InterNational Committee for Information Technology Standards (INCITS)
Secretariat: Information Technology Industry Council (ITI)
1101 K Street NW, Suite 610, Washington, DC 20005
www.INCITS.org

WG14 2187
INCITS PL22.11-2018-00007
Date: 2018-10-18
Reply To: Rajan Bhakta
PL22.11
Email: rbhakta@us.ibm.com

Draft Minutes for 15-18 October, 2018
MEETING OF ISO/IEC JTC 1/SC 22/WG 14 AND INCITS PL22.11
WG 14/N 2307

Dates and times

October 15-18, 2018 09:00-12:00 – Lunch – 13:30-16:30

Meeting location

Robert Mehrabian Collaborative Innovation Center (CIC)
4720 Forbes Ave
Pittsburgh, PA 15213
USA

Meeting Information:

Venue information: N 2247 N 2284

Local contact information:

Daniel Plakosh <dplakosh@sei.cmu.edu>

1. Opening Activities

1.1. Opening comments (Plakosh, Keaton)

Daniel welcomed us to PIT, and WG14.

1.2. Introduction of participants/roll call

Name	Organization	NB	Comments
David Keaton	Keaton Consulting	USA	WG14 Convener
Daniel Plakosh	SEI/CERT/CMU	USA	WG14 ISO eCommittee Secretary
Blaine Garst	The Planet Earth Society	USA	
Rajan Bhakta	IBM	CA	PL22.11 Chair
Clark Nelson	Intel	USA	
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Barry Hedquist	Perennial	USA	PL22.11 IR
Tom Plum	Plum Hall	USA	Phone
Martin Sebor	Red Hat	USA	
Larry Jones	Siemens PLM Software	USA	WG 14 Project Editor
Aaron Ballman	GrammaTech	USA	
Clive Pygott	LDRA	UK	Phone
Jens Gustedt	INRIA	France	
Robert Seacord	NCC Group	USA	
David Goldblatt	Facebook	USA	
Victor Yodaiken	FSM Labs	USA	Wednesday morning only
Peter Sewell	U. Cambridge	UK	Memory model study group
Ryan Steele	SEI	USA	
David Svoboda	SEI	USA	
Lori Flynn	SEI	USA	
Lars Bjonnes	Cisco	USA	
Will Klieber	SEI	USA	
Herb Sutter	Microsoft	USA	WG21 Convener, Phone
Niall Douglas		UK	Guest of WG21 Convener, Phone

1.3. Procedures for this meeting (Keaton)

The Meeting Chair and WG14 Convener, David Keaton, announced that procedures would be as per normal. Everyone was encouraged to participate in the discussion and straw polls.

Straw polls are an informal WG14 mechanism used to determine if there is consensus to pursue a technical approach or possibly drop a matter for lack of consensus. They are voted on by a show of hands for people that approve, reject or abstain, respectively (denoted by #approved/#reject/#abstain in the minutes) on the poll question. Straw polls are not formal votes, and do not in any way represent any National Body position. National Body positions are established in accordance with the procedures established by each National Body.

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

http://www.incits.org/standards-information/legal-info

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

The primary emphasis of this meeting was to begin a working draft for the next revision of the C Standard, review the progress of our subgroups and work on Defect Reports.

Participants need to register on ISO's site for this meeting.

David Keaton is the meeting Chair.
Rajan Bhakta is the Recording Secretary.

1.4. Approval of previous minutes [N2239]

The previous minutes are approved by unanimous consent.

The final approved Spring 2018 minutes are N2239.
(Ballman/Garst)

1.5. Review of action items and resolutions

Carry over:

1) Convener: Coordinate with WG21 for C/P papers: Still open.
2) Larry: Paper for N2161 (left shift). Still open.
3) Convener: CPLEX document. Close. Will be discussed later.

4) ACTION: Blaine to reconcile N 2019 and N 2026 for DR469. Superseded by new action item, see below.

5) ACTION: Barry to make the document on specific differences between C++ and C available to us. N2249.
DONE

6) ACTION: Jens to extend work on DR 486 and DR 495 for a more complete document that addresses inconsistencies concerning atomics (language and library).
OPEN

7) ACTION: Blaine, in relation to N 2019, DRs 469, 479, and 493, to write a paper that identifies more apparent problems for threads (language and library).
OPEN

8) ACTION: Martin to write up a proposal that forbids new types to appear in calls to offsetof.
OPEN

9) ACTION: Rajan to clarify the resolution of FP-CR20 with paragraph numbers for the occurrence of DECIMAL_DIG in C17 and give a complete list of changes in relation to DR501.
DONE

10) ACTION: Rajan to in view of DR 501, the FP committee should propose a new suggested corrigendum in coordination with the resolution of CR20.
DONE

11) ACTION: Aaron, for document N 2197, to look into the possible intersection with Martin's proposal for assert (N 2207).
OPEN

12) ACTION: Convener to add the proposed changes of N 2197 to SD3.
DONE

13) ACTION: Aaron to revise N 2198 (u8 character prefix) to reflect the discussion during the Brno meeting.
OPEN

14) ACTION: Martin to write new versions of papers N 2190, N 2191, N 2192, N 2193 that take their interrelationship into account.
OPEN

15) ACTION: Convener to update SD3 to the updated documents N 2214, N 2215, N 2216 and N 2217.
DONE

16) ACTION: Martin to for N 2209, integrate the issues that were raised during discussion.
OPEN

17) ACTION: Florian to work on updates for N 2225, N 2227 and N 2228.
OPEN

18) ACTION: Convener to add N 2225 to the bullet 12 of SD3.
DONE

19) ACTION: Florian, in the context of N 2226, to propose a paper that features a solution in the line of uselocale from POSIX.
OPEN

20) ACTION: Keld to update document N 2241 in the lines as suggested during discussion.
OPEN

21) ACTION: Aaron to coordinate with WG21 for left shift operator.
OPEN

1.6. Approval of agenda [N2184]

Some corrections in numbering needed.
Adding in PL22.11 agenda items as per discussion last WG14 meeting.
Adding 8.1 for discussion of PL22.11 Systematic review documents.
Adding N2306 under 3.3.

Motion to approve: Blaine, Barry (Motion, second).
Approved.

1.7. Identify national bodies sending experts

Canada
USA
UK

2. Reports on liaison activities

2.1. SC22

David: Went well, nothing to report.

2.2. PL22.11/WG14

Rajan: Nothing to report.

2.2.1. Conveners report and business plan [N2286]

2.3. PL22.16/WG21

2.3.1. WG14/WG21 Liaison report (C/C++ differences) [N2249]

Aaron: There must be things we missed (Ex. Unnamed parameters). Otherwise in Annex C in the C++ standard.

2.4. PL22

David: Had some telecons.

2.5. WG23

Clive: 3 documents out for vote. Language independent, C and Ada.

2.6. MISRA C

Clive: Close to producing a new version of the MISRA C document which will be based off of C11. Should be out the next meeting.

2.7. Other liaison activities

3. Reports from study groups

3.1. C floating point activity report

- Working closely with 754 (strong liaison, one member is chair and one is the editor!)
- Will continue to bind to IEEE-754 2019 to the TS's that remain after C2X

David: C17 was published in 2018 so it can be called C18 as well. We can now talk about C2X. Plan to rebase to C2X for parts that are still relevant.

3.2. CPLEX activity report

Documents on the agenda.

Clark: Thread based parallelism document was designed to work with either OMP or Cilk as it's goal.

Intel has lost interest in Cilk. The name is deprecated. They are no longer supporting the runtime technology.

Not clear if the principle of the document is interesting anymore.

The study group lost critical mass and hasn't met in over a year, close to 2 years.

Not sure there is any point in continuing work on N2170. No implementations.

Robert S: We did this because we saw a need. Are we replacing it with something else?

Clark: It was a language based approach to thread parallelism and not pragma based.

In the real world, it was library and pragma. This was a way to try to do it in language.

It was a want, not a need.

For SIMD parallelism, need language since it is much harder to do it library based. Not so for thread based parallelism.

Robert S: I thought parallelism was too hard for the average programmer. CPLEX was to raise the level of abstraction to make it easier for the average programmer.

Clark: New syntax didn't directly address what makes parallelism hard.

Robert S: Is the fallback OMP.
Jens: The industry has chosen it and it is used that way.
Clark: For C, it is by library calls.
Ryan: Can we use anything from C++? Task based?
Clark: C++ has higher level facilities like lambdas and libraries that cannot be easily projected into C.
David: Should we cancel the project? Due in a year.
Blaine: I presented a paper for closures. Apple does it. I need someone from GCC to implement it. Does the CPLEX group want to take that?
Clark: The group has pretty much evaporated.
Blaine: Tom S. may want to work on it.
Jens: This changes everything that is already there. It should be a new group.
Should we request ISO withdraw the CPLEX project?
Objections: None.
David: Request that ISO cancel the CPLEX project.
Clark: Array section work: The study group did no work on it.
Does WG14 want to work on it? I am asking WG14 to work on it.
Blaine: Did anyone implement it?
Lars: It's in GCC.
Martin: It is now gone from GCC.
David/Jens: Personal interest in this.
Anyone willing to work on Array Sections:
Clark, David.
Not enough interest to work on it.

3.2.1. Prior effort [N2170]

3.2.2. Array sections [N2081]

3.3. C safety and security rules study group

See N2306 for the report.
Clive: Started January 2017.
David: No deadline since this is not an ISO project.
Robert S.: Not present for part of the year do to other work so could not lead the group.
Aaron: No way to meet without an ISO chair (no telecon facilities from ISO).
Willing to be the chair until the next WG14 meeting.
David: Need a chair to reconstitute the group since the rules require it each WG14 meeting.
Blaine: What about the lack of expertise?
Aaron: Due to telecon issues, we lost participation.
David: Appointing Aaron as chair of C Safety and Security Rules Study Group. Get ISO to give him an account for telecon purposes.
Clive: At least two definitions of safety. Very precautionary approach like existing standards. The other way is type safety. No undefined behavior.
Aaron: The MISRA IP issue is still there. Andrew Banks is the contact, but not sure.
Robert S.: The idea was MISRA and the TS would merge. That changed and MISRA withdrew. This adds a question about the viability of the project since we don't want yet another competing standard. They don't want anything referenced from their document. Ex. Can't use the rule title. One big concept in MISRA is "essential types". It is a basis for a number of other things. Using it could cause IP issues.

David: Andrew seemed to be OK with it.
Aaron: Can we get anything in writing or email to avoid last minute issues.
Robert: Can we send the email to the ISO attorneys to see what they say?
Clive: Andrew is not the right person for legal issues. Need to talk to David Ward for that.
Blaine: If you have active action from MISRA as part of the study group, it seems it is OK.
Robert S.: They don't want the rule title being cited. They feel it is too much information.
Clive: It may be the amount of rules we mention. The online people providing MISRA checking services seem to be operating fine with no legal issues.
David S.: Seems we need to talk to MISRA's lawyers.
Robert S.: They are not responding.
David: We could go forward with Aaron as chair, or close the effort.
Robert S.: Other options like abandoning the scope increase to add safety and keep it as security.
MISRA brings up the level to static analysis passing as the minimum.
David S.: Can list both rules to avoid getting a single rule for both safety and security.
Aaron: We are doing it, but the security side is for vendors, while MISRA is for users.
David: We need to deal with the legal issues once and for all. The study group should propose next meeting whether it wants to drop safety or it has the legal issues handled.
Aaron: The Safety and Security Study Group should propose next WG14 meeting whether it wants to drop safety or it has the legal issues handled.
Aaron: Need a permanent chair or a temporary chair for the Safety and Security Study Group by the next WG14 meeting.
Clive: MISRA works in writing code from scratch. No reuse, no libraries in general. Big disconnect there.

3.4. C memory object model study group

Telecons about every two weeks.
Looking into defining the LLVM internal memory model.

3.4.1. Progress report 2018-09 [N2294]

4. Teleconference meeting reports

None

5. Future meetings

5.1. Future meeting schedule

April 29 - May 3, 2019 – London, England
October 21-24, 2019 – TBD, typically USA

5.2. Future mailings

Post Pittsburgh – 12 November 2018
Pre Spring 2019 (London) – 18 March 2019
Post Spring 2019 (London) – 13 May 2019

6. Document review

6.1. Standing Document 3 - consider whether to pursue items for C2x, and find document champions if so [N 2297]

This document's useful life ends this meeting. Any future proposals will need to be new papers.

1) N1730: Floating types are IEC or not feature test macros: Fred will champion it. We want to see more on this.
2) N1817: Two's complement left shift.
Aaron: C++ hasn't resolved this so leave it until they do something.
We will follow this one up.
3) N1793: Indeterminate values (wobbly bits). Leave it to the Memory Model Study Group.
Memory Model Study Group will champion.
4) N1812: const in signal handlers: Martin will champion.
5) N1736: chars/bytes: Martin will champion.
6) N1865: for loop initialization. In favor of seeing more on this topic: 13/3/1
Martin will champion.
7) N1899: integer width. For the parts not in the floating point TS.
David S.: Needs time to work on this.
Needs to be updated to match floating point TS part 3.
David S. will champion.
8) N1910: alignof on incomplete array types. Consensus for it. Aaron to champion.
9) N1911: Preprocessing line macro. Fred to champion.
10) N1923: Multi-dimension array qualification. Consensus for it.
David: Ask Martin U. If he wants to champion N1923.
11) N1870: TRUE_MAX macros for floats. Fred to champion.
12) N1962: Annex K: Fix or deprecate.
Martin: With contracts in C++ it went in an entirely different direction. Still prefer removal and will champion that.
Fix: Enough interest. Champion: Robert S.
Deprecate: Enough interest. Champion: Martin
Straw poll: Deprecate/Remove, Fix, Abstain. Deprecate/Fix/Abstain: 6/6/5.
Result: Go forward with both deprecate/remove and fix proposals.
13) DR482: Remove macro across include files.
Fred: Functions can span file boundaries and functions can be macros.
Aaron: Unspecified should allow ill-formed.
Clark: We've never done that before.
Fred will champion this.
14) N2008: Typed enums. Clive will champion this.
15) N2034: Trailing commas in macros. (N2160 is the latest)
David: Ask Thomas if he will champion trailing commas in macros (N2160).
16) N2017: Parallelism TS 21938-1. No consensus to include this. Drop.
17) N2043: Out of bounds store definition.
Clark: Seems an out of bounds read would stop and not be considered critical (segfault is OK compared to overwriting something).
Not sure what this was trying to accomplish, so no consensus.
Robert S. Can come back with something new.
18) N2269, etc. Attributes. Aaron will champion.
19) N2074: static in array bounds. Martin will champion.
20) N2078: Floating point TS's part 1, 2. Already going in. Editors will merge it.
21) N2083: Struct with Flexible Array Member allowed to be nested. Martin to champion.
Straw poll: Move forward with nested flexible array members: 11/4/3.
22) N2089, etc.: Specification of pointer provenance. Memory Model Study Group to champion.

23) N2098: Fractional digits in %a. Martin to champion.
24) N2101: Add __has_include. Aaron to champion.
25) N2117, etc.: Floating point TS parts 3-5.

Part 3: Floating point study group is willing to turn this into a conditionally normative annex (new proposal).

Part 4: Mathematical functions:

Takes a lot of work.

17 functions with variations for types (float, double, long double) and encoding (binary, decimal)

Martin: We could keep it in the TS. If we put in the standard, the bar is higher since we have to maintain it.

Blaine: Little harm in keeping it as a TS. We can get more implementation experience.

Robert S.: Like the idea of having it a la carte.

Aaron: Conditional inclusion is really subsetting the language.

Blaine: C11 already went that direction.

Aaron: I think that was a mistake and we shouldn't do it.

David: I agree. I went to a vendor and got hit with this where they wanted C11 conformance without atomics for an HPC bid. Looking to make it profile based via a paper. Ex. HPC profile, Embedded.

Blaine: RISCV{F,A,…} meant {floating point support, atomics support, etc.}.

Aaron: It is also a teachability problem.

Peter: From an object model point of view, the profile model is useful (prefer having selectable configurations). For RISCV, despite being involved in it, I can't tell you what the ACDL, etc. means.

Robert S.: For David's case, guidance won't help. It needs to be a requirement.

For teaching, developers want to hear less about the standard, but more on the implementation.

Straw poll: Add in both proposals for part 4 of the floating point TS as a conditionally normative annex of C2X: 4/6/6. No consensus.

Part 5: Evaluation format pragmas:

Blaine: We need implementation experience for this specific TS.

David: The lack of implementation experience is just how to do it with this spelling.

Jens: It has been out for a while but no one has implemented it as per this TS.

Blaine: There are edge cases. Different ways of handling it by different implementations.

Aaron: In WG21, implementers were wary of TS's since they are not the main standard.

Straw poll: Do we want to add in FP TS part 5 evaluation format pragmas into C2X: 1/8/6. Not moving this forward into C2X.

Part 5: Optimization control pragmas

Jens: Does anyone have these features spelled the same way?

Aaron: Or spelled any way but still a pragma.

Rajan: Not sure.

Martin: GCC does not have support for these pragmas.

Blaine: Looking to keep this as a TS. We could possibly have a solid implementation like what Joseph is working on and we can revisit these.

Aaron: Do we have a clock on the TS's?

David: No, since it has been published. It only starts if we say to ISO we want to revise it.

Straw poll: Do we want to add in FP TS part 5 optimization pragmas into C2X: 0/9/7. Not moving this forward into C2X.

Part 5: Reproducible results
Rajan: Depends on the optimization pragmas
Blaine: Doing this will still require rework since optimization pragmas are not present.
Not looking for the FP study group to rework this without optimization pragma.
Part 5: Rounding direction macro
Robert S.: Like the symmetry to Decimal.
Blaine: Implementations may not have hardware support for this.
Rajan: It is the same as the other rounding modes if the hardware doesn't support it.
Larry: All the rounding modes are optional in the standard.
Fred: It is coming since it is required for augmented arithmetic.
Straw poll: Do we want to add in FE_TONEARESTFROMZERO into C2X as per N2124: 7/1/7.
Editors to put N2124 into C2X.

26) N2123: Alternate exception handling.

Robert S.: How about the existing runtime constraint handling mechanism?
It needs to be retooled to work with multi-threading. It is consistent with the conservation of mechanisms principle.
Rajan: This works on blocks of code, not on functions.
Blaine: IEEE has over specified the things that can happen for an exception. Even though the pragma is ugly, the committee has other things to look at.
Martin: The closest thing would be the Microsoft structured exception handling.
Fred: You can enable trap handling. C does not require traps. SIGFPE is close, but not the same.
Robert S.: How does C++ do it?
Rajan: They defer to C.
No consensus to do anything.

27) C++ harmonization (trigraphs, u8, static_assert).

Trigraphs: Why not drop this or make it optional?
Rajan: Breaks portability.
Aaron: There is the issue with incompatibility with C++ and people putting trigraphs into comments.
u8'x' literals: Aaron: Using [] on a u8 string, you get a char. UTF-8 requires you to encode the first byte to US ASCII.
David: It is ASCII only, not UTF though.
Jens: It is implementation defined if there are other characters in there.
David: We can make it consistent with C character literals and have other characters in there (so int type). Or we can make it consistent with C++ and only encode the first byte.
Aaron: These are of type char.
David: We originally didn't do this since Bill Plauger said we don't do one byte.
Jens: Currently we only have literals for ints and wider types, u16, but nothing for character types.
Lars: How about the C++ proposal for a char8_t and use that for this?
Aaron: Were checking in C++ for doing it with decent performance.
No consensus to move forward for this. If someone wants to, needs to solve the problems above.
static_assert: Done.

28) Future directions.

Jens: Looking forward to removing other things from the standard like K&R, bool not be true/false.

David: We need someone to go through the deprecated and obsolescent features in the standard and track who owns each one. Also to look for items that need to be deprecated. Looking for the champion to come with the list and track who will do what.

Blaine: Do we want input from the editors into which would be easy to remove?

Aaron: No, it should be correctness based, not work based.

From 6.11:

1) New floating point types: Fred: Any sentiment to make it optional for freestanding? Drop this.

2) Linkages of identifiers: Yes, want to do it.

3) External names: Aaron: Should we have a paper to remove the obsolescent? Larry: No, no time limit. It can be a long time before it is obsolete.

5) Storage class specifiers only at the beginning. In favor of this change.

6) Function declarators: Larry: Need it for return function pointers to itself. No.

7) Function definitions: A different problem. Yes.

From 7.31:

8) #2: ATOMIC_VAR_INIT: Jens to champion.

9) Bool redefine macros: Agree.

11) ungetc: If removed it needs to avoid making it undefined behavior. Not adding to the list to do.

12) #2: Realloc with zero: Previously we made changes to the standard for this.

David: The degree of implementation is wide enough to make it unsafe.

We want this.

Blaine: We could add in the restriction for a floating point prefix.

Jens: We could also replace the macro/"ugly" keyword problem.

29) N2186: Evaluation formats.

Editors: Put N2186 into C2X.

Note for Jim from Jens: The diff was not good. Did not match paragraphs.

30) N2197: Harmonize static_assert with C++.

Done under N2265.

Every entry with a champion has consensus to move forward.

6.2. Clarifying the restrict Keyword [N 2260]

Editors: Move N2260 into C2X (no objections).

Should be there by next meeting.

Can also remove ATOMIC_VAR_INIT. Needs a proposal.

6.3. Clarifying Pointer Provenance (Q1-Q20) v4 [N 2263]

Extend the day until 5pm for this topic.

Discussion based on N2294.

Peter: Showing the code go through the Cerberus tool.

Adding provenance to the pointer values. Group agreed with 2.1 being undefined behavior.

Cast pointer to integer types and then back to pointers.

The above example has undefined behavior if provenance is tracked through integers as well.

If provenance is not tracked through int values, the cast from an integer to a pointer has choices.

1) Cast (int->pointer) time check through all addresses and assign provenance if there is a match in addresses (at the time of the cast).

Clark: This is not an interesting problem in the context of the standard. Implementations can do whatever for undefined behavior. Strictly conforming programs are not portable with this.

Jens: Conversion from integer to pointer is implementation defined.

Martin: A loop calling rand() until it matches an existing pointer could now use the rand value to access the object inhibiting optimizations since now they alias and objects are accessed through something without being based on the actual object (ex. A local variable).

Jens: As soon as you have a conversion int/pointer, all optimizations should go out the window and stop.

Peter: By not tracking pointer via integer it got rid of a lot of tricky cases.

2) Make it undefined behavior.
3) Restrict to objects whose address has been taken.
4) (3), as well as "escaped".
5) Give wildcard provenance and delay check until access time.

Peter: For sub-objects (2.3.1) what do people think is the intent of accessing the second member of a struct in a function if passed the address of the first member and cast appropriately and originating properly?

Is this well defined behavior in the current standard: 8/4/1

Jens: Cast to char/unsigned char says you want to access the representation of something so it is allowed.

Martin: It is not if it were a multiple dimensional array and using single dimension addition since the standard doesn't allow it to access a 2D array through 1D pointer arithmetic.

Clark: The standard doesn't necessarily say it is well defined, but without this, there is no reason for offsetof.

Unitialized reads:

Peter: There are trap representations and whether the address has been taken. Reading the text of the standard carefully this does not seem to be the intent of the committee.

The intent in Brno was for it to always be undefined behavior.

Robert S.: It could be indeterminate, or a trap representation.

Martin: It is undefined. If you were reading an 8-bit variable, and test for all 256 values, it may still never match since it could change under the covers.

Robert S.: It is unresolved for C17 and needs to be resolved for C2X.

Blaine: malloc followed by fread is a case too.

David S.: Once the array is read by fread, it is initialized to something. It doesn't necessarily have to be a valid value.

Robert S.: It is handled due to the unsigned char case.

Robert S.: We should get rid of trap representations, and get rid of uninitialized and make things wobbly.

Jens: Can't get rid of trap representations due to _Bool.

Robert S.: Make it no trap representations for integers other than _Bool. Annex L would also prohibit out of bounds reads for uninitialized variables.

Aaron: Does that still apply with the proposed changes for Annex L.

David: Pretty much every optimizing compiler wants it to be undefined behavior.

Aaron: Also for static analyzer tools since it allows diagnosing all those cases as being traps.

6.4. Attributes in C (updating N2165) [N 2269]

Aaron: Attribute names are not reserved identifiers. But it makes it undefined having a macro with the same name as a standard attribute.

Jens: This is progress. How does it handle non-standard attributes.
Aaron: It is QoI for non-standard attributes.
Jens: Future code could use similar names for macros.
Martin: Concern with the vendor namespace: There is not one. Already vendors are using/creating new attributes with leading underscores. The C library does not define attributes on library functions. This means #include'ing a standard header could add in a number of new macros or attributes using possibly colliding names.
Aaron: If it matches C++ I am fine with changes.
Martin: Easy to do. Add in __ to the standard attribute names.
Aaron: C++ does not do that! Users don't like adding underscores. They don't like clutter.
Jens: Can do something similar to the library with a header that defines nice names for the __ versions.
Martin: That doesn't solve the problem for third party tools.
Aaron: I remain unconvinced that this is a problem that is important enough to make a change where users have to keep in mind differences between C and C++.
Jens: Bug reports is not the best place to look for them.
Aaron: Adding _ to everything makes user code like library code.
Jens: How about adding recommended practice? Ex. Have _ versions of the attributes.
Martin: Why not require two leading underscores?
Aaron: Is this being designed for library code or user code?
Martin: Implementers have to provide both forms. This lets users use either form.
Aaron: This would be a difference from C++.
Aaron: Add in a requirement to have underscore versions of the attributes to N2269.
Jens: Would like to clean up the standard as well for any instances that can use these.
Jens: Do we have something that says removing attributes won't change the semantics?
Aaron: "Clear" is the operative word. It is there. Perhaps have it in the rationale? It is hard to do in vendor space as they may experiment with attributes. What it really means "A correct program with attributes remains correct if the attributes were removed."
Jens: For standards based attributes for sure.
Aaron: Calling convention attributes is something everyone has.
Aaron: Good for doing it for standard attributes but not as standardese. In the rationale is good. The guarantee is already in the definition of the attribute.
Jens: _Noreturn is an example.
Aaron: It follows the same principle: Remove it and correct programs still work.
Jens: Attributes change the reading of code. People can get used to knowing they can ignore attributes when reading code. Keywords can be used for semantic impact so reading code will have meaning.
Martin: Does C++ have a guarantee like this?
Aaron: No, but there is confusion there. I don't like the idea of closing off development on there. But I am OK with either.
Straw poll: Have a normative note for saying attributes have no semantic impact (a vote against prefers a non-normative note). 8/4/3. Result: Consensus.
Aaron: Update N2269 to make it normative that attributes have no semantic impact.

6.5. The deprecated attribute (updating N2214) [N 2266]

Martin: The recommended practice added addresses all my issues.
Jens: None of the examples have the string parameter.
Aaron: Add in an example to show the string parameter in use for the deprecated attribute.

Robert S.: There are a number of words like obsolete, obsolescent, deprecated, etc. Do you know how this works with that.

Aaron: Things need to be obsolete or obsolescent before they are deprecated.

Barry: The standard prefers deprecated.

Jens/Larry: The standard uses obsolescent.

Robert S.: Perhaps say obsolescent in the text so people know how the attribute should be used.

Unsafe should mean not deemed appropriate for safety systems.

Aaron: That's going too far. It's up to the user.

Robert: Perhaps add in other terms like safety, etc. to the list of things that could be deprecated.

Fred: Some italics and bold issues for the term deprecated.

6.6. The nodiscard attribute (updating N2215) [N 2267]

Martin: Is it specified how far the implementation is required to go?

Aaron: It is recommended practice. Encouraging diagnostics.

Aaron: Popular in C++. The standard library has this added in a lot of places. Currently I am not proposing this, but we may want to do it later. Microsoft found a very small false positive rate for Windows.

Robert S.: Clearly we should have done discard instead of nodiscard.

Aaron: If we had to do this again from scratch, we probably would.

Rajan: For the function pointer case, what happens?

Aaron: Currently not allowed. If dereferenced, again, not applied.

Martin: Any thought to add this as it can be confusing on where it applies?

Aaron: Attributes are dependent on their function so not really. The only case is for type attributes.

Martin: GCC does this for noreturn. It becomes part of the type so noreturn stays there.

Aaron: GCC has 3 different ways of doing noreturn and they are all different.

Rajan: What about to data pointers? Ex. scanf arguments.

Aaron: No, but I am proposing this just for compatibility to C++. If there is an interesting use case we can add it to C++ and C.

Martin: What about _Noreturn and nodiscard?

Aaron: Allowed.

Martin: There is a valid use case for this. Can't remember it though.

Aaron: This should be good to go once the attributes syntax paper goes in.

6.7. The fallthrough attribute (updating N2216) [N 2268]

Leaving this as is.

Good to be integrated as is once attributes syntax goes in.

6.8. The maybe_unused attribute (updating N2217) [N 2270]

Leaving this as is.

Good to be integrated as is once attributes syntax goes in.

David S.: A smart compiler should know this already.

Blaine: I agree with David S. Underwhelmed by this attribute.

Aaron: Common case is assert. Also struct type's used only in debug builds.

Good to be integrated as is once attributes syntax goes in.

6.9. Proposal for "defensive" Attribute [N 2258]

Aaron: WG21 adopted new attributes that are like this but not exact. 'likely' and 'unlikely' attributes.

Wonder how they would compose or work with these WG21 attributes.

Martin: I don't know how compilers would work with this for code generation.

David S.: This would mean you'd have poorly optimized code when you reach this.

Martin: This would mean discard all state knowledge for this.

It's like the 'cosmic ray' issue.

David S.: It is like volatile.

Martin: This would make more sense if it applied to a function. There is not technology to handle this.

Jens: We are doing the work of the composer. Let the author do something.

Peter: Has anyone implemented this?

David S.: We need a champion here.

Clive: I think I have to be the champion. This came out of MISRA. We have a number of customers why complain about this.

Aaron: I don't see this as a compiler attribute, but more from a static analysis tool.

Martin: Is there precedence for this?

Blaine: We could use enum values in combination like mutexes by OR'ing them together.

David: We need a better use case. We need prior art.

Rajan: The name should also change since 'defensive' has too many other meanings.

David: Tell Andrew what we need to move paper N2258 further.

6.10. Harmonizing static_assert with C++ (updating N2197) [N 2265]

Martin has a paper on assert.

Aaron: I didn't not take the words from Martins paper, but I am fine with changing to match.

Blaine: Typo for the ISO C Standard version number.

Aaron: That typo may apply to all the papers. Should be ISO/IEC 9899:2018.

Editors: Put N2265 into C2X.

Martin: No changes needed to the static_assert macro since it is an object macro correct?

Aaron: Correct.

6.11. Proposal for a new calling convention within the C language [N 2285]

There were some revisions on the reflector.

Came from the error handling discussion on the reflector.

Discuss with N2289.

Niall: Similar to std_error object in C++. This proposal hides the error channel into a new type. It also allows error propagation via a pragma to turn it off and on. This also has an error location. The odd/even coding will likely be dispensed with since it is weird. I like a standardized error object but it should be a magical object with magical properties. Prefer something like C++'s form. Also assuming the _Bool bit-field will not be volatile. This seems like an orthogonal concern to N2289.

Jens: The error coding with odd numbering causes problems since you can't change error numbers in industry. Ex. POSIX.

Rajan: Issues with the type return, opportunity cost, alignment and _Generic selection.

Straw poll: Do we like the general direction of N2285: 0/7/9. No consensus to move forward on this.

David: Let Jacob know what the sentiment for N2285 is.

6.12. Proposal to clarify undefined behavior range for implementations [N 2278]

Victor: Seems to be read as a legal document as a way for compilers to find loopholes to do things that should not happen. Compiler optimizations (like link time optimizations) seem to go to far taking these loopholes where they should not be used. Ex. Rollover of signed integer. Compilers assume it doesn't happen and remove checks for overflow inside loops that do overflow. If in an architecture that traps on overflows, it would be fine, but on Intel and ARM it doesn't. These things add surprises to the language. There are no papers that show optimization is actually beneficial.

Jens: The most important thing in the standard for optimization is the "as if" rule. In favor of adding in the normative text, but we should scale better and try to eliminate undefined behavior. Ex. Remove uninitialized variables.

Aaron: Agree that "as if" is important. The null pointer check is the example where optimization matters. A dereference above means that later checks are not needed.

Victor: No, the standard does not say you have to write error free programs.

Aaron: I need concrete numbers.

Blaine: I am supportive of this. An optimization that changes semantics of a program is wrong. Compilers are assuming you are writing for bug free programs. We should say optimization should have the same functionality as no optimization.

Martin: Both null pointer and signed overflow have options to disable those optimizations. Those optimizations are valid (in the case of null pointer checks).

Victor: I'm not proposing the optimization should be barred. But I'm saying that from a dereferencing a pointer cannot be used to remove null pointer checks. Linux runs in a dialect of C that is not standard since it uses the flags.

Martin: That was always the intent of the committee.

Robert S.: There are three reasons for undefined behavior. Difficult to diagnose, hard to accommodate corner cases, and to allow implementations to have extensions. Perhaps re-categorize the UB and have different rules for them?

Dan: A lot of this is brought up in an MIT paper about UB and optimizations.

Martin: Signed integer optimizations is very important for loop optimizations.

Rajan: There is benefits to signed overflow. Trust the programmer.

Victor: If Chris Latner says it is hard, it should not be considered that regular programmers would understand it. Agree with the idea about documenting the behavior if implementations differ.

Peter: Like the idea, but not sure how to define the translation. UB is a blunt instrument. We can imagine bounding effects in some cases, but not in others.

Victor: I don't intend to bound the effects of the error. I want to bound what the compiler can do with the error. I don't want a problem at point A cause something at point B to translate differently. In practice the null pointer dereference and follow up check for null is valid.

Peter: The standard only defines translation of the whole program, not parts.

Aaron: Can have undefined behavior sanitizers. Can be checked at runtime. Ex. UBSan

Victor: It is more of a debugger tool and not in production code.

Blaine: Micro-optimizations are becoming irrelevant since hardware is doing it internally. SPEC is used as a marketing tool. If optimization causes incorrect behavior, that is wrong. I think the abstract machine could use tweaks. I would love to see sub-dialects that would handle all the flags to get something for production code.

David: Chris L.'s quote was in context of getting people to switch to Swift so it is more marketing.
David G.: Differences between O0 and O3 disallows code semantics. It is not very implementable. If people have strong opinions about knowing if -fno-null-checks is important, I can test it and give it back to this group.
Robert S.: Replacing UB with something else, we could allow choices like trap or give something else valid.
Rajan: The standard allows choice so forcing the same output for O0 and any optimization and forcing those to be the same is very limiting and not good. Ex. Parameter evaluation order. The compiler should be free to choose what is beneficial for that program.
Martin: We are sympathetic and Peter's group is working on a way to do it systematically.
Rajan: How about an attribute to disable these optimizations?
Victor: I want opt-in for some of these optimizations.

6.13. Proposal to make aliasing consistent [N 2279]

Victor: Having aligned pointers is enough.
Martin: The effect of this is after a write to an escaped pointer means that compilers can not assume any other access is ameliorated. Basically it kills type based aliasing.
Victor: You can't write malloc or free. Any reallocate violates the lvalue requirements.
Jens: There is an exception for allocated memory to allow it to change type based on write.
Victor: For my own malloc this is not true.
Peter: You cannot use a random char array as if it were malloc but it should. We are looking at fixing.
Victor: The obvious case is computing checksums on packets.
Peter: Need to balance with alias analysis.
Jens: Can do it with char pointers.
Blaine: C has bugs like order of evaluation. We can fix it. zim is a language that does it. Need bug fixes for correctness. The reflector message about C growing to be C++ instead of focusing on smaller chips that are not even on C99. Ex. Jens can handle default initialization. Say the order of evaluation is fixed.
Robert S.: I like the fact C is trying to become a typed language, but this is moving away from that. This seems to be accomplished by no-strict-aliasing, which every compiler has.
Victor: The current problem is you can't write malloc/free in C. That is a fundamental flaw.
Robert S.: Would solving this problem narrowly be enough?
Victor: In practice it is done with flags.
Peter: If there was a standardized attribute to say no aliasing, would that satisfy this problem.
Jens: C++ had a reinterpret_cast.
David G: An attribute for aliasing would change semantics.
Rajan: Have it as an un-ignorable attribute: Keyword like _Noreturn?
Victor: There is that fundamental inconsistency in the standard about pointers that is the major problem, but I like the un-ignorable attribute.
Fred: I can't find the reference.
Victor: OK with writing a proposal for an attribute-like form for this.

6.14. Proposal to limit optimization to C semantics [N 2280]

Martin: Adding this sentence has no impact on anything. It won't prevent any of the cases you want prevented. This is already the case for well defined programs.

Victor: There is a place in the standard where it says order of evaluation is listed as giving a different result. Should those be taken seriously or not? It could be that you find undefined behavior and then do something other than what is specified in the standard just because you had UB.

David G.: Addition is left associative. The example shows that the compiler cannot reorder to introduce traps in a well defined program.

Blaine: This lays the groundwork for future changes. The abstract machine needs to talk about the properties of memory.

Peter: The semantics of a program has a candidate set of executions, and if any one of those has undefined behavior causes all to have undefined behavior due to code motion. You can make it tighter for specific UB, but not in general.

Victor: Martin is right, but it is more of an admonishment. If you say any UB causes anything to happen, it is an incorrect approach. My first proposal was trying to limit the damage.

Jens: Annex L tries to handle this by limiting the unbounded behavior.

Martin: Were you trying to constrain unspecified behavior?

Victor: No, I hadn't thought of it. Just want it documented. Not to be a surprise. C is essentially unsafe. No way around that. It is designed to do things in an unsafe environment. Ex. PCI address. The compiler should change overflow to something like 'rm -rf'. If the programmer does a write to a hardware address that causes the computer to go on fire, that's fine.

Peter: How do you draw a line between the catastrophic failure and doing something bad but safe? For whole program behavior standard specification, we can't talk about specific cases and talking about 'later'.

Victor: I want the compiler writers to not take advantage of weaknesses in the standard. They should co-operate with the programmer. They should follow the intent of the programmer.

Rajan: It is not programmers vs compilers. Compilers are for programmers. They are customers that make them.

Victor: But compilers should not remove code the programmer put in.

Martin: The other camp is that programmers need the help of the compiler. The analysis depends on the optimization. It can be used to find bugs from the programmers. It has to accommodate the systems programmers and the novice programmer.

Victor: C should not specify what happens on integer overflow. The ambiguity is fine, undefined is fine. I'm not OK with programmer surprises.

Martin: The analysis that depends on the optimizer will stop if you try to stop optimization. That is based on the assumption that undefined behavior doesn't occur.

Victor: I don't see that at all. In the SPEC benchmarks it turned a loop into an infinite loop as an optimization. Later it emitted an error message about the loop.

Martin: The analysis has to benefit the code gen too. Otherwise they'd use a static analysis tool.

Victor: The transformations I am talking about are not optimizations. They make the program incorrect.

Aaron: Sympathetic to the issue. But compilers can't tell the intent of the programmer.

Rajan: Seeing the problem would be helpful. It sounds like some of this is compiler bugs.

Dan: I am surprised. If a programmer writes something they intended it. It is blindingly obvious.

Robert S.: Maybe diagnose undefined behavior in the code. Perhaps make it a requirement for the compilers to follow the TS.

Victor: There was a security bug in the Linux kernel where this happened.

6.15. Zero overhead deterministic failure [N 2289]

Herb: About 50% of C++ users turn off exception handling. That and runtime type information are still not zero overhead. We want to make them zero overhead. We love the exception model, but want to have a way to opt in to a statically known type. Currently the normal return channel is wasted when an exception is thrown. Currently it cannot be deterministically thrown. It is trivial to do for static types, but no way for dynamic. Having this through C would be like a holy grail for systems programmers.

Niall: This also allows dealing with errno. Especially helpful for GPU's that use a non-standard form of C. It would be good to use normal functions and have the compiler use the fails mechanism for errno and causes it to delay the setting of the real errno until a point where you may be able to eliminate it entirely or if not allow more pure calls.

Aaron: What happens with function pointers? When you form a function pointer to something has the fails keyword assigned to it? i.e. Does it affect the functions type?

Niall: It is for backwards compatibility, a newer compiler can create wrapper code to the older code and use the new method for new code. For casts from fails to a non-fails pointer, you will get UB. There should be something that can be done at the compiler level.

Jens: If I want to add to existing code, casting to a non-fails function pointer should work. For non-fails and cast to fails should be UB.

Niall: You can't undo this, since the fails function has extra state and you can't just throw that away.

Martin: I like the idea in the paper and interoperability. My concern is with the ABI changes. How does it work with the POSIX layer or other layers? Any implementation experience?

Niall: Seems good since you can't mix. We are still trying to determine the state of deterministic exceptions. If C wants to lead the calling convention we can use it for the prototype implementation. If C hates this we'd use a C++ approach. Hard to judge without a real compiler. Don't know about the overhead due to code bloat, etc.

Herb: Need to demonstrate ABI compatibility. We are ok with being given a list of concerns to address.

Aaron: I like the paper, but have the opposite opinion to Jens. The failure channel seems to be fundamental to the type and so make it an error to cast. To much loss from fails to non-fails.

Rajan: Does this mean the function signature will be different?

Niall: There should not be any mangling required. It wouldn't help the reinterpret_cast case.

Herb: You could view this as a function type issue, but it could be a calling convention.

Niall: It would be different function parameters between fails and not fails.

Jens: This is already too complicated for C. We should just leave this as a bit that says it failed. Calling convention would not need to change since you have saved registers. The compiler would need to know to check for the flag.

Niall: Do you want an Either type?

Jens: It should be up to the function caller. If the caller needs a union, the caller defines it. If it is simple, it can be re-interpreting the return value in a different way.

Niall: It is already there in this proposal.

Jens: I want to see one proposal for the bit, and another for the Either type mechanism.

Blaine: I support Jens position on this. We have headaches on per-thread errno. There are hard issues if the library function wants to handle errno in a different way. I don't see exceptions as being interesting in C yet.

Straw poll: Does this group like the direction of N2289: 15/2/2.

Rajan: Don't like this from two points: 1) Changes the C model and idea. It is used as the lingua franca across a number of other languages and systems and OS's because of it's simple calling convention. Now there is another burden for everyone to consider at least two difference C calling conventions. One for what's always been there and one for 'fails'. That takes away a fundamental benefit of C which makes is the basis for everything else. 2) The assumption that the side channel bit is always available. This is not the case everywhere and using it does affect other downstream and upstream components like dumps and other things. The bits may not be available and if repurposed it causes issues with compatibility with interlanguage calls.

Blaine: EH should not be part of the language. It should be handled elsewhere. Perhaps have errno cleaned up? Or explore out of band communication through something that has no ABI impact. Can it be done that way?

Herb: The 50% data point for no EH was 20% in total, the rest was in part. It does cause fracture in the community. C++ and Java have the same problem with exceptions, but it is not inherent in the exception model. This is the C way by returning a value. You get what you see with the call. C got it right with return by value. It is just compiler work. For C you have the same issue with not being able to change the function signature. But since errors is really not functions doing what they should do, having an out of band error seems very promising.

Blaine: The opening comments said you have examined other languages, but Swift and Objective C don't require dynamic allocation. I think there are alternatives. I am not sure what the vote showed what everyone wanted to have.

Aaron: There are a number of calling conventions in C. This is just another one of those. The trouble with exceptions are two forms. Throwing/catching and how programmers are good at knowing how exceptions work. This proposal keeps the code flowing downwards which is easier to understand from both cognitive and optimization perspective.

Blaine: How does this work with setjmp/longjmp. Need to preserve the side channel.

Niall: I am an author of Boost. We have done this in there already. There is much less change than some people are thinking.

David G.: These functions return a struct, but pass the failure in a side channel. Why is there ABI concerns?

Herb: I view this as an extension, not a new calling convention. For returning out of band, people can write it by hand. It would become sugar vs writing by hand and that results in errno.

Jens: I was not intending to say we should not have the second level. Just that we can enter the first (side channel) but not the second.

Straw poll: Do we want to see this proposal in two parts (a function returns an extra bit for failure, and the second part being types to return extra information): 6/7/6. No consensus.

6.16. pow divide-by-zero case [N 2271]

David: We just got this. The other items had time to be reviewed is over time. Perhaps put this in a review document then put it in.

Aaron: We should leave it to the document champion.

Blaine: Have a section in the meeting minutes that list papers approved in C2X. Also the editors have a list of changes that went in and what was changed every meeting/every draft.

Larry: That was traditionally an the editors report.

David: If anyone objects for any reason we will delay putting it in for 6 months/until the next meeting.

Result: Put into C2X.

Rajan: Put in meeting minutes any items that will go into C2X from this meeting.

6.17. Min-max functions [N 2273]

Lars: Are these required functions.

Yes.

Blaine: I would like to see a simpler proposal that deals with C2X.
Martin: Is there a better way to design these API's like parameterize and overload?
Jens: For consistency this should be in tgmath.
Fred: It is there in clause 16.
Martin: I am concerned about the _num and _mag types.
Jens: Other features added have a prefix. These don't. They pollute the user namespace.
Looks good. Needs to be updated for C2X.
FP study group: Split N2273 into parts 1 and 2 changes based on the TS's, and a separate one for part 3 that fits the conditionally normative part 3 annex.
Keep the spelling as is.

6.18. Augmented arithmetic functions [N 2274]

This requires a new rounding mode that is only used for these functions.
Jens: _t is not a reserved. By convention, _t is an alias for a type in POSIX and reserved there. We could invent a prefix for this (Ex. aug_).
Robert S.: This is against the convention in C right now to have _t.
Jens: We never have struct tags with _t.
Jens: Object to adding it to part 4 due to the names.
Blaine: IEEE is asking for all these things. Do we want to commit people to build emulations?
David: This is getting into the territory of the latest IEEE standard coming out and not yet finished.
Blaine: We should keep this with the CR's for updates to the next version of the TS.
David: It may be easier to have all future changes in a document for a new CD for the new version of the IEEE spec.
Nothing to do at this time in WG14, but the floating point group may want to bring this forward again for a future revision of the TS.

6.19. C support for IEEE 754-201x [N 2275]

David: Good job on getting ahead of the IEEE spec and minimizing the changes needed. Since we did not vote in part 4, the FP group can do the update to the TS's.
Priorities should be do parts 4 and 5 updates to IEEE and then rebase on C2X and then bring it forward for a new published revision.

6.20. C2X Proposal for new string representations for NaNs [N 2290]

Martin: This was participated by an optimization in GCC which was based in the standard for sprintf output. I cover a lot of overlap in N2301.
Do we want to discuss both together?
Fred: The current draft of 754 has a 'should' for how SNAN is printed and it is SNAN. In the previous version of 854, they said it started with NAN but nothing what comes after it.
Aaron: How much existing code do we think we will invalidate?
Jens: Show a case where they print out only 3 characters for floating point.
Aaron: My case is just after checking for NaN and then knowing it comes out as a 3 character.
Blaine: What is the spelling for infinity?
Rajan/Fred: INF or INFINITY.
Martin: I am in favor of this, but no way of selection of which form for the programmers.
Robert S.: Should we have a separate flag, conversion specifier, etc. for using NaN instead of something else like NaNS. Ex. z.
Fred: The Fortran standard has the same text as ours (C).

Jens: Not in favor of it now. Want to look at Martin's paper.
David: If we do this, should we consider the "should" recommendation?
Straw poll: Should we try to address the issues brought forward in N2290: 5/5/4. No consensus to address this issue.

6.21. Proposal for sub-setting _Thread_local if __STDC_NO_THREADS__ is defined [N 2291]

Jens: Why is this needed at all?
The implementation can just parse and ignore.
Rajan: No, some cases it is expected to be static.
Martin: The use case is for a library.
Martin: Is there precedence for a keyword being conditional. Seems to be late for this change.
Blaine: This is feedback after years. It is a problem.
Jens: It is not a defect.
Straw poll: Do we want to see changes along the lines of N2291: 3/9/3. No consensus for this change.

6.22. Make mblen, mbtowc, and wctomb thread-safer [N 2281]

Rajan: Agree with the goal in principle, but not with the words. For example, mblen cannot be state independent if the state is locale dependent.
Blaine: Perhaps say a call to setlocale for a stateful encoding may also introduce a data race.
Jens: What that suggests is dealing with setlocale is not referred to here. There are two problems: having the state change via setlocale or with the function itself.
Jens: No reason to make the second change as it is already covered in the section preamble.
Fred: Is 'other calls' concurrent, sequential or both?
David: It is for any other call.
Fred: Currently it doesn't say anything about needing to be sequential and that needs to happen.
The paper needs more work.
Issues: Dealing with setlocale, duplicated text about data races with the same function, and the data race with 'other calls'.
Rajan: Perhaps say "not required to avoid data races as long as the LC_CTYPE category does not change" or something similar.
Blaine: This does not seem to do what is intended. It should it be possible to clearly state that you can get data-race free with proper specification.
Blaine: This paper needs more positive assertions of being data race free in the presence of possible changes to/from state dependent encodings. It doesn't seem the words here achieve the goal.

6.23. Additional multibyte/wide string conversion functions [N 2282]

Still require the proper proposal format (including prior art). It is listed in this paper that he has prior art in his compiler.
Fred: Can one wide character be more than one c16?
Rajan: Yes in general, but the wide characters listed here are for c16's if given in context.
Jens: Would IBM be willing to provide the second implementation?
Rajan: Can't speak for my company right now.
The committee likes the direction but wants to see other implementations of this before standardizing it.

6.24. Alignment requirements for memory management functions [N 2293]

Jens: The users who want it portably likely used aligned_alloc.

Blaine: This is why we have aligned_alloc.

David G.: I care for performance reasons.

David G.: Making it forced to be max_align_t would not be good for my use cases. Similar sizes tend to be allocated and deallocated together.

Larry: DR75 says what it says because it says that's what the standard says.

Jens: Uses that need an alignment should use aligned_alloc.

Robert S.: Sounds like we are going to be adopt this proposal. If we don't adopt this we will say existing code is not conforming.

Fred: If you malloc 10 bytes? Long double on Intel does this.

David: It needs to be 8 byte aligned. If they need alignment 16, it would not be correct in the object model.

Jens: I don't think the words here disallow that.

Fred: I don't know if the case would segfault or have a performance hit.

Martin: Are you saying it rounds down to the nearest power of 2?

David G.: In practice you are only talking about 1, 2, 4, 8 since nothing has a larger max_align_t greater than 16.

David: If you allocate 10, and have 8 alignment, it may hit cache line crossing and have a performance hit. But with this change that implementation would align to a larger increment.

Jens: Even for the long double case they do it right.

David: Sometimes they have size 96 and align to 32 bits.

Straw poll: Should we adopt N2293 for C2X in the next WG14 meeting, unless there are objections in the meantime? 15/0/0. Consensus to put it in.

6.25. Library Functions and Compound Literals [N 2299]

Martin: Having implementations recognize that the function is special (a standard library function) is superior to the macro mechanism. C++ does this too, via intrinsics or defining inline. C++ is starting to provide compound literals. Only expecting this for functions that take pointers, not for things like isalpha.

Jens: Since using macros is still allowed, this seems reasonable to me.

Blaine: Looking for a shall.

David G.: A 'may' for a user is a shall for the implementation.

Jens: Can't use 'may'.

Larry: No, you can.

Rajan: Need to say something clearer for the second example to show it is not allowed.

Aaron: I agree.

Larry: The description for macros says any comma separate arguments.

Martin: Will work on the wording.

Rajan: Suggestion is to add an example using VA_ARGS that works as per Jens assertion.

6.26. Aliasing by String Functions [N 2300]

Clang and GCC aggressively optimize away strlen calls by tracking strings dynamically.

Aaron: Do we need to do anything for wmemcpy or anything like that?

Martin: No.

Lars: Any impact to Annex K.

Martin: Not discussing that.

Jens: There could be a problem on platforms with wchar_t is the same as char.

David G.: If extern int n were a union of a char array and an int, it would not work. It would not be about to be optimized for the previous example either (example 2).
Martin: I would have to see the example.
Larry: Even with the new words, if n were a pointer they could still alias (n with s).
Martin: Yes, there will still cases that cannot be optimized.
Robert S.: The first change seems to conflict with the definition of arrays.
Martin: Happy to adjust the wording.
Larry: There are too things that are fuzzy with characters so the second change should not happen. It should use byte.
Martin: Fine to go back to it.
First set of changes for string aliasing look good and need to be reworded as per the discussion above.
Martin: There is still an issue for cases where you don't have an effective type.
David: That's what is lost due to the second change deletion part. It stops storing to malloc to memory.
For the second part, of using bytes in place of characters, seems ready to go.
Rajan: There were issues in the preamble (7.22) that also need to be fixed.
Martin: What needs tweaking?
Larry: 7.24.1 needs to be changed to deal with the 'character' term instances there. Ex. May need to say character or byte.

6.27. fprintf Formatting Underspecified for NaN [N 2301]

Rajan: We already have existing exploitation of NaN(n-char-sequence). OK with swapping the defaults.
Aaron: Prefer the default to be NaN.
Jens: What is the use of the n-char-sequence?
Rajan: An integer code that gives the reason for the NaN.
Jens: Why have this as recommended practice instead of forcing it (why not requiring restricting the amount of output)? There is ambiguity for the character sequence. It should have a way of dealing with ")".
Ryan: Is there a default standard now?
Fred: Yes, as long as NaN is in front, then it is fine to have anything following it.
Fred: n-char-sequence cannot be ")". It is defined in 6.4.2 to not include that.
Aaron: Are the n-char-sequences based on some side channel?
Rajan: No, it is part of the payload.
Martin: I don't consider the adding of a way to choose which way to give the output as a new feature, I consider a bug.
Fred: Printing a number with %f you can get a large number that is a much larger number of characters.
Martin: For NaN all the specifiers give you NaN or NaN(n-char-sequence). There is no way to constrain it.
Fred: Can have the # switch between the two forms of infinity.
Fred: Can use the precision to constrain the output of the NaN/infinity.
Straw poll: Do we want to address the issues brought forward in N2301: 12/0/1. Consensus to address the issue.
Straw poll: Do we want to constrain the maximum amount of output for n-char-sequence? 12/0/2. Consensus to constrain the output.
Jens: Prefer % form. What happens if n-char-sequence is constrained?

Blaine: We should have a macro that gives the maximum size of n-char-sequence.
Straw poll: Do we want precision control or flag control? 7/4/3. Preference for precision control.
Clive: Call flag a hash sign.
Rajan/CFP Group: Write up a proposal for precision control for NaN's.

6.28. nextafterl(1.L,2.L) [N 2302]

We will put N2302 into C2X in the next WG14 meeting barring objections.
Fred: We should do this for the part 1 similar functions as well.
David: Need a paper.

7. Clarification requests

7.1. Discussion on the Clarification Request Process

From this point on, once we close off on all the defects, we should not need the compendium any more.

7.2. IS 9899:2011/9899:2018 Clarification Requests [N 2257]

Blaine: We should get champions for C2X resolved defects.
Rajan: If they have words, we should just have them as is for C2X resolved defects.
DR432: Take with DR467. Fred to champion with a new paper.
DR476: Put in C2X.
David S.: Does this address or affect the clearing password case which is optimized out?
Aaron: No, but it clarifies the problem.
DR488: Put in C2X.
DR494:
Robert S.: Do we not want other parts of the size expression to not be evaluated.
Rajan: This is the same as sizeof mod VLA's. Should not be an issue. We should go forward with this.
Jens: The wording "size expression" is problematic.
Larry: It is in the context of the size of the array. It makes sense the way it is.
Put in C2X.
DR497: Put in C2X.
DR498:
Robert S.: This should go into J.2 as well.
Rajan: We should use the new paper (N2281) and not this one.
See discussion in N2281.
DR500: Put in C2X.
Items marked 'Put in C2X' goes to the editors for inclusion.

DR469: Blaine will look at it as a combined paper.
DR479: See DR469.
DR493: See DR469.
DR486:
Blaine: Would like to drop this from the CR process and wait for a paper.
Jens: I was hoping to not get into syntax, but now may need to if I have to deal with DR495 as well.
DR496: Leave in open.
Martin: The other questions are not answered.

Fred: Martin already has an action item for this.
Martin: We should not advance this unless we answer everything. Joseph asked a question which was not addressed here in the change.
Blaine: Process wise, can we take this committee discussion and the previous one and use both together to solve this problem?
Rajan: Not good to take short cuts with the process.
Blaine: I believe this is make-work and we did address Joseph's comments.
Martin: Different compiler vendors had different readings of the original words. I am not convinced things are ready.
Blaine: I believe this is forward progress.
Barry: In the past if we have a new paper questioning something in a DR we bring it back to open.
Martin: What we have is not good. I am taking on the responsibility of writing a paper to resolve this.
Jens: Can you separate it from this question and the union type question?
Martin: If it makes sense.
Blaine: I will explicitly fold in the October 2017 and April 2018 Committee Discussion and proposed change into the October 2018 status.
DR499: Move to C2X.
Fred: 'their' is ambiguous.
David: 'their' is plural so it refers to the structure members.
DR501: Move to review.
See N2253.
Move the last committee discussion into proposed TC.

In addition to normal CR processing, the following items have new material to consider:

7.2.1. Update to N2108 suggested TC for CR501 [N 2253]

Move to DR501 into review after incorporating N2253.

Martin: Since cast and assignment can remove extra range and precision, does this apply to parameters to function calls?

David: Yes, since it is as if by assignment.

7.3. TS 17961:2013+Cor 1:2016 Clarification Requests [N 2150] (processing is complete)

7.4. TS 18661 Clarification Requests [N 2256]

Blaine: DR12 has a mistake with the wrong committee discussion. It was for DR13.

Rajan: Keep DR's moving forward as is and let the FP group deal with rebasing the changes after we have a working draft.
Consensus to keep the process going.
Review:
DR17: Move to closed (C2X).
DR18: Move to closed.
DR19: Move to closed (C2X).
Open:
DR13: See N2252. Move to review.
David: Should we incorporate Parts 1 and 2 first or make these CR changes first then add it to C2X?

Jim: Give all (CR/DR) changes integrated into the Floating Point TS's by the end of this meeting to the WG14 editors for Parts 1 and 2. Subsequent changes will be based on the C2X working draft.

WG14 editors will endeavor to have a working paper by next meeting.

Since the changes were already present in the log.

DR16: Move to review.

DR20: See N2254. Move to review.

DR21: See N2283. Move to review.

DR22: (Typo: Should be against Part 3) See N2255. Move to review.

In addition to normal CR processing, the following items have new material to consider:

7.4.1. Update to N2213 suggested TC for CFP CR 13 [N 2252]

7.4.2. P1 CR for obsolescing DECIMAL_DIG [N 2254]

7.4.3. P3 CR for obsolescing DECIMAL_DIG [N 2255]

7.4.4. P2 CR for llquantep invalid case [N 2262]

CR24. Move to review.

7.4.5. P1 CR for remainder NaN case [N 2272]

CR24. Move to review.

7.4.6. printf of one-digit character string [N 2283]

7.4.7. P1 CR for totalorder parameters [N 2292]

Lars: Can we leave the TS as is, and fix it in the C2X.

David: For code that doesn't have signaling NaNs taking the address is more clutter in the code

Jens, Blaine, Aaron, Martin: Prefer intrinsics. The issue is taking the address of the function.

Now CR25. Move to review. The change will be reflected in C2X.

8. Other business

8.1. Systematic review

ISO/IEC TS 18661-3:2015, Information Technology -- Programming languages, their environments, and system software interfaces -- Floating-point extensions for C -- Part 3: Interchange and extended types.

Motion: Do

IBM (Bhakta) yes
Perennial (Hedquist)yes
Keaton Consulting (Keaton) yes
Intel (Nelson) absent
Plum Hall (Plum) absent
LDRA (Pygott) yes
Red Hat (Sebor) yes
Tydeman Consulting (Tydeman) yes
SEI/CERT/CMU (Plakosh)yes
Cisco (Bjonnes) yes
GrammaTech (Ballman) yes
The Planet Earth Society (Garst) absent

Motion Passes 9-0-0-12 (3 absent)

ISO/IEC TS 18661-3:2015, Information Technology -- Programming languages, their environments, and system software interfaces -- Floating-point extensions for C -- Part 4: Supplementary functions

Motion: Do

IBM (Bhakta) yes
Perennial (Hedquist)yes
Keaton Consulting (Keaton) yes
Intel (Nelson) absent
Plum Hall (Plum) absent
LDRA (Pygott) yes
Red Hat (Sebor) yes
Tydeman Consulting (Tydeman) yes
SEI/CERT/CMU (Plakosh)yes
Cisco (Bjonnes) yes
GrammaTech (Ballman) yes
The Planet Earth Society (Garst) absent

Motion Passes 9-0-0-12 (3 absent)

9. Resolutions and decisions reached

9.1. Review of decisions reached

Disband CPLEX study group and cancel projects in progress.
Put N2124 into C2X.
Put N2186 into C2X.
Put N2260 into C2X.
Put N2265 into C2X.
Put N2271 into C2X.
TS 18661-{1,2} to have all CR's in closed state as of the end of this meeting to be integrated into a working drafts and given to the editors for inclusion into C2X.
TS 18661-3 to have all CR's in closed state as of the end of this meeting to be integrated into a working draft that is a conditionally normative annex for inclusion into C2X.
Adopt N2293 for C2X in the next WG14 meeting, unless there are objections in the meantime.
Put N2302 into C2X in the next WG14 meeting barring objections.
DR476: Put in C2X.
DR488: Put in C2X.
DR494: Put in C2X.
DR497: Put in C2X.
DR500: Put in C2X.
DR499: Move to C2X.
TS 18661-{4,5} to have all CR's in closed state put into working drafts and rebased to C2X whenever appropriate to prepare for re-issuing the TS's.
CR25 to be put in C2X.

9.2. Review of action items

David: Request that ISO cancel the CPLEX project.

David: Appointing Aaron as chair of C Safety and Security Rules Study Group. Get ISO to give him an account for telecon purposes.
Aaron: The Safety and Security Study Group should propose by the next WG14 meeting whether it wants to drop safety or it has the legal issues handled.
Aaron: Need a permanent chair or a temporary chair for the Safety and Security Study Group by the next WG14 meeting.
David: Ask Martin U. If he wants to champion N1923.
David: Ask Thomas if he will champion trailing commas in macros (N2160).
Aaron: Add in a requirement to have underscore versions of the attributes to N2269.
Aaron: Update N2269 to make it normative that attributes have no semantic impact.
Aaron: Add in an example to show the string parameter in use for the deprecated attribute.
David: Tell Andrew what we need to move paper N2258 further.
David: Let Jacob know what the sentiment for N2285 is.
Rajan: Put in meeting minutes any items that will go into C2X from this meeting.
Rajan: Write up a proposal for precision control for NaN's.

9.3. Identification of PL22.11 Voting Members

Cisco
GrammaTech
IBM Corp
Intel
Keaton Consulting
LDRA Technology
Perennial
Plum Hall
RedHat
SEI/CERT/CMU
The Planet Earth Society
Tydeman Consulting

9.3.1. PL22.11 Members Attaining Voting Rights at this Meeting

None

9.3.2. Prospective PL22.11 Members Attending their First Meeting

None

9.4. PL22.11 Members in Jeopardy

9.4.1. Members in jeopardy due to failure to return Letter Ballots

None

9.4.2. Members in jeopardy due to failure to attend Meetings

None

9.4.2.1. Members who retained voting rights by attending this meeting

None

9.4.2.2. Members who lost voting rights for failure to attend this meeting

None

9.4.3. Members who previously lost voting rights who are attending this meeting

None

10. Thanks to host

11. **Adjournment**
    (Hedquist/Ballman)
    No objections.