposed, Douglas in favor. It would be more consistent with the other pragmas. Standard currently defines the on-off switch as having "DEFAULT". Default could mean the same as OFF, or it could be Implementation defined. Current wording actually allows DEFAULT to be OFF without change. Update the [ON|OFF] to *on-off-switch*.

Name is very long, but it is utterly unambiguous. Abbreviate to FLOAT_CONST_DECIMAL64.

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

9.17 C99 consistency changes in TR 24732 (WG14/N1293) (Peterson)

This paper has a number of relatively minor editorial issues to make the TR more consistent with IEE 754 terminology, and C99 terminology.

Only point 5 is substantive.

Fred and Edison agree that all five points seem to be correct. ACTION Edison to apply edits as described in N1293 to the Decimal Floating Point TR.

9.18 Updated DTR 24732 (WG14/N1290) (Kwok)

Walked through the most recent edits. All appear OK.

Rich noted that there is another issue related to N1294, with respect to the printf %a and %A specifiers and FLT_RADIX.

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

Update on IEEE 754r (again): Mike Cowlishaw has reported overnight that IEEE 754r recirculation ballot finished last Saturday, and it met approval criteria. Final recirc before a June meeting, which is expected to approve the spec. However, we have been burned in the past. We should do the paperwork to ask for an extension, and ballot as soon as possible.

Are we ready to include this in C1x? Not ready yet for inclusion. Need to get some implementation and usage experience. Fred points out that there are two APIs that are not really decimal FP related, and might want to go into the revision: xxx_SUBNORMAL_MIN and xxx_MAXDIG10. Both Fred and PJ have noted problems with these not being in the standard.

Straw poll: should we move these two things to the C1x draft:

For: 10

Against: 0

Abstain: 7

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

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

### 9.19 N1271 *fpclassify() with binary and decimal FP*

This is a paper from Fred.

Paper describes a problem with fpclassify.

PJ already has an action item to work on type generic macros and overloads for C.

ACTION: Fred to continue to work on N1271, coordinating with PJ with respect to type generic macros, and to produce a fully-formed proposal.

The example in C99 7.12.3.1 para 4 is confusing. Should it be removed?

Douglas sees no reason to remove this without decimal fp. However there is a problem if sizeof a float is the same as the size of a double or long double. There is hardware that shows this.

Straw poll on removing example:

For: 16:

Against: 0

Abstain: 2

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

## 10. Defect Report Review

### *10.1 Discussion on reflector email SC22WG14.11414-11419*

### *10.2 Review/Resolve defect reports*

Randy to chair the defect session.

Start with items in REVIEW

**344**: Move to **CLOSED**.

Discussion on how to deal with closed defects for inclusion into C1x. JB will produce an internal only document that looks a lot like a TC, but is never published. The editor will then apply the edits as a batch.

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

**341**: Update to suggested TC, adding a para break before “Also add a forward reference to "type names (6.7.6)”

We cannot avoid make a change, and this seems to be a safe one. There does not appear to be any far-reaching consequences. Not altogether happy with the aesthetics, but this seems to be a satisfactory way forward.

Move to **CLOSED**.

**339**: We agree that this is the same issue as DR328 (which is CLOSED but not PUBLISHED). Move this to **CLOSED**.

**335**: We have only answered one of the questions. Do not appear to have an answer for "What is the maximum width of a _Bool bit-field allowed that does not cause a constraint violation?"

Previous committee discussion said "The width of a **_Bool** bit-field is at most the implementation defined width of the type **_Bool**."

6.7.2.1 para 3(including published Tcs) now states this. Add to committee response:

6.7.2.1 para 3 states: "The expression that specifies the width of a bit-field shall be an integer constant expression with a nonnegative value that does not exceed the width of an object of the type that would be specified were the colon and expression omitted." <<para break>>
Therefore the width of a **_Bool** bit-field is at most the implementation-defined width of the type **_Bool**.

Since we are only moving a sentence from the committee discussion to the committee response, and that the submitter is happy with the response, move this to **CLOSED**.
Consider the Open defects
345: for question 1, we now believe (contrary to Spring '07 discussion) that the standard is wrong here.
Possible change: 6.9.1p9 "which is in effect declared at the head of the compound statement that constitutes the function body*"* change to "which has scope that begins just after the completion of its declarator, and cannot be redeclared ...". Para now reads:
"Each parameter has automatic storage duration. Its identifier is an lvalue, which is in effect declared at the head of the compound statement that constitutes the function body (and therefore which has scope that begins just after the completion of its declarator, and continues through the compound statement that constitutes cannot be redeclared in the function body except in an enclosed block). The layout of the storage for parameters is unspecified.

For question 2, we believe that the scope of the function name f should start *before* the open brace of the function body. This does allow auto variables to mask the name of the function. The scope of a parameter must end at the end of its declarator, and must continue to the end of the function body.
Clarify that the scope of a function name begins at the end of its declarator (which is the right parenthesis).
Make some sort of change to 6.2.1 para 4:
If the declarator or type specifier that declares the identifier appears within the list of parameter declarations in a function prototype (not part of a function definition), the identifier has *function prototype scope*, which terminates at the end of the function declarator.

Idea is to extend function prototype scope to cover this case. This means that we need to examine every use of function prototype scope to make sure we don't break that! (particularly 6.7.5.2 Para 5). C++ has already covered this case (3.3.2 Para 2 of N2588).

ACTION: Clark to propose new words for DR345, based on the C++ solution.
Thursday update: Clark presents his analysis of the situation. We have already got all of the normative requirements needed. For Question 1, there is a suggested footnote, and we should include this. it looks like the footnote is normative, though it isn't really. Add some references to the footnote showing why it isn't normative.

Question 2 proposal seem to be acceptable.

Accept Clark's write-up as the Technical Corrigendum for this DR, and move to **REVIEW.**

342: We moved this to review in Kona, but there was a note in the minutes that the response was too brief. This note actually referred to the previous version, and Randy had fulfilled an AI in Kona to reword. Therefore move *again* to REVIEW (at least with respect to C99). *REVISIT TOMORROW*.

Thursday update:

Rich reports that his compiler segfaults in this case!

Some compile OK, others give diagnostics.

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

Question on whether the revision should make this explicitly undefined (i.e. add text to that affect). Lot of discussion, no decision.
ACTION Tom to develop a proposal to remove K&R functions and making VLAs optional.
Straw poll: who would like to see a paper on making VLAs optional?
For: 6
Against: 5
Abstain: 7

340: It appears that the current implementations differ in this area. Gcc works one way (as described), EDG works another. Probably best solution is to make this undefined behavior. Describe the case in 1145 as not having a composite type. Make this explicitly undefined.
REVISIT TOMORROW
Thursday update: see 342 above.
338: the idea of trap representation was intended to help not to confuse. Rich offered to respond to Doug's email (11380) point-by-point. David believes it not necessary. Clark points out that there are really three possible states for a variable: has a value, has a trap representation, and uninitialized. It is possible that uninitialized is the same as one or the other of the other two.
C++ Core Issue 129 also speaks to this issue.
Footnote 41 in 6.2.6.1 Para 5 is also relevant.
ACTION Tom and Rich to propose a new TC for DR 338.
Update: Rich has proposed words. We like these words (wiki, dr388_response.txt). Use these and move to REVIEW.
334: This needs fixing. Possible it might fit into the work PJ is doing on type generic/overloads. Bill volunteers for additional work (it is a necessary part of that proposal). Add to committee discussion: “This matter will be addressed in a future revision of the standard. See WG 14 Nxxx”. Leave open until “xxx” is a known number!
ACTION JB to provide list of DRs that are to be handled in C1X.

**329**: Previously it was noted that there is no rounding requirements for some of the cases mentioned. We believe that the proposed TC is correct, and nobody has found any additional rounding requirements. Move this to **REVIEW**.

314: Both Joseph and Randy have previously promised (or at least suggested) papers on this subject. Current response is good. However, is this is an issue that should be addressed in C1x? Looked again at N1237 (reviewed in Kona).

- publish an answer that C99 is unspecified/undefined, and we will fix it in C1x
- just come up with a fix for C1x

We agree it should be possible for an implementation to combine the translation units of a program into a single internal representation and optimize that, unifying structure and union types across translation units where required to be compatible in the process.
Normally, it is possible to create a single translation unit equivalent to multiple translation units, after renaming static variables and functions and structure and union tags. DR314 question 3 is an example where such renaming is not possible. My proposal is to add a requirement that it is possible.

JB notes that this has been hanging around for several years. Needs someone to actually come up with the right words.

The current Proposed Response is good, but still needs direction for C1x.

Reminder of earlier action from Kona: 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.

Change answer for question 3 to “This will be addressed in a future revision of the standard. See WG14 Nxxx.” Move to REVIEW. Do not close until xxx is a known value!
Thursday

Several papers on the wiki.

DR_Unicode: Missing Iterator Interfaces

This is a potential defect report from the Austin Group.

The assertion that there is no wide support for this is incorrect. There is a problem with the API. It is widely implemented, and most users have understood the problems/limitations with UTF-16 in the presence of surrogate pairs.

Nobody has adopted char16_t and char32_t and not made them 16/32 bits respectively, nor used any encoding other than UTF-16/UTF-32 respectively.

The points raised are genuine; we need to make sure that C follows C++'s lead and make Unicode mandatory. Make it a defect; proposed Committee Discussion should state “Future revisions of this TR (or editorial actions incorporating this TR into a future revision of the C standard itself) should mandate Unicode encodings.”

This DR is still in a mess, and needs rewriting, and other parts to be submitted.

ACTION: Rich and Nick to resubmit the DR on Unicode issues, and any other related issues.

EncodePointer: Dealing With Pointer Subterfuge
The issue is that storing a function pointer may represent a security risk. This paper is in response to the action item on Arjun. The solution presented encrypts function pointers. This is not a full solution, and suffers from key management issues (per process key, including timestamps), and the plaintext function pointer is visible at some points. However, it greatly improves overall security.

Tom feels that adding something like this to C1x would be beneficial.

PJ points add that for minimal systems (e.g. embedded), the encrypt/decrypt functions need not do anything.

David pointed out that the code-generator could automatically encrypt/decrypt pointers as it wants anyway. Is this just QoI? The performance implications of always encrypting/decrypting pointers would be prohibitive. Not to mention cross-language issues.

Sentiment for adding APIs.

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

FLTDEN: Two Sets of Macros for <float.h>

This is the paper to answer the AI on moving the non-decimal FP material out of the TR into C1x.

The sentence "Their values are typically, but not always, FLT_MIN * FLT_EPSILON, DBL_MIN * DBL_EPSILON, LDBL_MIN * LDBL_EPSILON, respectively." should be removed (it could be included in the rationale).

ACTION: Fred to tidy up the FLTDEN paper and present editing instructions to the editor.

LONGJMP: longjmp() from signal handler

This paper requests conditionally codifying existing practice to permit longjmp from a signal handler.

POSIX does not include longjmp (or siglongjmp) in the async-signal-safe list.

Straw poll:
Fred to do something: 9-0-8

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

PTRFLOAT: Conversion between pointers and floating types

What happens if you try to convert a pointer to a float or vice versa? Should this be a constraint violation? It has always been undefined. Actually making it explicitly undefined would be a minimum step.

Straw poll: should it be a constraint violation: 11-2-4

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

COMPMAC: **Various models used for the [type-generic FP] comparision macros of 7.12.14.**

This is a contribution to help with the type-generic/overload AI for PJ. Thanks Fred!

No further explicit action required.

Email SC22WG14.11416 – Initializing Static Data

Unions only initialize the first element to zero or NULL pointer as appropriate.

This means that if the first element of a union is not the largest element, some of the static data is uninitialized.

Discriminated unions (where the first element describes the type of the union) will frequently suffer from incomplete initialization as a result.

PJ remembers that this was discussed when the defect was being processed. In almost all real-world cases the right thing is done. We just don't carry any guarantee.

Tom believes C++ describes a multi-phase initialization process, which starts by flooding everything with zeroes. Clark described this in more detail ... not flooded with zero bytes.

Randy argues that the current words are right, especially in the case of mixing floating point values, pointers and integers.

Willem points out that the problem is more to do with mixing different sizes of integral types. We made previously conforming (to K&R) code non-conforming. People find this surprising.

There is a problem if there is any pointer or floating point value in a union ... a zero flood will not produce validly initialized data. It becomes a QoI issue. There is no requirement that the values are not initialized.

Keith suggests adding a footnote to highlight the problem.

Rich pointed out that this was a quiet change, which we didn't really notice.

The argument is that programs that have unions like this are inherently non-portable (because of the float/pointer issue), and we have now put people on notice.

Should state that the first element is initialized as currently, the remaining bytes, if any, are set to zero without regard to their type.

Do this to structs as well.

All of static memory is filled with zero. Then floats and pointers are patched.

This does users a favor; since it does what everyone expects and believes is done already.

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

***SC22WG14.11422*** **Signed char and padding bits**

There appears to be an unintended difference between C and C++ w.r.t. padding bits. C++ explicitly forbids them; C appears not to forbid them. We should try to harmonize with C++. Why would anyone ever want padding bits in a signed char? The minimum change to C would be to exclude signed char from the phrases talking about signed integer types permitting padding bits (in 6.2.6.2 p2).

*Change* "For signed integer types, the bits of the object representation shall be divided into three groups: value bits, padding bits, and the sign bit." to "For signed integer types other than signed char, the bits of the object representation shall be divided into three groups: value bits, padding bits, and the sign bit. The signed integer type signed char shall have no padding bits." or similar.

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

ACTION Tom to propose actual text to remove trap representation for signed char types.

IMAGCOMP.TXT **Pure imaginary types with ++, --, <, <=, >=, >**
This changes some possibly expected behavior, incrementing and decrement should not be in this list. Adding one to an imaginary number might be expected to be shorthand for adding one to the complex number (0+xi), and might expect (1+xi) and not (0+(x+1)i). The binary operators behave the other way.
Is there any real-world experience, or reported problem, with this? No.
Fred recalls that Sun observed the relational operator issue in the past.
PJ believes that there would be a problem with not ordering complex numbers, but ordering pure-imaginary. Fred disagrees.
PJ believes that only the classification part is useful. No support for increment/decrement. No support for relational macros. David points out that tgmath already requires imaginary handling.

11. *Open for any unfinished business.*

## 12. Administration

## *12.1 Future Meetings*

### 12.1.1 Future Meeting Schedule

Document on the wiki for the Santa Clara meeting. N1299.
Dates and host for the meeting are:

September 8th -12th, 2008

Mark Terrel

mterrel@cisco.com

1 (303) 503-9855 (cell)

**<u>Meeting Venue</u>**

Cisco

Building 24

510 McCarthy Blvd.

Milpitas, CA  USA 95035

During a revision of both C and C++, it is particularly important to keep the two groups closely aligned. The fall meeting will be followed immediately by the C++ meeting in San Francisco.
We have two offers for 2009. Keith/Plantronics has invited us in Fall to the Santa Cruz area. Edison/IBM is working on a Canada invitation to Markham, Toronto in April. Still looking for European hosts for 2010.

### 12.1.2 Future Agenda Items

### 12.1.3 Future Mailings

Post Delft- 5/16.

Pre Santa-Clara – 8/11

## *12.2 Resolutions*

### 12.2.1 Review of Decisions Reached

### 12.2.2 Review of Action Items

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

OPEN

ACTION: PJ to write paper on quick_exit.

OPEN.

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

OPEN.

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.

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

OPEN.

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

OBE.

ACTION: Robert Seacord and David Keaton to write a paper on static assert.

OPEN

ACTION: Tom to provide list of undefined behavior that should become constraint violations to match C++ usage.

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

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

Under way.

ACTION: Tom and Arjun to write a paper on hiding function pointers in regard to N1279 constructor/destructor attributes.

CLOSED.

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

OPEN

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

OPEN

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

OPEN

ACTION: PJ to present actual edits for threads for the next meeting.

OPEN

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

OPEN

ACTION: PJ to turn TR19769 into editing instructions.

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

In Progress

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

OPEN

ACTION: Fred to continue to work on N1271, coordinating with PJ with respect to type generic macros, and to produce a fully-formed proposal.

OBE

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

OPEN

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

OPEN

ACTION: Clark to propose new words for DR345, based on the C++ solution.

DONE

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

OPEN

ACTION: Tom and Rich to propose a new TC for DR 338.

DONE

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

OPEN

ACTION: Rich and Nick to resubmit the DR on Unicode issues, and any other related issues.

DONE

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

OPEN

ACTION: Fred to tidy up the FLTDEN paper and present editing instructions to the committee.

Dup of earlier.

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

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

OPEN

### 12.2.3 Thanks to Host

Acclamation for Willem Wakker, ACE, AT Computing and NEN. Another successful meeting!

### *13.3 Other Business*

### 14. Adjournment

Adjourned at 14.57, Thursday April 17..

## Agenda for the J11/U.S. TAG Meeting, Tuesday April 15th 4:04pm

Attendees:

| | | |
|---|---|---|
| | | |
| John Benito | Blue Pilot | |
| Robert Seacord | CERT | |
| Mark Terrel | Cisco | |
| P. J. Plauger | Dinkumware, Ltd | |
| Tana L. Plauger | Dinkumware, Ltd | |
| Christopher Walker | Dinkumware, Ltd | |
| Cecilia Galvin | Freescale | |
| Rich Peterson | Hewlett Packard | |
| Edison Kwok | IBM | |
| John Parks | Intel | |
| Clark Nelson | Intel | |
| Arjun Bijanki | Microsoft | |
| Keith Derrick | Plantronics | |
| Tom Plum | Plum Hall | |
| Bill Seymour | Tydeman Consulting | |
| David Keaton | self | |
| Randy Meyers | Silverhill Systems | J11 Chair |
| Douglas Walls | Sun Microsystems | J11 IR |
| Fred Tydeman | Tydeman Consulting | J11 Vice Chair |
| Nick Stoughton | Usenix | Secretary |

1. Select US delegation for the next two meetings.
   Not required at this meeting.
2. 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.
3. Discuss if J11 should submit the CERT Secure Coding Standard to WG14 as a candidate for publication as a Type 2 or Type 3 technical report.
   Is the type 2 or 3? Type 3. Document is still under development, will go to Addison Wesley shortly. Robert believes that having some sort of ISO/IEC imprimatur on it will help give it weight and gain acceptance.
   Tom agrees that this is an important area, and a type 3 TR would be a good thing to have. This deals explicitly with security issues, rather than safety ones.
   Keith agrees that WG14 should be involved with this, but is it typical WG14 type of

business? WG14 has never done a type 3 TR before.
Nick observes that OWG-V is considering asking to become a full WG at the next SC 22 ... and this might belong to them in some respect anyway. However, it is a C specific document, so where should it go? PJ observes we are busy, and may not have the bandwidth to work on it.
J11 believes the document has merit, would be worthwhile to progress to a TR type 3. However, it is unclear that WG14 have the bandwidth to handle it for some time.
Nick points out that NP is not needed till the document is ready. JB counters that getting the 5th country is hard, and doing the work only to fail is bad. He does not subscribe to this method.
Clark thinks the document is good enough without ISO/IEC. Why do all the work for little extra gain?
Robert points out that it is a source of material for C1x. The Japanese NB has helped in the development if the document so far.
Keith notes that we can mine the document for revision help anyway, without progressing it. Can we feel out the other NBs to see what level of support there is?
David thinks of this as a C binding to security. It completes our complement of documents, and is an important addition.
This should be an international effort.
Clark is already concerned about the number of action items that don't get resolved meeting to meeting.
Straw poll on who has time to work on this project? 4-12. Take no further action at this time.

4. Discuss PL22.

Everyone should try to attend the first meeting. We need to make sure that something bad doesn't happen to us, as far as is possible. The initial meeting is April 28.
The current situation is that the US has reorganized itself to allow individuals to increase their participation while reducing the dues payable for doing so.
There was a lot of confusion over the reasoning for this change, and it is certain that it will take some time for the new methodology for working.
Still a big question of who gets to make recommendations (TC or TG).

5. Form US position on FCD 24747

Motion: J11 recommends the U.S. vote to approve FCD ISO/IEC 24747, Information technology - Programming languages, their environments and system software interfaces - Extensions to the C Library, to Support Mathematical Functions

Mover: P. J. Plauger (Dinkumware)
Second: John Benito (Blue Pilot)

Roll Call vote:

Blue Pilot - Yes
CMU/SEI - Yes
Cisco - Abstain

Dinkumware - Yes
Feescale/Metrowerks - Yes
Hewlett-Packard - Yes
IBM - Yes
Intel - Yes
David Keaton - Yes
Microsoft - Yes
Plantronics - Yes
Plum Hall - Yes
Sun Microsystems - Yes
Tydeman Consulting - Yes
USENIX Assoc. - Yes

| | |
|---|---|
| Yes: | 14 |
| No: | 0 |
| Abstain: | 1 |
| Total: | 15 |

Membership eligible to vote: 17

Motion passes

6. 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.

7. Adjourn at 17:09.