MEETING OF ISO/IEC JTC 1/SC 22/WG 14 AND INCITS PL22.11
WG 14: Draft Minutes (N2941)

Agenda from N2915 (http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2915.htm updated to various times until http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2936.htm during the meeting)

Dates and Times

Each day will have a half-hour break from 16:00-16:30 UTC.

Monday,	31 January, 2022	14:30 – 18:00 UTC
Tuesday,	1 February, 2022	14:30 – 18:00 UTC
Wednesday,	2 February, 2022	14:30 – 18:00 UTC
Thursday,	3 February, 2022	14:30 – 18:00 UTC
Friday,	4 February, 2022	14:30 – 18:00 UTC
Monday,	14 February, 2022	14:30 – 18:00 UTC
Tuesday,	15 February, 2022	14:30 – 18:00 UTC
Wednesday,	16 February, 2022	14:30 – 18:00 UTC
Thursday,	17 February, 2022	14:30 – 18:00 UTC
Friday,	18 February, 2022	14:30 – 18:00 UTC

Meeting Location
This meeting is virtual via Zoom.

1.1 Opening Comments (Keaton)

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Notes
Aaron Ballman	Intel	USA	C++ Compatibility SG Chair
Alex Gilding	Perforce / Programming Research Ltd.	USA	
Barry Hedquist	Perennial	USA	PL22.11 IR
Clive Pygott	LDRA Inc.	USA	WG23 liaison
David Keaton	Keaton Consulting	USA	Convener
David Svoboda	SEI/CERT/CMU	USA	Undefined Behavior SG Chair
David Vitek	Grammatech	USA	

Elizabeth Andrews	Intel	USA	
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Freek Wiedijk	Plum Hall	USA	
Lars Bjonnes	Cisco Systems	USA	
Maged Michael	Facebook	USA	
Martin Sebor	IBM	USA	
Nick Dunn	NCC Group	USA	
Rajan Bhakta	IBM	USA, Canada	PL22.11 Chair, scribe
Robert Seacord	NCC Group	USA	
Tom Honermann	Intel	USA	
Aaron Bachmann	Austrian Standards	Austria	Austria NB
Bill Ash	SC 22		SC22 manager
Corentin Jabot	Freelance	France	Guest
Dave Banham	BlackBerry QNX	UK	MISRA Liaison
Eskil Steenberg	Quel Solaar	Sweden	Sweden NB
Etienne Alepins	Thales Canada, Avionics	Canada	
JeanHeyd Meneide	NEN	Netherlands	Netherlands NB
Jens Gustedt	INRIA	France	France NB
Joseph Myers	CodeSourcery / Siemens	UK	UK NB
Kayvan Memarian	University of Cambridge	UK	
Martin Uecker	University of Goettingen	Germany	
Michael Wong	Codeplay	Canada, UK	WG21 Liaison
Miguel Ojeda	UNE	Spain	Spain NB
Nick Stoughton	Logitech	USA	SC22 Austin Group Liaison
Peter Sewell	University of Cambridge	UK	Memory Model SG Chair
Peter Sommerlad	Peter Sommerlad - Better Software	Switzerland	Invited Guest
Philipp Krause	Albert-Ludwigs-Universitat Freiburg	Germany	Germany NB
Roberto Bagnara	BUGSENG	Italy	Italy NB, MISRA Liaison
Steve Downey	Bloomberg	USA	Guest
Ville Voutilainen	The Qt Company	Finland	Finland NB
Wenge Rong	Beihang University	China	China NB

1.3 Procedures for this Meeting (Keaton)

Straw polls taken; anybody can vote.

Only one conversation at a time. Ex. If audio is broken and they are using Chat, don't talk via audio, or vice versa.

Document numbers: ISO outage is still present. Dan is able to dispense document numbers since he knows what the next numbers will be. He can sync with ISO when it returns. For homework, let Dan know it is homework, so he knows it is higher priority.

1.4 Required Reading

1.4.1 ISO Code of Conduct

1.4.2 IEC Code of Conduct

1.4.3 JTC 1 Summary of Key Points [N 2613]

1.4.4 INCITS Code of Conduct

1.5 Approval of Previous WG 14 Minutes [N 2914] (WG 14 motion)

Approved with some typo corrections pending.

1.6 Review of Action Items and Resolutions

ACTION: Ballman to change the WG14 page to promote the full document log. - Done

ACTION: Keaton to investigate status of n2566. - Open for now (will check during the break). Done during break.

ACTION: Add n2761 to the papers-of-interest list. - Open

Assigned to Aaron Ballman.

ACTION: Tydeman to take n2797 to the liaison SG. - Done, but need to follow up.

ACTION: Steenberg to send commentary on the problems with namespaces to the reflector. - Done

ACTION: Meneide or Gustedt to bring a continuation paper about the use of extended integer constant macros in #if. - Done

1.7 Approval of Agenda [N 2915] (PL22.11 motion, WG 14 motion)

Some papers were missed that would have ended up in section 7, so it does not affect this meeting.

5.14 was not intended for C23 so he wants to move it out.

Will not discuss the working draft updates in detail since they are not present.

"typeof" will be discussed before lambdas, and Bachmann papers before typeof.

Jens: Suggest not putting in the working draft updates since we can't see the documents (not present).

Seacord papers will be swapped due to guests needing to attend at certain times.

No objections.

Approved.

1.8 Identify National Bodies Sending Experts

Austria, Canada, China, Finland, France, Germany, Italy, Netherlands, Spain, Sweden, UK, USA

1.9 INCITS Antitrust Guidelines and Patent Policy

1.10 INCITS official designated member/alternate information

See Rajan for any questions or corrections.

1.11 Note where we are in the C23 schedule [N 2864]

Due to the large influx of last-minute documents, this schedule is now unlikely to be met. We should do our best but will probably need an extension.

The ballot resolution meeting will become a normal meeting.

We have only one 9-month extension if we need it.

2. Reports on Liaison Activities

2.1 ISO, IEC, JTC 1, SC 22

JTC1 plenary is virtual.

SC22 is likely going to be virtual as well.

2.2 PL22.11/WG 14

2.3 PL22.16/WG 21

WG21 has had no plenary meetings since the last WG14 plenary meeting.

There is a WG21 plenary happening next Monday (Feb 7). The larger new changes to C++ expected to come out of that meeting are:

P0533R9 constexpr for <cmath> and <cstdlib> (https://urldefense.proofpoint.com/v2/url?u=https-3A__wg21.link_p0533r9&d=DwIBaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=MPb4GyWs7nd-w3OlFPs29W1dB3gHMdsdghhjcQMf428&m=__AAGFnOMAi31jAxlB3l7p3dDU-olIm8vapJFpnbYjM&s=9x11ij7f8GJfTvwPhjZNMrVQ5R-VLJMnCvcPTeEIYZs&e=) -- adds constant expression support to many interfaces originating from the C standard library.

P0627R6 Function to mark unreachable code (https://urldefense.proofpoint.com/v2/url?u=https-3A__wg21.link_p0627r6&d=DwIBaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=MPb4GyWs7nd-w3OlFPs29W1dB3gHMdsdghhjcQMf428&m=__AAGFnOMAi31jAxlB3l7p3dDU-olIm8vapJFpnbYjM&s=NgZ0QAaFTTYd2NJOMUQPe9o4_VHTIw2ivlWOOtcBQU8&e=) -- adds a library function to mark unreachable code. Note https://urldefense.proofpoint.com/v2/url?u=http-3A__www.open-2Dstd.org_jtc1_sc22_wg14_www_docs_n2757.pdf&d=DwIBaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=MPb4GyWs7nd-w3OlFPs29W1dB3gHMdsdghhjcQMf428&m=__AAGFnOMAi31jAxlB3l7p3dDU-olIm8vapJFpnbYjM&s=7BMsk0qjcuGMXJkGNO37FRRs65crD4eXPnghXy135mk&e= touching on the same subject in C.

There are also significant changes to C++ specific functionality such as the new ranges and views STL interfaces. Of potential interest to the UB study group, there is a new library interface for returning expected results from a function in https://urldefense.proofpoint.com/v2/url?u=https-3A__wg21.link_P0323R11&d=DwIBaQ&c=jf_iaSHvJObTbx-siA1ZOg&r=MPb4GyWs7nd-w3OlFPs29W1dB3gHMdsdghhjcQMf428&m=__AAGFnOMAi31jAxlB3l7p3dDU-olIm8vapJFpnbYjM&s=L_8jXffj62kTL4y8OZD9NX_b-RbVqg8qJ2L46I6sGNk&e= . In addition to new features, the Core team fixed seven bugs (six as defect reports) and the Library team fixed 21 bugs (none as a defect report).

WG21 is still on track to ship C++23 on time.

Adding post C23 release schedule (AKA "train schedule") to the agenda for Other Business.
No objections.

2.4 PL22

2.5 WG 23

Nothing changed since the last meeting.

2.6 MISRA C

Continuing covering C11/C18.
Plan to keep an eye on C23 to prepare for the work when it is published.

2.7 Austin Group

2.8 Other Liaison Activities

DIN now has a standard under Creative Commons license: DIN SPEC 3105

3. Study Groups

3.1 C Floating Point Study Group activity report

Continuing to validate WG14 changes are integrated correctly into the standard.
Continuing to handle issues related to C floating point from the wider community.
Starting a backlog of items to handle post C23.

3.2 C Memory Object Model Study Group activity report

Looking to adopt TS 6010 as it is.

3.3 C and C++ Compatibility Study Group activity report

SG22 continues to meet on an almost monthly cadence and processes about three papers a month.
Since the last report (Nov 2021), processed 3 papers (did not meet in January due to holidays).
Next meeting is next week.
Focus has continued to be on papers for C++23 and C23.
Switching gears after February to more forward-looking papers.

3.4 Undefined Behavior Study Group activity report

Met every other week.

Decided to not to submit any papers due to C23, though one is ready.
After this C23 deluge of papers is done, at least one paper will be submitted.
One paper is for uniform compiler switches for UB.
Also looking at a paper for J.2 and another paper for codifying UB.

4. Future Meetings

4.1 Future Meeting Schedule

Please note that in-person meetings may be converted to virtual meetings due to coronavirus considerations.
18-22 July, 2022 – Strasbourg, France (tentative)
Decide whether to make this a virtual meeting. If so, should it be two weeks long?
Took the host's suggestion to meet in May and July virtually and put off what to do in the future based on our ballot schedule.

May 16-20 virtual
July 18-22 virtual

4.2 Future Mailing Deadlines

Note: Please request document numbers by one week before these dates.
Post-Virtual-202201 – 25 February 2022
Pre-Strasbourg – 17 June 2022 (final round of proposals by this date) – Will change based off of the decision in 4.1.
Post-Strasbourg – 12 August 2022 – Will change based off of the decision in 4.1.

5. Document Review

5.1 Working draft updates

Meneide, C Working Draft [N 2912]
Meneide, C Working Draft - Editor's Report [N 2913]
Drafts not posted yet.

5.2 Bachmann, Add timegm() as non-optional part of to C2X [N 2833]

Updated: http://wg14.schlepptop.dd-dns.de:5033/n2833_edited.html

Some belief that these functions can be changed externally by the OS. No real consensus that this is the case.
Some concerns about data races, but also belief section 7.1 is sufficient. No intent for anything new being proposed for data races.

Straw poll: Adopt N2833 alternative 1 (with the typo of mktime->timegm in the returns section fixed) into C23?
16/0/5. N2833 alternative 1 with typo fix goes into C23.

5.3 Bachmann, Deprecate the %n format specifier in C2X [N 2834]

Some implementations like musl libc strongly do not support this change.
Some belief that this is not a security issue, and that the untrusted source is the issue.

Straw poll: Put N2834 with the 'obsolescent" alternative into C23 with the same changes to the wide character functions 7.29.2.1#8 and listing the change in 7.31 (future library directions)?
8/5/7. No consensus to adopt this based on the reasons for abstention (leaning towards no).

5.4 A Provenance-aware Memory Object Model for C (1.5 hours)

TS 6010 continuing discussions (previous working draft for reference [N 2676]) (1.5 hours)
TS updates is at a standstill with no forward progress anticipated. Looking to move forward to ballot with the TS as it is now.

Needs focused long-term attention. The current TS only covers a small fraction of the issues. The mailing list has been dead as people have moved on to other things.

Consideration for a math model of the standard beyond something like Cerebus.

Discussion of adding an annex to the TS for other questions or issues to consider like the uninitialized reads.

Straw poll: Does WG14 wish to see TS6010 working draft (N2676 or something similar) in some future version of the standard?
21/0/1. Clear consensus.
Straw poll: Does WG14 wish to see TS6010 working draft (N2676 or something similar) in C23?
10/8/5. Clear indication people think this is important.

Straw poll: (Opinion) Is WG 14 willing to move TS 6010 to DTS ballot as it stands now?
19/1/3. The committee is OK to move forward.

5.5 Seacord, Annex K Repairs [N 2809]

Belief that changes to the constraint handler are not synchronized. The floating-point environment does inherit what is there from the thread that started the new thread so this may not be an issue.

No real implementors of Annex K.

Straw poll: Does WG14 want proposed wording 2 given in N2809 into C23?
4/7/7. No consensus.
Straw poll: Does WG14 want proposed wording 1 given in N2809 into C23?
2/7/10. No consensus.
Straw poll: Does WG14 wish to address thread specific runtime constraint handling in Annex K?
9/2/8. Consensus to try again.

5.6 Seacord, Identifier Syntax using Unicode Standard Annex 31 [N 2836]

No intent to handle homoglyph attacks in this paper.

Goal of the paper is to keep C and C++ in sync.

Straw poll: Does WG14 want N2836 in C23?
16/1/3. Goes into C23.

5.7 Seacord, calloc wrap-around handling [N 2810]

Editorial correction: The wording should change "nmemb*size" to "nmemb and size" or removing the word “of”.

Straw poll: Does WG14 wish to adopt N2810 as-is into C23 (modulo editorial corrections)?
20/0/0. Consensus.

5.8 Floating point (3 hours total / 20 minutes each)

5.8.1 Tydeman, *_HAS_SUBNORM==0 implies what? [N 2797] (the green part only, which was skipped at the previous meeting)

Straw poll: Does WG14 want N2797 (the green text) in C23?
15/0/2. Consensus. Goes into C23.

5.8.2 Tydeman, DFP: Quantum exponent of NaN (version 2) [N 2754]

Straw poll: Does WG14 want N2754 in C23 (with "would be" changed to "is" in the footnote)?
17/0/3. Goes into C23.

5.8.3 Thomas, C23 proposal - Remove default argument promotions for _FloatN types [N 2844]

Some implementations could consider _Float32 to be a fuzzy alias of float.
DR206 states default argument promotion was only for K&R C.
The architecture can do it in the ABI still regardless of the language level.

Straw poll: Does WG14 want N2844 in C23?
14/1/2. Goes into C23.

5.8.4 Thomas, C23 proposal - Revised suggested change from N2716 [N 2847]

Straw poll: Does WG14 want N2847 in C23?
10/0/5. N2847 goes into C23.

5.8.5 Thomas, C23 proposal - Type annex tgmath.h narrowing macros with integer args [N 2849]

The _Float32x and _Decimal32x cases need to be addressed.

Action item: CFP to look into _Float32x and _Decimal32x narrowing functions for N2849 for next week.

5.8.6 Thomas, C23 proposal - 5.2.4.2.2 cleanup-update [N 2879]

Straw poll: Does WG14 want N2879 in C23?
13/0/4. Put N2879 into C23.

5.8.7 Thomas, C23 proposal - overflow and underflow definitions-update [N 2880]

One comment/concern about wording of "however for types with reduced precision..." with regards to the overflow threshold.

Straw poll: Does WG14 want N2880 in C23?
12/0/5. Put N2880 into C23.

5.8.8 Thomas, C23 proposal - Normal and subnormal classification-update [N 2881]

Straw poll: Does WG14 want N2881 in C23?
13/0/3. Put N2881 into C23.

5.8.9 Thomas, C23 proposal - Clarification for max exponent macros-update [N 2882]

Straw poll: Does WG14 want N2882 in C23?
15/0/3. Put N2882 into C23.

Alex: Incremental improvements are awesome and this has really helped with progress. Like the pipeline. Many small papers with incremental changes. Easier to understand, faster to process.

5.9 Lambdas (3 hours)

5.9.1 Gustedt, Improve type generic programming v4 [N 2890]

https://hal.inria.fr/hal-03165732
https://hal.inria.fr/hal-03165731
https://hal.inria.fr/hal-03165736
https://hal.inria.fr/hal-03553612
https://hal.inria.fr/hal-03259337
Statement expressions noted as more common implementation practice. Concerns that they are not permissive enough and hence the need for something more like lambdas.

C++ compatibility is next to trivial.

Concerns about continual wording issues with each draft of this. Indicates implementation practice needed for C. Counter is C++ has this and C++ compilers are often also C compilers. Follow on counter is that C++ experience is not relevant for something that needs C context.

5.9.2 Gustedt, Type inference for variable definitions and function returns v5 [N 2891]

Belief that an example shown ("static auto const* string3 = "a";) is valid though listed invalid. Some discussion on this.

Discussion about preventing shadowing with the given wording. Does not seem to be the case as it should only apply to typedefs.

Belief that "same type" should instead use compatible types due to no definition of that term. Disagreement from the author on this due to the non-transitive relation for it.

Belief that C code is easier to understand than C++ code and "auto" would take that away.

Belief that the implementation experience described is not valid or sufficient. Counter arguments of compilers using different syntax for similar functionality.

Straw poll: Does WG14 want something along the lines of N2923 (updates N2891, auto) in C23?
7/7/5. No clear consensus.

Straw poll: Does WG14 want something along the lines of N2923 for auto objects in C23?
10/6/1. Direction for auto type objects for C23.

5.9.3 Gustedt, Basic lambdas for C [N 2892]

Discussion on the usefulness of captures since it can be called with arguments instead. Counter is using auto to freeze it at the point of capture is useful.

Discussion on using Blocks and translating them to lambdas. Various limitations and difficulties there.

Straw poll: Does WG14 want something along the lines of N2892 with some specification in place of auto return in C23?
5/6/6. No consensus for basic lambdas with some return type invention.

Straw poll: Does WG14 want something along the lines of N2892 with lambda returns via trailing return type as per C++11 in C23?
4/7/7. No consensus for basic lambdas with trailing return type.

Discussion about needing more C implantation experience before putting it into a standard.

Straw poll: Does WG14 want something along the lines of N2892 in a future version of C?
11/4/3. Sentiment in favor of lambdas for a future C standard.

Discussion on possibly getting a study group together for this.

5.9.4 Gustedt, Options for lambdas [N 2893]

Not discussed due to 5.9.3's votes.

5.9.5 Gustedt, Type-generic lambdas v4 [N 2894]

Not discussed due to 5.9.3's votes.

5.10 Dependencies on lambdas (1 hour)

5.10.1 Uecker, Function Pointer Types for Pairing Code and Data [N 2787]

Issues with having a wide function inside a function itself.

Discussion about bringing this to C++ via the compatibility study group.

Discussion about fixing the layout vs making it a struct. Restricting implementations being a side effect of that which some are not in favor of doing.

Straw poll: Does WG14 want something along the lines of N2862 in C23?
4/6/9. No consensus to put wide pointers into C23.

Straw poll: Does WG14 want something along the lines of N2862 in a future C standard?
7/3/8. Consensus to look for wide pointers in a future C standard.

5.10.2 Gustedt, A simple defer feature for C [N 2895]

Paper does not work without lambdas.

Discussion of bringing this as a part of the study group for lambdas or vice versa (study group for defer could include lambdas). Also, discussion on TS vs study groups and how they are not mutually exclusive, or the same.

Discussion of the cost of implementing this. Basically, an array per block or function.

Discussion of type specific cleanup vs the current object specific cleanup.

5.11 Ballman, Fixes for potentially reserved identifiers [N 2762]

Discussion on what “implementation” refers to.

Discussion on the scope of this. Ex. Private symbols in the library too? Counter is this is not introducing a new problem or making an existing one worse. Disagreement on that assertion.

Noticed that the atomic functions don’t necessarily have external symbols.

Straw poll: Does WG14 want to adopt N2762 with the new footnote removed into C23?
18/0/2. N2762 (fixes for potentially reserved identifiers) without the footnote goes into C23.

5.12 Ballman, The noreturn attribute (updates N2700) [N 2764]

Editors will need the correct date value. There is a drafting note for that.

A note that you could end up with a triple underscore prefixed Noreturn. That was acceptable.

Straw poll: Does WG14 want to put N2764 into C23?
13/0/4. Put N2764 into C23.

5.13 Ballman, Literal suffixes for bit-precise integers [N 2775]

Straw poll: Does WG14 want to put N2775 into C23?
12/2/4. Put N2775 into C23.

Noticed that from MISRA's point of view, it is good to have things more explicit and match up exactly for arithmetic.

5.14 Ballman, Bit-precise I/O (replaces N2824) [N 2858]

Withdrawn.

5.15 Gilding, The `constexpr` specifier [N 2851]

Discussion on the index and array name and if it works in either order. Belief is that it works for both forms.

Concern about recursion without limits becoming an unbounded limit to determine whether it is a constant or not. Ex. The null pointer case.

Discussion on how this is different from unsequenced.

Discussion on why this is not appropriate for an attribute.

Discussion on splitting this up for objects vs functions.

Discussion on structure vs structure or union. Wider issue in the standard that is not addressed and may be inconsistent here.

Discussion on how this is not for floating-point right now.

Further concerns brought up about needing limits. Discussion on how C++ has a limit then does runtime evaluation.

Discussion about UB and how it is not necessarily decidable. Better wording needed for this and contexts where a constexpr function is not a constant/compile time evaluated function.

Concern about getting a VLA.

Concern that the floating-point TS part 5 has pragmas that can change the environment in a block that may cause it to be runtime based.

^Straw poll: Does WG14 want the constexpr feature for objects for C23?
11/4/8. WG14 wants the constexpr object features for C23.

^Straw poll: Does WG14 want the constexpr feature for member and array element access for C23?
5/6/12. No consensus to have constexpr member and array element access features for C23.

^Straw poll: Does WG14 want constexpr function calls for C23?
6/9/7. No consensus to have constexpr function calls for C23.

^Straw poll: Does WG14 want constexpr along the lines of N2917 in a future version of C?
16/1/7. WG14 wishes for something along the lines of N2917 in a future version of C.

Further discussion about diagnosing UB. Some believe it is not that difficult, others disagree.

^Straw poll: Does WG14 want to make any UB in constexpr (in an evaluated part) a constraint violation for C23?
12/4/7. WG14 wishes to make any UB in constexpr (in an evaluated part) a constraint violation for C23.

5.16 Gilding, Queryable pointer alignment [N 2852]

N2918 is the new version.

Discussion of implementation experience (Microsoft with the isaligned function) and how it is expected to use secret platform knowledge to get the result and not tracking anything from the specifier.

Discussion about freestanding and what to do there (stdlib is not in freestanding, but this could be added to 4#6).

Straw poll: Does WG14 want something along the lines of alternative 1 in N2918 in C23?
14/0/3. Direction in favor.

5.17 Gilding, Relax requirements for variadic parameter lists [N 2854]

Updated to N2919.

Concerns this would invalidate stack walking. Belief that compilers would add a built-in function to handle that.

Concerns about C++ compatibility. Decided to discuss this in the C/C++ study group meeting. Belief from some that fixing the C hole introduced by the removal of features in C23 that will hit users is more important than unlikely but potential C++ issues in this case.

Straw poll: Does WG14 want to adopt something along the lines of N2919 for C23?
13/0/4. Direction to go forward with this.

(Scribe note: This is from a revisit on second week of the meeting)

Statement that the liaison meeting did not go as well as hoped. No real objections though.

5.18 Gilding, Tail-call elimination [N 2855]

Updated to N2920.

Discussion on how the larger problem is lacking call-with-current-continuation in C. Back and forth about using attributes. Counter is the point of this feature is to not make it ignorable and lifetimes change.

Discussion on how main should be disallowed from this. Intent to fix this.

Discussion on void returns and user written code (vs machine generated) use of this feature.

Concern about no C implementation experience. Counter is that it does ship via compilers doing it as an optimization. Counter to that counter was no interface implementation experience.

Concern about the current wording not saying just autos since other resources could continue to be used. Intent to fix that.

Minor point about fixing the second line with a comment in example 1 to remove the "+ 1".

Discussion about how this would not work on some ABIs. Intent to make an exception to allow not doing this feature if not possible. Belief that the reason this was not done before was because of the ABI issues for bot C and C++.

^Straw poll: Does WG14 want something along the lines of N2920 in C23?
8/7/8. No consensus to put mandatory tail calls into C23.

5.19 Krause, @ and $ in source and execution character set [N 2701]

Specifically, not part of the basic character set.

No user complaints, but belief people expect it to work, and it does so a lot of implementation experience.

^Straw poll: Does WG14 want to add @ and $ to the source and execution character sets without requiring them to be single bytes?
10/3/7. Add @ and $ to the source and execution character sets in C23 as per N2701.

^Straw poll: Does WG14 want to add ` to the source and execution character sets without requiring it to be single byte?
6/3/10. Add ` to the source and execution character sets in C23 as per N2701.

^Straw poll: Does WG14 want to add @, $, and ` to the source and execution character sets requiring them to be single bytes?
11/4/4. Add @, $ and ` to the source and execution character sets as single bytes in C23 as per N2701.

5.20 Krause, No function declarators without prototypes [N 2841]

Discussion on how some code could break and would now require a cast to make the function pointer work legally.

Some discussion of configure scripts that use this feature so removing it would affect them. Also known cases of polymorphic function lists and function dispatch tables. This would break all that code.

^Straw poll: Does WG14 want to put N2841 in C23 (with editorial changes)?
19/3/2. Put N2841 (remove function declarators without prototypes) into C23.

5.21 Gustedt, Only reserve names of optional functions if necessary v2 [N 2860]

Concerns about platforms that do not have weak symbols or split C libraries. Also concerns about other languages using the C library now needing to know about adding in other libraries without the programmer being aware.

Discussion about the 3.6-3.8 changes. Needed due to block scope declarations.

Issues with incompatible types being used with user code with select libraries that fails when ported to places with a single C library.

Discussion on how this is not an issue in practice and no user complaints for it.

^Straw poll: Does WG14 want the changes 3.1-3.5 as given in N2860 into C23?
4/3/12. No consensus to put any part of N2860 into C23.

5.22 Gustedt, Make call_once mandatory [N 2840]

Discussion about having this for freestanding. Some for, more against.

^Straw poll: Does WG14 want N2840 (call_once) into C23?
10/0/8. Put N2840 (call_once) into C23.

5.23 Keywords and related topics (2 hours)

5.23.1 Gustedt, Revise spelling of keywords v6 [N 2884]

Split up the paper to discuss thread_local separately due to N2815 concerns.

Discussion about making empty headers obsolete. General sentiment to not have them removed since people want to write code that works for various standard versions.

Belief from some that a lot of code will break. Especially for the true/false upcoming paper.

^Straw poll: Does WG14 want to make stdalign.h obsolescent in C23?
1/7/12. No consensus to make stdalign.h obsolete in C23.

^Straw poll: Does WG14 want something along the lines of Changes 1-5, 6 (removing the word "obsolescent"), 8 and 9 from N2884 into C23?
14/2/2. Clear direction for keywords to be added for the _Capital forms as in N2884.

(Scribe note: Back to this paper after discussing N2922 to shift to discuss thread_local)

Summary of the C++ liaison meeting discussing the thread_local change was given. Essentially no consensus between the committees but C++ did want _Thread_local to stay. In the follow up discussion, C++ was looking to another solution like ‘extern “C”’

(Scribe note: Chat showed the vote as 4/1/5 for WG21 wanting to keep the header to get thread_local).

Straw poll: Does WG14 want something along the lines of N2884 changes 10 and 11 (thread_local as a keyword) for C23?
9/4/6. Generally in favor of making thread_local a keyword.

5.23.2 Gustedt, Make false and true first-class language features v6 [N 2885]

Updated to N2922.

Discussion about whether bool could be wider like unsigned int. Speculation DSP’s may have that.

Same issues with regards to making headers obsolete were brought up.

Issues with changing non-zero value for true were brought up but countered with the authors assertion this doesn’t remove the possibility, just leaves it as UB and when stored it has to be 1.

^Straw poll: Does WG14 want to mark <stdbool.h> as an obsolescent header as in N2922 for C23?
5/7/6. No consensus to make stdbool.h obsolescent as per N2922.

^Straw poll: Does WG14 want something along the lines of N2922 ('true' and 'false' as keywords) with changes 1-3,5-14,16 in C23 while removing the obsolescence of stdbool.h?
14/3/2. Sentiment to have changes 1-3,5-14,16 in C23 while removing the obsolescence of stdbool.h in N2922 in C23.

5.23.3 Gustedt, Add annotations for unreachable control flow v2 [N 2826] (1 hour)

Discussion on the naming of the feature and concerns it may invade user namespace. Author responded with no real collisions when searching common code bases.

Large discussion on C++ and their proposals of unreachable and assume. Some belief they are not competing, some statements that assume is a superset of unreachable. Proposal was for a targeted and easy to explain feature.

Discussion about making it a regular function. Library vs language (compiler) views and purposes. Also discussion about making the symbol not specified if it actually exists to allow both cases. Concerns about trying to take the address of it were brought up in that case. Some want that, some don’t.

Discussion about explicitly saying UB vs implicit UB (via violating a "shall"). Some people uneasy about bringing in (more) UB. Author says this is to make unintentional UB explicit so not really adding more, and it is to help. Statement that people who don't want this extra explicit UB don't have to use it.

Discussion on the scope and limits of the 'unreachable'. Does it apply up the flow chain to a conditional? Author mentioned there is recommended practice to put it at the top of the path. Statement that 'assume' would not have this problem. Author agreed since the expression would be in the right place.

Discussion about allowing this for freestanding as well.

^Straw poll: Does WG14 prefer the syntax variant over the macro variant of unreachable as in N2826 in C23?
3/12/5. WG14 does not prefer the syntax variant for unreachable as in N2826.

^Straw poll: Does WG14 prefer header stddef.h for unreachable as in N2826 in C23?
11/2/7. WG14 prefers stddef.h for unreachable (N2826) in C23.

^Straw poll: Does WG14 want to integrate the macro unreachable feature as described by changes 5-6 in N2826 in stddef.h into C23?
13/2/6. Put the macro version of unreachable as per N2826 in stddef.h in C23.

5.24 Gustedt, Add new optional time bases v4 [N 2647]

Plan on updating the text "execution platform" to "execution environment".

Discussion about effects of external events. Statement that POSIX says external events can modify the monotonic clock. Author states that was not their understanding. Further discussion about using the POSIX specification which uses different terms (like "real time") than what is in the C standard and references "clock_set_time" which does not exist for C.

^Straw poll: Does WG14 want something along the lines of TIME_MONOTONIC as in N2647 in C23?
11/0/11. WG14 wants something like TIME_MONOTONIC as in N2647 in C23.

^Straw poll: Does WG14 want something along the lines of TIME{_THREAD}_ACTIVE as in N2647 in C23?
11/0/10. WG14 wants something like TIME{_THREAD}_ACTIVE as in N2647 in C23.

5.25 Gustedt, Properly define blocks as part of the grammar v2 [N 2818]

Large discussion on how this draft of the paper refers to a previous draft of the standard and not the latest one. Resolved by having homework to update the paper to refer to the latest C standard draft.

Discussion about labeled-statement being special or intended to be replaced by one of the blocks. Intent and text in the paper keeps it as is.

Reflector discussion on what happens with unlabeled-statement.

^Straw poll: Does WG14 want something along the lines of N2818 (rebased onto the current working draft) into C23?
14/0/7. WG14 wants something like N2818 (blocks definition in the grammar) rebased on the latest draft of C23 for C23.

(Scribe note: This part is post homework completion)

^Straw poll: Does WG14 want N2818 (rebased onto the current working draft) into C23?
18/0/1. Put N2818 (rebased onto the current working draft) into C23.

5.26 Gustedt, Disambiguate the storage class of some compound literals [N 2819]

^Straw poll: Does WG14 want N2819 into C23?
17/0/4. Put N2819 (lifetime of compound literals) into C23.

5.27 Gustedt, Unsequenced functions v4 [N 2887] (1 hour)

Discussion on intent of compilers to check that these properties actually exist in the code. There is recommended practice to diagnose incorrectly applied properties. Belief a static analyzer can find instances of violation. Discussion of how the main use case is where it can't be checked by compilers (otherwise they would not need the property attributes).

Concern about applying this to function types. Discussion back and forth on that. WG14 wanted consideration for it last time, but concerns no implementations do type attributes for these and that it's hard for it in Clang. Discussion about only making the type attribute idea only apply to these attributes (properties). Discussion on how GCC's attribute placements are prior art and the problems that come from them, along with the lack of documentation for them. Discussion on separating out the type support and how it is hard to do given how integrated it is.

Discussion on how these properties need access to all the source (all called functions down the chain) to be verified.

Discussion on how putting the property on the declaration, it applies to those uses. On the definition and it applies to all calls to it.

Discussion on how adding these in will cause it to be banned in some environments like MISRA. Counter is that these attributes are optional, so they don't need to be used in those environments. Should not hold up use in other environments.

Issues about headers getting out of sync from implementations causing problems. Comment that GCC has analyzers that can detect the absence of these properties but not the converse.

^Straw poll: Does WG14 want other syntactic positions of the attributes in N2887 beyond the current standard type attribute's locations in C23?

1/14/7. No desire to have attributes in N2887 (unsequenced functions) in positions beyond where type attributes are now in the C standard.

^Straw poll: Does WG14 want composition of attributes as in N2887 (unsequenced functions) as a general rule?
12/4/5. WG14 wants composition of attributes as in N2887 (unsequenced functions) as a generally applicable rule for attributes.

Discussion on bringing this to the C++ liaison group. Agreement to do so.

^Straw poll: Does WG14 want something along the lines of [[unsequenced]] as in N2887 (unsequenced functions) in C23?
10/2/10. WG14 wants something along the lines of [[unsequenced]] as in N2887 (unsequenced functions) in C23.

^Straw poll: Does WG14 want something along the lines of [[reproducible]] as in N2887 (unsequenced functions) in C23?
10/3/9. WG14 wants something along the lines of [[reproducible]] as in N2887 (unsequenced functions) in C23.

^Straw poll: Does WG14 want something along the lines of the [[stateless, effectless, independent, idempotent]] attributes as in N2887 (unsequenced functions) in C23?
5/5/13. No clear consensus to have [[stateless, effectless, independent, idempotent]] attributes as in N2887 (unsequenced functions) in C23.

5.28 Honermann, char8_t: A type for UTF-8 characters and strings (Revision 1) [N 2653]

Concerns about warnings being emitted when using string and character functions with this due to type issues. Counter is that it depends on whether char is signed or unsigned. There is a tradeoff for portability. In practice people who turn on -Werror (or the equivalent) then do a number of exceptions to those to not diagnose certain cases.

Discussion of how the examples are not meant to be gold standard code. Just examples.

Discussion on the need for efficient functions (since the restartable ones are not).

^Straw poll: Does WG14 N2653 (UTF-8) in C23?
18/0/2. Put N2653 (char8_t) into C23.

Thursday, 17 February

5.29 Sommerlad, Make assert() macro user friendly for C and C++ v2 [N 2829]

Discussion of how the current wording says “scalar expression” so this could be seen as a bug fix.

Concerns about making this a special macro definition vs all the other macros in the standard. Makes it inconsistent. Counters included incremental improvement or that this was the only macro that is a problem (author mentioned specifically for C++).

^Straw poll: Put N2829 (assert macro change) into C23?
13/3/4. Put N2829 (assert macro change) into C23.

5.30 Meneide, Modern Bit Utilities r0 [N 2827]

Updated to N2903.

In section 3.5, the #if inconsistent parenthesis will be fixed.

Discussion on CHAR_BIT == 8 being the only case allowed for these functions. Arguments both ways. DSP's mentioned for cases where it is not true. Author willing to go whichever way the committee wants.

Long discussion on the rotate functions. Why have both left and right? Why not just one with a negative argument to indicate the other direction? Some want three sets: left, right, and direction given by sign. Some want only one set (direction by sign). C++ preference discussed as well.

Discussion on the size parameter being last vs first. Charter principle given as the reason. Back and forth on this topic. Some want size first, others want size after.

Discussion on type generic macros (as in a reflector message from Joseph) used here. Rotate amount type and base item type being rotated can be a different operation. Author stated the type generic macros were added for extended integer types to without concrete function names. Issues with using the results of the type generic functions in something causing UB (Ex. printf).

Discussion on making the count parameter be an unsigned type.

Discussion on using size_t for the return type for everything including bit-precise integer types.

(Scribe note: Continued on Friday)

Discussion about differences between C++. Author will add rationale stating C wanted unsigned types for counts while C++ wants int's.

Noticed that the introduction parts there is talk of the first leading zero or one functions, but not in the normative wording. Author will add it.

^Straw poll: Does WG14 want the memreverse and endian load/store functions to only be required if CHAR_BIT == 8 similar to N2903?
6/5/8. No clear direction for memreverse and endian load/store to have CHAR_BIT = 8.

Discussion about memreverse working on 8 bits vs (or in addition to) plain memreverse. Implementation practice was 8 bits only so that was what was done. Names reflect that allowing extensions to do something else with different names.

^Straw poll: Does WG14 want new signed-count rotate functions in addition to what is in N2903?
8/6/6. Weak sentiment for new signed-count rotate functions (bit utilities) into C23.

Discussion on how 2's complement does the right things for rotate. Author will just do the signed count rotate functions. Also, the author will check with Jens in C++ to see why C++ didn't know this or choose this.

^Straw poll: Does WG14 want something along the lines of N2903 (bit utilities) into C23?
19/2/1. WG14 wants something along the lines of N2903 (bit utilities) into C23.

Discussion about discussing the prefix names at some point and more generally, naming new identifiers in the standard.

^Action item: JeanHeyd: I can write a policy paper or something for naming of new identifiers in the standard.

Discussion about the floating-point names not being reserved. Counter is that they follow established math.h function name schemes and there is no issue.

5.31 Meneide, Unicode Sequences More Than 21 Bits are a Constraint Violation r0 [N 2828]

^Straw poll: Put N2828 into C23?
18/0/2. Put N2828 (Unicode sequence > 21 bits) into C23.

5.32 Meneide, Not-So-Magic: typeof(), revision 3 [N 2724]

Discussion about another place that needed a change (VLA->VM).

Discussion about how the implementation(s) that did not keep all qualifiers considered it a bug to do so.

(Scribe note: Revisited paper after homework)
Updated to N2927.

Discussion on how typeof is similar to sizeof and alignof with regards to evaluation.

2022/02/03:
Straw poll: Does WG14 want to put N2927 into C23?
10/0/6. Put N2927 into C23.

5.33 Meneide, Preprocessor embed, revision 4 [N 2725]

Updated to N2898.

Author mentioned they got a physical letter from a company that is Japanese and German based asking me to put a blog post or something for this feature.

Discussion about using attributes for the trailing limits (parameters). Countered by the fact they cannot be ignorable. The term attribute was used as the grammar term.

Discussion about the limit parameter needing to be non-negative. Also, addition of trailing underscores to avoid macro expansion.

Discussion about implementation experience. Concerns there are no shipping compilers with this.

Discussion sidetracked into the general issue about not following the charter as to having features being added to C23 that are not in shipping compilers. This will be discussed in a future meeting. For this feature, discussion about other vehicles like TS's.

Discussion about mixing binary with text. Arguments against included other tools are made for that like objdump, while arguments for are requests for something like this, including C++.

Discussion about doing the core embed first then add in the parameters later. Countered by user experience asking for those parameters, otherwise the feature causes errors.

Discussion about how compilers cannot do this right and need the feature.

Discussion about how compiler implementers are less open to experimentation now.

^Straw poll: Does WG14 want the embed parameter specification as shown in N2898?
12/2/8. WG14 wants the embed parameter specification as shown in N2898.

5.34 Meneide, Consistent, Warningless, and Intuitive Initialization with {} [N 2796]

Updated to N2900.

Discussion about future improvements as per some reflector messages. Specifically, 20045 (12 Aug 2021) and 20354 (3 Sep 2021).

Straw poll: Put N2900 ({} initialization) without the optional changes into C23?
17/0/3. Put N2900 ({} initialization) without the optional change into C23.

^Straw poll: Put something along the lines of N2900 optional change 0 into C23?
0/10/10. No consensus to put N2900 optional change 0 into C23.

5.35 Uecker, Consistency of Parameters Declared as Arrays (updates N2779) [N 2906]

Question if the following code (Scribe note: Copied from the chat) is a constraint violation. Answer was no since the types are not compatible anymore.

```
_Generic(foo,
   int (*)(double [3]) : 1,
   int (*)(double [4]) : 2,
   default : 3
```

```
    );
```

Discussion about having multiple ways of talking about things that are generally the same can be confusing and error prone.

Discussion about implementation experience being needed. Author mentioned it was implemented in a local branch of GCC. Others mentioned MISRA rule 17.5 is related.

Discussion about how strengthening the type system will cause problems, but those problems are good to be detected. Counter was that changing the type system should require new syntax and risks with reinterpreting existing syntax. Counter to that was that this feature is to help fix existing code.

Concern that change 2a will allow VLA typedef redeclaration now with different sizes. Author did not intend that and will look into it.

^Straw poll: Does WG14 want something along the lines of N2906 in C23?
13/1/5. WG14 wants something along the lines of N2906 (consistency of arrays as parameters) in C23.

^Straw poll: Does WG14 want something along the lines of N2906's typedef redeclaration with type compatibility in C23?
12/1/7. WG14 wants something along the lines of N2906 (consistency of arrays as parameters) typedef redeclaration with type compatibility in C23.

5.36 Uecker, Forward Declaration of Parameters [N 2780]

Discussion about this being good for fixing existing interfaces, but not for new ones.

Discussion about this having existing practice being good, though not very prevalent being concerning. Also issues about it being broken with no complaints in the past in GCC.

Discussion about using something like attributes and C++ compatibility. Counter for attributes was that it doesn't work for multidimensional VM types. C++ also doesn't have VLA's so already an issue there.

Discussion about how this fills a hole we introduced in C23. Most common complaint heard by a member was taking away the ordering due to the K&R removal and having a fix for it would really help the community. Without this, they will keep using K&R.

Issues with wording to say something about composite types and what attributes appertain to. Same for something needed to be said about the size expression being evaluated.

Discussion about changing this so later parameters could be referred to in previous parameters. Issues there with mutual dependencies and different scopes. Also, no implementation experience. It can also break existing code. Index of parameters was also discussed as a way of referencing them.

^Straw poll: Put something along the lines of N2780 into C23?
8/7/6. No consensus to put something along the lines of N2780 into C23.

Discussion of the C++ feature in progress regarding naming arguments (call side). Counter is that feature does not solve this issue.

^Straw poll: Does WG14 want something to solve the forward declaration problem?
13/4/5. Generally in favor of some solution to the forward declaration problem.

^Straw poll: Does WG14 want something like forward parsing to solve the forward declaration problem?
7/5/10.

5.37 Steenberg, break [N 2859]

Discussion about numbered breaks being problematic. Discussion about labelled loops being used in other languages and the applicability to C. Some support for that instead. Counter was position of the label after the user code has the same issue as right now so might as well use goto for that.

Discussion for and against that this is clearer than goto.

Discussion about the order of the stack for the statement. No preference for either order by the author. Concerns users will want one order or another. Special case of continue being present means there will be a syntax error if it is done in any other order.

Various members support the fact that this is a problem in real code.

Straw poll: Does WG14 want a new way to break out of multiple loops in a future C standard?
10/6/7. Generally in favor of having a new way to break out of multiple loops.

5.38 Ojeda, #once (updates n2742) [N 2896]

Discussion about how there is the ability for this to mean nothing, both for and against.

Discussion about no support in C++ right now causing preprocessor divergence.

Request to not require the directive to be first. Agreement by the author that the requirement could be relaxed. Discussion on issues with this and how implementations would need to handle cases where it is not first, especially if the preprocessor state is changed.

Discussion about this being something new while the #define mechanism is known by everyone.

Discussion about having this treated as #error being odd. Preference to keep it as a constraint.

Discussion about current methods to check if files are the same all have false negatives. These problems cannot be fixed by users in their code. Putting it into the standard makes the problem worse as it is now blessed. Counter is that the identifier form is good and covers that case.

Discussion about how without in real world systems, without the ability to #ifdef the #once, it won’t be used since there are always some compilers that support it first while others don’t, and common code needs to deal with it.

Straw poll: Does WG14 want something along the lines of #once (any form) like N2896 in C23?
9/8/6. No consensus to put #once (N2896) in C23.

6. Clarification Requests
The previous queue of clarification requests has been processed.

7. Other Business
The following papers will be deferred to future meetings unless there is time available at this meeting.
7.1 Svoboda, Towards Supplemental Integer Safety [N 2792]
7.2 Douglas, C2x fopen("x") and fopen("a") v2 [N 2857]
7.3 Köppe, Comma omission and comma deletion [N 2856]
7.4 Bachmann, Make pointer type casting useful without negatively impacting performance - updates n2484 [N 2658]

7.5 Gustedt, Remove `ATOMIC_VAR_INIT` v2 [N 2886]
Started discussion but stopped quickly as the author was not ready to discuss the paper yet.

7.6 Gustedt, Require exact-width integer type interfaces v2 [N 2888]
7.7 Gustedt, Pointers and integer types [N 2889]
7.8 Ojeda, memset_explicit (updates n2682) [N 2897]
7.9 Uecker, Safer Flexible Array Members [N 2905]
7.10 Uecker, Wording Change for Variably-Modified Types [N 2907]
7.11 Uecker, C23 Atomics: Proposed Wording Changes (updates N2771) [N 2909]

Papers not intended for C23

7.12 Steenberg, Redefining Undefined Behavior [N 2769]
7.13 Gilding, The `void`-_which-binds_: typesafe parametric polymorphism [N 2853]

7.14: How to schedule after C23 (Item added to the agenda during the meeting)
Question: Should we adopt a train schedule like WG21?

Discussion about how we need to have something in flight at least every 4 years to avoid disbandment. Frequency of the release discussed, with 3 or 4 years being most commonly suggested. Matching C++'s schedule discussed. Experience there is they still get a lot of papers come in near the ship date deadlines. Some users like rapid releases, others don't. Issues of instability.

Discussion between bug fix releases and feature releases. Alternate or not? Belief that having a bug fix release constrains proposals. Arguments both for and against. Belief maintenance is needed and having it scheduled ensures it happens. Concern with maintenance releases include being less productive and proposals dropping off.

Discussion about inventions in C23 that were not implemented in C. Argument that this causes a snowball effect of features being built on top of feature without any implementation experience causing serious bugs in the standard and for users.

(Scribe note: Picked up the discussion again on another day)

Belief people like a fixed schedule but needs more discipline. Lack of polish is a risk. Also letting things in before they are ready is a risk. Need to ensure features get in when they are ready and not based on the standard release.

Discussion about how people are still gradually adopting C99, implementations ignoring the standard (Ex. atomics), people wanting new features are C++ users already and not C users.

Further support for fixes and completion rather than new features. Discussion about what our mission should be. Concerns if we get too close to C++, why is C there? Belief it is the small semantic gap between the program and the machine.

Discussion on how it is not practical to limit what proposals come in.

(Scribe note: Picked up the discussion yet again on yet another day)

Discussion about not making an absolute focus for each release (bug fix vs new features). Just have it as a focus (but not exclusive of everything else). Suggestion to have just bug fixes as prioritized but features allowed for all releases. Mention that C++ does do the priority plans without hard rules on what gets into a release.

Further discussion about existing implementation support being needed.

Discussion about how alternate releases allow ironing out issues between releases before things pile up and how that would not be as big of an issue if the charter was followed regarding implementation experience.

Discussion about the size of the feature/specification not being important. It is how it interacts with the rest of the language.

Discussion about what the bug fixes are mattering. For example, the memory model.

7.15: N2931 (homework from CFP) updating N2849

^Straw poll: Put N2931 into C23?
15/0/4. Put N2931 (type generic narrowing macros CFP) into C23

7.16: N2934 revised spelling of keywords (homework) updating N2850

^Straw poll: Put N2934 into C23?
16/1/3. Put N2934 (revised spelling of keywords) into C23.

7.17: N2935 make false and true language features (homework)

^Straw poll: Put N2935 into C23?
15/2/3. Put N2935 (make false and true language features) into C23.

8. Recommendations and Decisions reached

8.1 Review of Decisions Reached

^Action: Editor: Put N2833 (new time functions) alternative 1 (with the typo of mktime->timegm in the returns section fixed) into C23.
^Action: Editor: Put N2836 (Unicode identifier syntax) in C23.
^Action: Editor: Put N2810 (calloc wraparound) into C23 (modulo editorial corrections).
^Action: Editor: Put N2797 (HAS_SUBNORM), the green text, in C23.
^Action: Editor: Put N2754 (quantum exponent of NaN) into C23 (with "would be" changed to "is" in the footnote).
^Action: Editor: Put N2844 (remove default argument promotion for _FloatN) into C23.
^Action: Editor: Put N2847 (numerically equivalent for quantum exponents) into C23.
^Action: Editor: Put N2879 (floating point model categorization) into C23.
^Action: Editor: Put N2880 (overflow/underflow update for double-double) into C23.
^Action: Editor: Put N2881 (normal and subnormal classification) into C23.
^Action: Editor: Put N2882 (max exponent macro update for double-double) into C23.
^Action: Editor: Put N2762 (fixes for potentially reserved identifiers) without the footnote into C23.
^Action: Editor: Put N2764 (noreturn attribute) into C23.
^Action: Editor: Put N2775 (bit-precise literal suffixes) into C23.
^Action: Editor: Put N2927 (typeof) into C23.
^Action: Editor: Add @, $ and ` to the source and execution character sets as single bytes in C23 as per N2701.
^Action: Editor: Put N2841 (remove function declarators without prototypes) into C23.
^Action: Editor: Put N2840 (call_once) into C23.
^Action: Editor: Put the macro version of unreachable as per N2826 in stddef.h into C23.
^Action: Editor: Put N2818 (blocks as grammar, rebased onto the current working draft) into C23.
^Action: Editor: Put N2819 (lifetime of compound literals) into C23.
^Action: Editor: Put N2653 (char8_t) into C23.
^Action: Editor: Put N2829 (assert macro change) into C23.

^Action: Editor: Put N2828 (Unicode sequence > 21 bits) into C23.
^Action: Editor: Put N2900 ({} initialization) without the optional change into C23.
^Action: Editor: Put N2931 (type generic narrowing macros CFP) into C23
^Action: Editor: Put N2934 (revised spelling of keywords) into C23.
^Action: Editor: Put N2935 (make false and true language features) into C23.

^Action: WG14: Decide what to do about the Strasbourg meeting in the May or July 2022 meeting.

WG14 wishes to see TS6010 (memory object model) working draft (N2676 or something similar) in some future version of the standard.
WG14 sees TS6010 (memory object model) working draft (N2676 or something similar) as important for C23.
WG14 is willing to move TS6010 (memory object model) to DTS ballot as it stands now if requested.
WG14 wishes to address thread specific runtime constraint handling in Annex K.
WG14 wishes for something along the lines of N2923 for auto objects in C23.
WG14 wants lambdas for a future C standard.
WG14 wants wide pointers in a future C standard.
WG14 wishes for something along the lines of alternative 1 in N2918 (query pointer alignment) in C23.
WG14 wishes for something along the lines of N2919 (relax varargs leading argument requirement) for C23.
WG14 wishes for a new way to break out of multiple loops (N2859) in a future C standard.
WG14 wishes for the constexpr object features for C23.
WG14 wishes for something along the lines of N2917 (constexpr) in a future version of C.
WG14 wishes to make any UB in constexpr (in an evaluated part) a constraint violation for C23.
WG14 prefers stddef.h for unreachable (N2826) in C23.
WG14 wants something like TIME_MONOTONIC as in N2647 in C23.
WG14 wants something like TIME{_THREAD}_ACTIVE as in N2647 in C23.
WG14 wants composition of attributes as in N2887 (unsequenced functions) as a generally applicable rule for attributes.
WG14 wants something along the lines of [[unsequenced]] as in N2887 (unsequenced functions) in C23.
WG14 wants something along the lines of [[reproducible]] as in N2887 (unsequenced functions) in C23.
WG14 is generally in favor of some solution to the forward declaration problem.
WG14 has weak sentiment to have forward parsing to solve the forward declaration problem. (Scribe's interpretation of the vote)
WG14 wants something along the lines of N2906 (consistency of arrays as parameters) in C23.
WG14 wants something along the lines of N2906 (consistency of arrays as parameters) typedef redeclaration with type compatibility in C23.
WG14 has weak sentiment for new signed-count rotate functions (bit utilities) into C23.
WG14 wants something along the lines of N2903 (bit utilities) into C23.
WG14 wants the embed parameter specification as shown in N2898.

No consensus to adopt N2834 (deprecate %n) with the 'obsolescent' alternative into C23 with some changes.
No consensus to put proposed wording 2 given in N2809 (Annex K repairs) into C23.
No consensus to put proposed wording 1 given in N2809 (Annex K repairs) into C23.
No consensus to put something along the lines of N2923 (updates N2891, auto) into C23.
No consensus for basic lambdas with some return type invention.
No consensus for basic lambdas with trailing return type.
No consensus to put wide pointers (N2862) into C23.
No consensus to put #once (N2896) in C23.
No consensus to have constexpr member and array element access features for C23.
No consensus to have constexpr function calls for C23.
No consensus to put mandatory tail calls (N2920) into C23.
No consensus reserve names of optional functions (N2860) into C23.
No consensus to make stdalign.h obsolete (N2884) in C23.
No consensus to make stdbool.h obsolescent as per N2922.
No consensus to have attributes in N2887 (unsequenced functions) in positions beyond where type attributes are now in the C standard.
No clear consensus to have [[stateless, effectless, independent, idempotent]] attributes as in N2887 (unsequenced functions) in C23.
No consensus for the syntax variant for unreachable as in N2826.
No consensus to initialize whole union then largest member for {} (N2900) into C23.
No consensus to put something along the lines of forward declaration of parameters (N2780) into C23.
No consensus for memreverse and endian load/store to have CHAR_BIT = 8.

8.2 Review of Action Items

Carry over action items:
Aaron Ballman: Add N2761 to the papers-of-interest list. - Open - Stricken
Ballman: That is not a paper we have voted into C23. No content for the paper. The meeting minutes don't list it.
Keaton: That paper is in error. There probably was a paper. I need to look for it.

New action items:
Keaton: Look into meeting in May 16-20 and July 18-22, 2022, virtually. - Done
CFP: Look into _Float32x and _Decimal32x narrowing functions for N2849 for next week. - Done
Jens: Update N2884 for this meeting. - Done
Jens: Update N2885 for this meeting. - Done
Jens: Update N2818 to the latest C standard draft for next meeting. - Done
JeanHeyd: Write a policy paper for naming of newly proposed identifiers for the standard.
Keaton: Look into what paper to add to the papers-of-interest list.
Keaton: Add C charter discussion for the next meeting.

9. PL22.11 Business (Friday, 18 February)

9.1 Approval of Previous PL22.11 Minutes [pl22.11-2021-00012] (PL22.11 motion)

Motion: Aaron Ballman (Intel), Barry Hedquist (Perennial)

9.2 Identification of PL22.11 Voting Members

9.2.1 Members Attaining initial Voting Rights at this Meeting

None.

9.2.2 Members who regained voting rights

None.

9.3 PL22.11 Voting Members in Jeopardy

9.3.1 Members in jeopardy due to failure to vote on Letter Ballots

Cisco (since 2020).

9.3.2 Members in jeopardy due to failure to attend Meetings

9.3.2.1 Members in jeopardy who retained voting rights by attending this meeting

None.

9.3.2.2 Members in jeopardy who lost voting rights for failure to attend this meeting

None.

9.4 PL22.11 Non-voting Members

9.4.1 Prospective PL22.11 Members Attending their First Meeting

None.

9.4.2 Advisory members who are attending this meeting

None.

9.5 Other Business

9.5.1 PL22.11 Meeting Votes

PL22.11 Action Item #0050

This action item is a request for members to review the committee scope. We are asking all the committees to conduct this review to ensure the accuracy and completeness of the scope statements. This review may lead to edits, refinements or improvements to the existing scope of work. As a reminder, the scope of the committee is publicly available from www.INCITS.org and also included in the marketing slide decks prepared last year. It is important that we have the best descriptions available when engaging interested parties to highlight and promote the important work that is done within the committees.

The current scope for INCITS/PL22.11 can be reviewed from https://www.incits.org/committees/pl22.11.

That scope says:

Task Group PL22.11 (formerly J11), Programming Language C is responsible for the technical development of the standard for C programming language. The goal, which J11 believes had been successfully achieved, is to make it possible for C program to be portable among operating systems and across a wide variety of computers. Extensive work has been done to make this standard acceptable in both the National and International arenas.

PL22.16, C++ has the following item in its scope that says

"High level of compatibility with the ISO C standard and suitability for the International community are two associated goals established by PL22.16 that will help to extend the useful life of this standard and increase the audience of its users."

The proposed addition to the PL22.11 Scope is to similar words to the C Committee's scope without turning C into C++:

MOTION: At the end of PL22.11 existing scope, add:

An appropriate level of compatibility with the ISO C++ standard and suitability for the International community are two associated goals established by PL22.11 that will help to extend the useful life of this standard and increase the audience of its users.

QUESTION: Are you in favor of adding the above scope to the existing PL22.11 statement of its scope.

Discussion about "appropriate" not being a good word. Arguments for include C having the choice what to support, and mentioning "high level" implies we are trying to be C++. Arguments against include the fact that C++ says "high level" and we should reciprocate.

Amendment: "A high level" as the first three words to the motion.
Moved by: David Keaton (Keaton Consulting), Barry Hedquist (Perennial)
No objections.
Vote (one per member organization):
For: 9
Against: 2
Abstain: 0
Motion passes.

10. Thanks to Host

10.1 Thanks and apologies to Intel, the originally intended host

10.2 Thanks to ISO for supplying Zoom capabilities

11. Adjournment (PL22.11 motion)

Motion: Robert Seacord (NCC), Clive Pygott (LDRA)
Meeting adjourned.