Draft Minutes for 30 March - 03 April, 2020

MEETING OF ISO/IEC JTC 1/SC 22/WG 14 AND INCITS PL22.11
WG 14 / N 2519 (FKA N 2516)

Dates and Times:

30	March,	2020	01:30 – 4:30
31	March,	2020	01:30 – 4:30
1	April,	2020	01:30 – 4:30
2	April,	2020	01:30 – 4:30
3	April,	2020	01:30 – 4:30

Meeting Location
Teleconference

1. Opening Activities

1.1 Opening Comments (Keaton)

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Notes
Aaron Ballman	GrammaTech	USA	WG21 liaison
Andrew Banks	LDRA Ltd.	UK	MISRA liaison
Rajan Bhakta	IBM	Canada	PL22.11 Chair
Roberto Bagnara	University of Parma	Italy	Italy NB
Melanie Blower	Intel	USA	
Alex Gilding	Perforce / Programming Research Ltd.	USA	new voting
Lars Gullik Bjonnes	Cisco	USA	
Barry Hedquist	Perennial	USA	PL22.11 IR

Name	Organization	NB	Notes
Tommy Hoffner	Intel	USA	
Larry Jones	Siemens PLM Software	USA	
David Keaton	Keaton Consulting	USA	Convener
Will Klieber	CERT/SEI/CMU	USA	
Philipp Krause	Albert-Ludwigs-Universität	Germany	Host
Maged Michael	Facebook	USA	new voting
Kayvan Memarian	University of Cambridge	UK	
JeanHeyd Meneide	Self	Netherlands	
Joseph Myers	CodeSourcery / Siemens	UK	
Niall Douglas	Ireland ned Productions Ltd	Ireland	
Clive Pygott	LDRA Ltd.	UK	WG23 liaison
Robert Seacord	NCC Group	USA	new voting
Martin Sebor	IBM	USA	
Peter Sewell	University of Cambridge	UK	Memory Model SG
David Svoboda	CERT/SEI/CMU	USA	Recording Secretary
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Martin Uecker	University of Goettingen	Germany	
Florian Weimer	IBM	USA	
Freek Wiedijk	Plum Hall	USA	
Michael Wong	IBM	Canada	WG21 liaison

1.3 Procedures for this Meeting (Keaton)

Keaton: We do straw polls in WG14, rather than formal votes. The representatives to ISO WG's are individual experts, and are
expected to vote for the technically best decision.

David Keaton is the Meeting Chair.
David Svoboda is the Recording Secretary.

1.4 JTC 1 Required Reading

- 1.4.1 ISO Code of Conduct
- 1.4.2 IEC Code of Conduct
- 1.4.3 Key points

1.5 Tabled: Approval of Previous Minutes [N 2451] (PL22.11 motion, WG 14 motion) (Ballman, Seacord)

Keaton: Is there a PL22.11 motion to approve minutes from the last meeting?
Blower: There are some typos to be fixed.
Hedquist: Are the changes editorial?
Blower: Mostly.
Hedquist: There should be two publications: one for WG14 and one for PL22.11.
Keaton: We will table this vote to the end of the meeting (section 9). Addressed Friday.

1.6 Review of Action Items and Resolutions

- Gustedt will investigate document control system alternatives, and will talk to Keaton and Plakosh about the possibilities.
  - OBE
- Keaton: please change the convener's report to explicitly talk about the floating liaison to C++.
  - Done in the template for the next report
- Action item for Keaton to find out what happened to N2008.
  - Open
- Keaton to put a note into SD 3 about the resolution of N 2324 macro invocations that span files.
  - Open
- cf N2390 Blower will investigate whether there is language in the standard stating that simple assignment to atomic variables is indeed an atomic operation, and write a paper if necessary.
  - Done
- Gustedt will update wording for N2391.
  - Done, N2461
- Keaton will discuss N2419 with the author.
  - Done
- CFP will reword N2400 as a recipe.

    - Done, N2490
- Seacord will write a new paper for N2438 on realloc with size 0.
    - Done, N2464
- Pygott will investigate supplying a bug tracking system.
    - Partially done, open

1.7 Approval of Agenda [N 2504] (PL22.11 motion, WG 14 motion) (Bhakta, Hedquist, approved)

1.8 Identify National Bodies Sending Experts

Canada, Germany, Italy, Netherlands, UK, USA

1.9 INCITS Antitrust Guidelines and Patent Policy

1.10 INCITS official designated member/alternate information

2. Reports on Liaison Activities

2.1 SC 22

2.2 PL22.11/WG 14

- 2.2.1 Keaton, Document system

  We will be introducing C-numbered documents after this meeting (rather than N-numbered documents), each with a revision number. Minutes & other meta-information will continue to use N-numbers. C-numbers will be technical proposals that are revised over time. The N- vs. C- numbering system is evolving. For now, you will still get N- or C- numbers through Dan Plakosh.

2.3 Ballman, PL22.16/WG 21

WG21 finished their work on C++20, modulo balloting.
Their next meeting will be in November in New York, and they will begin working on C++23.
They have a new direction committee, which wants a better working relation with WG14.
They also want to hear WG14 proposals; they are willing to review them.

Keaton: We would welcome review, they would provide lots of different expertise.
Bhakta: What process can we use to pass papers to WG21 for review?
Ballman: We can discuss that. WG21 has no expectations yet. WG21 could co-locate with WG14 during WG21 meetings.

2.4 Keaton, PL22

Chris Tandy has passed away.
Bryce Lelbach (from Nvidia) has been nominated to chair.

2.5 Pygott, WG 23

Tabled: Clive Pygott is not here yet.

2.6 MISRA C

- 2.6.1 Banks, MISRA C Liaison Report [N 2445]

  MISRA C 2012 5th edition has been published.
  MISRA C Compliance 2020 has been published.
  We are monitoring TS 17961 and ISO 24772.

2.7 Other Liaison Activities

3. Reports from Study Groups

3.1 C Floating Point activity report

Bhakta:
We are continuing to work on bindings to IEEE-2019.
We are working on creating updated TS's for the ones voted in (part 3) for inclusion into the working C2X draft.
We are continuing to work on and open DR's against the TS's and integrated C2X draft.

3.2 C Safety and Security Rules Study Group

Keaton: Chair Charles Wilson has changed jobs. Still in WG14 (but not attending)

3.3 C Memory Object Model Study Group

Sewell: There is Strong support, subject to implementation experience. Current implementations are not consistent with themselves or each other. Uecker is working on GCC, and others are working on LLVM to get real compiler implementations.

2.2.2 Keaton, Project Editor

Larry Jones resigned as project editor for the C standard after 22 years. We are looking for a new project editor and backup project editor. At least one should be a native English speaker. And one should have some experience with LaTeX.

4. Teleconference Meeting Reports

4.1 Report on any teleconference meetings held

None

5. Future Meetings

5.1 Future Meeting Schedule

Please note that in-person meetings may be converted to virtual meetings due to coronavirus considerations.

- 3-7 August, 2020 – Freiburg im Breisgau, Germany (tentative)
  (Freiburg in-person or virtual latest decision date: 25 May, 2020)

We will decide by May 25 if this meeting should be virtual.

- 12-16 October, 2020 – Minneapolis, Minnesota, US (tentative)
  (Minneapolis in-person or virtual latest decision date: 3 August, 2020)

We will decide by Aug 3 if this meeting should be virtual.

- Spring, 2021 – Strasbourg, France (tentative)

Jens Gustedt was the host.

- Fall, 2021 – TBD
- 31 January - 4 February, 2022 – Portland, Oregon, US (tentative)

5.2 Future Mailings

- Post-Virtual / (after March 2020) – 27 April 2020
- Pre-Freiburg – 6 July 2020
- Post-Freiburg / Pre-Minneapolis – 14 September 2020
- Post-Minneapolis – 9 November 2020

Keaton:
I will decrement all of these dates by three days. So 27-Apr -> 24-Apr Getting document numbers is painful, Dan Plakosh traditionally does them in bulk on Sunday evenings. That is why I want to decrement these dates by 3 days, so they fall on a Monday. I would like everyone to request document numbers 1 week before deadline (Friday). So Dan can assign document numbers on the weekend.

8.23 Other Topics

- Ballman: Can we discuss interoperability with WG21?

  Keaton: In order to co-locate meetings, I must meet with Herb Sutter, who has a hard time finding meeting venues.
  Sewell: Perhaps we should co-locate one meeting with WG21 as an experiment?
  Bhakta: Companies will send less people if meetings are co-located, which cuts down on expertise.
  Keaton: If we have back-to-back (serial) meetings, we can schedule similar topics (e.g. memory model) near the weekend both committees meet.
  Could we stagger meetings so one is Wed-Sun, with partial overlapping?
  Bhakta: The answer is the same; less people would go.
  Pygott: Given the possibility of teleconferencing meetings, could WG21 invite us to join virtually?
  Ballman: Given their size they have shown little interest in holding teleconferences.
  Keaton: Decision: We could invite WG21 to join our teleconference, as long as we are having them.

- Keaton: Tabled: How well do virtual meetings work for WG14, we will review? Addressed Friday.
- Ballman: Can we produce a written list of how-we-do-things for new members? Can we document things on Keld's website?

  Keaton: We can change that now. I have a volunteer who will revamp our website.
  Ballman: Does this require a C-numbered paper, or should we iterate on the website?
  Keaton: If someone sends wording to me, I will check for a procedural reason to suggest a change, and post to the reflector.
  Ballman: Action Item: I will construct wording for you.

- Pygott: What do we follow by discussing document numbers, since we should not be using Defect Reports (DR's) anymore?

  Pygott: AKA do we want to move away from DR's, and move towards some issue tracker or wiki?
  Hedquist: We replaced DR's with Clarification Requests (CR's). We just have not implemented them yet.
  Keaton: We thought ISO required us to use DR's. We actually do not, that is my impression.
  Seacord: Did Pygott investigate bug tracking systems?
  Pygott: We use Mantis for tracking specific changes to documents. We use a wiki for creating the documents.
  ???: MISRA wiki & Mantis are on a standalone box in London, not hosted by LDRA. Self-hosting is free. Hosting on Mantis's tracking server is not free.
  Martin: We are also using Gitlab (not Github). Is "publicly accessible" a requirement? You can read it w/o an account, but need an account to submit anything.
  Keaton: "publicly accessible" is not required by ISO. I like "world-visible, but group-writable".
  Uecker: University of Goettingen TWDC runs computing clusters for the Max Planck Institute. They could host for us, providing everything Gitlab offers. Here is a link: https://gitlab.gwdg.de/iso-c. (This is otherwise unrelated to Gitlab)
  Keaton: The ISO text "source code" (AKA LaTeX) must remain private, so non-commercial Github could not host it.
  Banks: Gitlab allows private repositories for non-profits.
  Keaton: But ISO is not a non-profit, and neither are the national bodies. Also, Github is now owned by Microsoft, and that may create conflicts-of-interest.
  Blower: How does WG21 handle this? They use Github. Eric Keene suggested that WG21's C++ technology is available to us: https://github.com/cplusplus
  WG21 achieves privacy through the C++ foundation, which pays for a private Github repository.
  Keaton: Uecker, can you separate things on your solution, so that source code is group-readable but issues are world-readable? Also, can your Gitlab solution generate document numbers?
  Uecker: Action Item: I will investigate (document number generation, and source-code can be made group-readable)
  Bhakta: We could have one project for clarification reports, and another project for documents. Each project has issues, and Gitlab manages the numbering automatically.

- Can virtual meetings start one hour later?

  Keaton: Decision: Not this week, but perhaps for subsequent teleconferences.

6. Document Review

Monday

- 6.1 Weimer, More optionally per-thread state for the library implementation [N 2444]

  Decision: Should this document be incorporated into the standard, modulo any changes for C11? 16-0-4

Tuesday

- 6.2 Tydeman, printf of NaN() [N 2446]

  Bhakta: Uecker had a paper which is more complete, but this paper is still progress.
  Straw Poll: Should N2446 be incorporated into the standard? 14-0-1

- 6.3 Tydeman, Missing example in 6.5.6 [N 2447]

  Uecker: That "should" should be changed to "may":
  OLD: Conforming implementations should diagnose an "array bounds violation."
  NEW: Conforming implementations may diagnose an "array bounds violation."
  Fred: I agree to this change.
  Fred: I would also suggest removing this clause:
  REMOVE: Hence a slice of an array is not an independent object.
  ???: We do not consistently recommend when diagnostics get generated; this is inconsistent.
  Bhakta: We should not wordsmith the text. We would like something similar with adjusted wording.
  Straw Poll: The committee would like something along the lines of N2447 in C2X: 16-0-1

- 6.4 Meneide, nodiscard("should have a reason") [N 2448]

  We started to table this because Meneide is not here yet. However, discussion arose:

  Bhakta: What character set are these string literals?
  Ballman: String literals are also used in #error and $\text{static}_{\text{assert}}$.
  Ballman: N2448 does not specify these. Keaton, do we need a paper for this change request?
  Keaton: Yes, we need a paper.
  Action Item: Ballman will study the character encoding type for string literals from

#error, static$_{assert}$, and now nodiscard() this and write a paper.
Straw Poll: The committee would like to incorporate N2448 into C2X: 15-0-0

- 6.5 Follow-up from earlier meetings on C Memory Object Model Study Group discussions

  Tabled: Sewell is not here yet. Addressed Tuesday.

- 8.1 Thomas, C2X proposal - Names and locations [N 2476]

  Thomas was not here so Bhakta championed this paper.

  Straw Poll: Does the committee want to make suggested change #1 as described in N2476? 14-0-1
  Straw Poll: Does the committee want to make suggested change #2 as described in N2476? 9-1-7

  There was some discussion about whether this constitutes consensus. We decided that it does constitute consensus.

  Tydeman: Suggest we replace "copy" with "move"?
  Seacord: Let somebody submit a paper to that effect. We have already voted on this.
  ???: What is the difference between "deprecate" and "obsolesce"?
  Ballman: "deprecate" has implications for the "deprecated" attribute.
  Straw Poll: Does the committee want to make suggested change #3 as described in N2476? 14-0-1
  Straw Poll: Does the committee want to make suggested change #4 as described in N2476? 14-1-0
  Bhakta: Fred declared that this is a normative change. So all changes pass.

- 6.13 Seacord, Zero-size Re-allocations are Undefined Behavior [N 2464]

  Straw Poll: Does the committee want to adopt N2464? 16-0-0
  Tydeman: There is a statement in one of the annexes about undefined behavior that says: "realloc of 0 is an obsolescent feature." This is in s7.31.14p2. It should be removed.

- 8.3 Ballman, Allowing unnamed parameters in a function definition [N 2480, an update to N2381]

  Bhakta: Strike or correct the footnote; it is non-normative but significant, and the variable is accessible because it is on the stack.
  Keaton: We can strike the footnote, because it is non-normative.

There was some discussion on the footnote, but no discussion on the rest of the paper.

Straw Poll: Does the committee want to adopt N2480 with the footnote reading: "A parameter that has no declared name is inaccessible by name within the function body."
? 14-2-1
Straw Poll: Does the committee want to adopt N2480 with the footnote removed? 12-2-3
Action Item: Ballman: Please submit a followup to N2480 with the updated footnote reading for our future project editor; it already has committee approval.

- 6.5 Follow-up from earlier meetings on C Memory Object Model Study Group discussions
  - The Memory Object Model Study Group requested time to follow up on the previous meeting's discussions. For reference, the papers from the previous meeting are shown here:
  - Sewell, Moving to a provenance-aware memory model for C: proposal for C2X [N 2362]

    Sewell: How do we educate WG14 so that they feel confident enough to approve this?

    There was lots of discussion.

    Keaton: The committee should just man up and study the whole subject, eventually approving everything.
    Sewell: Gustedt was our project editor, but is not anymore. We need a new one.
    Sewell: N2362 is close to a TS. How could we make it into a TS?
    Keaton: Propose it as a new work item. Get 5 countries to vote to make it a TS. The process takes roughly a year, and involves several rounds of voting. No editing is required to create a new work item. For a TS, the committee must agree on the technical content, but need not agree that it go into the standard.
    Sewell: I now think we should make this document a TS.
    Keaton: Let us make time, later this week, to vote on making N2362 into a TS.
    Tabled: Vote on making N2362 into a TS. Addressed Wednesday.

  - Sewell, Exploring C Semantics and Pointer Provenance [N 2311]
  - Sewell, C provenance semantics: examples [N 2363]
  - Sewell, C provenance semantics: detailed semantics [N 2364]

Wednesday

- 6.6 Gustedt, Revise spelling of keywords v4 [N 2457]

Ballman: s6.4.1p2 is confusing because we now have two different types of keywords.
Gustedt agreed with this edit, but did not have a better solution. He prefers that no keywords can be subject of #define or #undef directive Relevant text is also in s6.10.8.4.
Myers: Eliminate them as macros, these are now keywords.
Ballman: These keywords could be defined in older headers as macros.
Krause: C allows us to have macros that happen to be keywords. So I am fine with the current text.
Ballman: Keywords may be already defined as macros. This is an obsolescent feature.
Bhakta: Why are these different than regular keywords?
Ballman: If I understand correctly, they could already be #define'd or #undef'd before. Code could #include <stdbool> and #undef bool.
Wiedijk: Old code would no longer compile if this was included…I cannot define these as macros anymore. Right?
Bhakta: Right!
Meneide: Perhaps Gustedt wanted "bool" to always mean the same thing on any platform…that is why he disallowed re-defining it.
Krause: It is better to leave this out of the standard.
Gilding: It sounds like this harms compatibility. Myers' suggestion to have them defined in the headers.
This changes the compatibility test from "Is bool available?" to "Has bool been defined by the header?".

There was much discussion.

Ballman: Old spellings are deprecated in this paper, in footnote 75.
Keaton: We should converge on what we want to ask Gustedt to do with this paper.
Ballman: We could propose to remove this sentence from 6.4.1p2:
"None of these shall be the subject of a #define or #undef preprocessing directive."
Keaton: It sounds like the proposal is to move this footnote to Future Language Directions:
"75) These alternative keywords are obsolescent feature and should not be used for new code."
Myers: There is a typo: "correspondig"
Myers: For s7.18p2, we should remove "bool" from the set of macros that can be undefined and redefined.
Ballman: I am happy to champion this paper if Gustedt is no longer interested.

- 6.7 Gustedt, Make false and true first-class language features v2 [N 2450]

  Keaton: This is a mistake on my part.

- 6.8 Working draft updates
  - Gustedt, ISO/IEC 9899 working drafts November 2019 [N 2454] / February 2020 [N 2478]

    Ballman: I found one change, which I posted about on the reflector. I might have missed other changes.
    Keaton: Please review these and note other necessary changes, for our next editor.

  - Gustedt, ISO/IEC 9899 working draft w/diffmarks November 2019 [N 2455] / February 2020 [N 2479]
  - Gustedt, ISO/IEC 9899 editor report November 2019 [N 2456]
- 6.9 Gustedt, Make false and true first-class language features v3 [N 2458]

  Ballman: There are some editorial problems:

  - s6.4.4.5.1: editorial typo "for use as are integer literals". Gustedt will fix this.
  - Footnote 88: "presents results", was awkward, I proposed rewording, and Gustedt agreed.
  - "pp number" in prose. Gustedt is aware of this problem, but did not have a solution.

  Meneide: will not making these keywords break my code that #define's true & false?
  Keaton: If I recall correctly, Gustedt believed that WG14 had 20 years to make this change, so it is OK if minor breakage of this nature occurs.
  Keaton: It sounds like we want this; we just need to accept the other paper (N2457) first.

- 6.10 Gustedt, Add an interface to query resolution of time bases v3 [N 2459]

  Keaton: I had made a typo, this section was originally 6.19.
  Does anyone want to see this go forward?
  Bhakta: Yes.
  Ballman: It just brings in some POSIX concepts to C. It sounds reasonable to me.
  Keaton: Are we ready to vote on it?
  Svoboda: No.
  Action Item: Keaton: Add a vote for N2459 on next meeting agenda. We subsequently voted on it and struct this action item.
  Bhakta: Why are we not voting? Is it because of Lack of expertise? I would vote yes.
  Svoboda: We are not voting because of lack of expertise and a champion to promote this paper, since Gustedt is not here.
  Meneide: It is a C implementation of a standard POSIX function ($\text{clock}_{\text{getres}}$).
  Pygott: Why is "base" an int rather than an enum type?

Ballman: "base" is usually an int in other standard functions.
Hedquist: We have always viewed POSIX extensions of C standard items as good. Also Linux is getting standardized. Should we worry about Linux incompatibility as well?
Bhakta: It is useful as an independent function; without it there is no independent way to get the resolution of your clock.
Wiedijk: The $timespec_{get}$ function was in C11.
Svoboda: It is in the Linux Manpage project, but not on my mac. Is it in POSIX?
Seacord: Yes, it is in POSIX ISO9945-2009 or newer.
Myers: $clock_{getres}$ has been in POSIX since 1993.
Keaton: I think we are ready for a vote.
Straw Poll: Does the committee wish to adopt N2459 with $clock_{getres}$ changed to $timespec_{getres}$ in the proposed wording? 12-0-5. Passes.
Keaton: So there is no need to vote on this at next meeting. That action item can be stricken.

- 6.11 Gustedt, Add new optional time bases v3 [N 2460]

  Meneide extemporaneously championed the paper.

  Bhakta: There are some issues with incomplete specifications throughout the paper. For example $CLOCK_{MONOTONIC}$ and $TIME_{CPU}$ depends on concepts not in C2X.
  Svoboda: Bhakta, does POSIX define $CLOCK_{MONOTONIC}$ to your satisfaction? (This is academic because we cannot just reference the POSIX standard in C2X).
  Bhakta: Yes, but it is also specific to the POSIX platform, so that also makes it inappropriate for C2X.
  Gilding: Is it acceptable to allow $TIME_{THREADCPU}$ to degrade gracefully?
  Meneide: There is optionality in the wording for $TIME_{THREADCPU}$. It might be useful to have monotonic time in C2X.
  Bhakta: I am okay with something along the lines of this proposal, but not this proposal as written.
  Krause: Embedded platforms can allow us to implement $CLOCK_{MONOTONIC}$, but they may not let us implement $TIME_{UTC}$.
  Tydeman: $TIME_{THREADCPU}$ is optional according to text color.
  Straw Poll: Do we want something along the lines of $TIME_{MONOTONIC}$, as proposed in N2460, in C2X? 12-0-3 passes
  Straw Poll: Do we want something along the lines of $TIME_{CPU}$, as proposed in N2460, in C2X? 5-2-9 fails
  Straw Poll: Do we want something along the lines of $TIME_{THREADCPU}$, as proposed in N2460, in C2X? 4-4-6 fails

- 6.5 Follow-up from earlier meetings on C Memory Object Model Study Group discussions

  Keaton: I would like to vote on having the committee pursue a TS based on N2362.
  Seacord: Who would be doing the work to make this a TS?
  Sewell: There is enough resources to do the TS, perhaps make this one bundle.
  Ballman: Is this a new work item (with a stopwatch), or are we requesting authors?
  Keaton: It should be a new work item. The deadline is three years after the first draft is attached to a new work item.
  Ballman: Should a new work item go in to the poll?
  Keaton: Yes

  There is more discussion.

  Straw Poll: Should the committee pursue a new work item for a Technical Specification on provenance along the lines of N2362? 19-0-0

- 6.12 Gustedt, Synchronization at thread and execution termination v3 [N 2461]

  Ballman extemporaneously championed the paper:
  At Ballman's suggestion, Gustedt introduced "handler" to discuss functions used as atexit handlers and $at_{quickexit}$ handlers.

  Seacord: s7.22.4.7p3 speaks of "sequence points". Are we not using "happens-before" now?
  Bhakta: We still have "sequence points" throughout the standard.
  Ballman: If multiple threads register handlers, that should be specified, like C++ does. We also need to notify WG21 about it.
  Svoboda: This paper should also add these sequence points to Annex C, which lists sequence points in the standard.
  Tydeman: According to s6.5.2.2p10, every function call is a sequence point.
  Bhakta: But the paper adds sequence points to handlers.
  Seacord: This paper leaves under-specified that handlers in different threads could complete in any order.
  Sewell: The standard should globally eliminate "sequence point", and use "sequence before/after", AKA "happens-before".
  Keaton: Good idea. But that should be in a separate paper.
  Straw Poll: Should the committee pursue something along the lines of N2461? 15-0-2

- 6.14 Seacord, $intmax_t$, a way forward [N 2465]

Bhakta: Your paper looks just like N2498, except for the widest-int types, right?
Seacord: This paper changes the interface to use the new typedefs (AKA intbasicmax types). It includes replacing $intmax_t$ with $intbasicmax_t$.
Hoffner: Should widest type target largest syntactically-supported data type or largest efficiently-implemented data type?
Tydeman: I did not see any preprocessor arithmetic in this paper. It uses $intmax_t$.
Seacord: This is addressed in page 3, paragraph 3.
Ballman: We do have $int32_{fastt}$; we could always add $int_{widestfastt}$. But the concept of "widest int" is a mistake. Introducing ext_{int}, means there is no widest type (in the English sense), or a type that is a disaster (e.g. it must support 16-million-bit ints in Clang). If ext_{int} fails, at some point, architectures will want 128-bit ints, so API breakage will still happen. I prefer using $intbasicmax_t$, and kill $intmax_t$.
Seacord: I agree with regard to conflict with "widest" paper, but I disagree with regard to API breakage. The widest type should not be used in API's, and this is easy to diagnose.
Gilding: If we standardize $intbasicmax_t$, this will still break API's (when someone wants "long long" to be 128 bits).

There was more discussion.

Bhakta: For s7.8.1p7 wprintf lines, the text is the same for %i and %j. Is this a cut-and-paste error?
Seacord: Yes, good catch, thanks.
Svoboda: I recommend separating this into two proposals, one with widest type, and one with $intbasicmax_t$.
Svoboda: This should be voted on with N2498.
Seacord: I agree. Let us defer votes until we hear Uecker's paper. I also agree with splitting this proposal.
Bhakta: I would like to see an update to 2.4 to see what happens to the functions there.

- 6.15 Uecker, $intmax_t$, again [N 2498]

Seacord: If we turn functions like strtomax() into function-like macros, we would still want the API functions, and eventually actual functions with larger types.
Myers: I am dubious about the obsolescence of %j specifier. It is useful for POSIX types like $time_t$. I am also dubious about strtomax().
Uecker: I do not like library function-like macros that hide underlying functions.

There was more discussion.

Krause: Do not replace "max" functions with generic macros. Use "This function might be implemented purely as a macro."
Ballman: With either paper, because the preprocessor is designed to work with $intmax_t$ and $uintmax_t$, this prohibits the ability to specify large integer literals. How big a problem is this?
Myers: I am dubious about asterisks in conversion specifiers. That makes the variadic argument type unknown until run time.
Uecker: Yes, that is a problem, I can take it out.

Thursday

- 6.14 + 6.15 discussion (cont from Wednesday)

Svoboda: Does Seacord's paper have a format specifier for widest type, or something analogous to Uecker's %wN specifier?
Seacord & Uecker: That is in both papers, but Seacord did not have %W*, and Uecker had it, but removed it.
Svoboda: We should discuss first the $intbasicmax_t$ aspects, and then the $widest_{type}$ aspect. Ideally Seacord & Uecker would collaborate on a single way forward on $intbasicmax_t$, or we unify their papers now.
Seacord: I tried to replace $uintmax_t$ with long long int. Gustedt noted that some platforms use long while others use long long as the widest type. Using long long int as the maximal type behaves differently on platforms that use long int as the longest type, even though they also support long long int.
Bhakta: Seacord's paper is more portable. Uecker's paper is cleaner, but portability is more important than cleanliness.
Ballman: I like the %W format specifier, but is there any implementation experience with that? Microsoft has I32 and I64 types.
Bhakta: I hate I32 and I64 because of lots of bugs, but customers like it.
Ballman: I also like fixing API's to no longer be tied to $intmax_t$. I would like to see the smallest possible fix, like removing $intmax_t$ from API's. I prefer not replacing $intmax_t$ with anything.
Gilding: This removes semantics of a "widest integer type". Is that the intent of either paper?
Seacord: The concept of "widest integer type" does not make sense. There is the largest of standard integer types, largest of standard+extended integer types, and largest integer type altogether, which includes 16 million bits.
Krause: $intmax_t$ should be a distinct type from long long, but such code is so exotic we should not worry about breaking it.
Seacord: My paper preserves the implementation that $intmax_t$ still exists and is not

necessarily the largest type.
Keaton: There are Two decisions: one about what to do about basic integer types, and one about a maximal integer type.
Bhakta: Uecker's paper addresses both together. There are no other proposals, besides Aaron's simple proposal to deprecate $intmax_t$.
Seacord: "intwidest" from Seacord's paper and "$intbasicmax_t$" from Seacord's paper. There is also the conversion from $intmax_t$ to long long from Uecker's paper.
Blower: I would also like to see a vote on Ballman's proposal.
Wiedijk: What constitutes "basic" types?
Keaton: The standard partitions integer types as "basic" integer types and "extended" integer types.
Bhakta: The standard defines long long as the maximal basic (signed) integer type.
Seacord: These "n-sized" integer types are still speculative. Could they be considered something other than "basic" or "extended"? They render my "widest integer type" concept as bonkers.
Wiedijk: I do not see why API compatibility is important.
Gilding: I care less about basic vs. extended, than which will be used in implementations.
Keaton: There are some 8-bit systems where "int" requires multiple cycles.
Bhakta: There is the precedence of lock-free macros for atomic types.
Uecker: We already have int_{fastN} types.
Krause: Often there is not even a largest type supported in hardware. Some hardware supports larger types that only support addition not multiplication, or support multiplication but not division…how do those rank as widest types?
Keaton: OK, we should vote on:

- $intwidest_t$ from N2465?
- $intbasicmax_t$ from N2465?
- keeping $intmax_t$ with Uecker's changes from N2498?
- deprecating $intmax_t$, as suggested by Ballman?

Pygott: Should we have a separate vote on %W from N2465 and N2498?
Keaton: Yes
Bhakta: Another voting possibility: Do we want to go in the direction of N2465 or direction of N2498 or deprecating $intmax_t$? An exclusive vote: choose one of three directions.
Seacord: I would like to separate out the $widest_t$ in my paper; it has less support.
Svoboda: Keaton's voting scheme addresses your concern.
Seacord: My $intbasicmax_t$ does not grow with hardware, while Uecker's $intmax_t$ does.
Ballman: My deprecation suggestion does not apply to what to do about API's.

Keaton: We should have a three-way vote regarding what happens to existing function signatures that use intmax_t: intbasicmax_t from N2465 or long long from N2498, or keeping intmax_t (AKA status quo).
Uecker: Possible three-way vote: deprecate intmax_t, introduce intwidest_t (keeping intmax_t fixed), make intmax_t grow as originally intended.

There was much discussion about what straw polls to take.

Wiedijk: Is an implementation allowed not to support long long? Also they are called "standard integer types" not "basic integer types".
Ballman: The standard does talk about basic types, but those are char and the floating-point types.
Straw Poll: Do we want to make any changes to intmax_t? 12-5-0 (passes)
Keaton: There are three options for types (not API's): intbasicmax_t, long long, or deprecate intmax_t.
Keaton: After some feedback, I now think there are three options for the idea of a maximum standard integer type (regardless of API's), something along the lines of: intbasicmax_t, long long, or deprecate the idea of a maximum standard type.
Seacord: We should have a two-way vote on having a typedef for maximum int type vs. not having an explicit max type.
Straw Poll: Should the committee pursue something along the lines of intbasicmax_t from N2465, strictly as a type declaration? 6-9-3 (fails)
Straw Poll: Should the committee pursue something along the lines of a growing type as described in N2465 and N2498? 6-9-3 (fails)
???: This does not kill intmax_t but it does make it "open season" on intmax_t. We would welcome papers that kill it (and presumably modify or delete API's that rely on it).
Keaton: It sounds like we need to solicit a proposal to deprecate intmax_t, and perhaps replace it with something better.
Krause: Here is a way forward: Identify a few API's to change (say from intmax_t to long long), and a few functions that can be removed, because they are specific to intmax_t.
Ballman: I have no interest in resizeable types. So I think we should deprecate intmax_t without replacing it. We need a non-resizeable replacement for intmax_t, and we still need to discuss formatted-io conversion specifiers.
Keaton: But we voted against intbasicmax_t.
Ballman: We cannot use a specific type like long long. So we need a new type, unless we break ABI compatibility.
Uecker: I do not see breaking ABI's with using long long.
Ballman: Clang lets you overload function names in C (via C++). Using long long

would make this overloading awkward.
Uecker: Do we want to remove it or just deprecate it?
Straw Poll: Should the committee pursue something along the lines of deprecating intmax$_t$ as a type? 7-3-5 (consensus)

Myers: We will need to do something about preprocessor arithmetic, too.
Ballman: We need to remove %j as well.
Seacord: So our options are: eliminate API's using intmax$_t$, or replace intmax$_t$ with another fixed-width non-growing typedef, or replace with long long. Are there any other options?
Keaton: The middle option is out, because we voted against intbasicmax$_t$.

Seacord: So let us deprecate intmax$_t$ and API functions with "max" in the name that use intmax$_t$, and convert remaining functions to use long long.

Bhakta: Deprecating functions is not the only way to go forward. I do not want that.
Seacord: What do you see as a way forward?
Bhakta: We should use long long int as a complete replacement for intmax$_t$.

Keaton: We have reached the limit of design work we can do at this meeting. We should not design more on the fly; we need a proposal to refine our thinking.
Uecker: Can we vote on the %w specifier and them move on?
Straw Poll: Should the committee pursue something along the lines of %w from N2498 and N2465? 9-2-5 (passes)
Straw Poll: Should the committee pursue something along the lines of intwidest$_t$ from N2465 vs. intmax$_t$ as a growing type from N2498, strictly as a type declaration?

Straw Poll: Should the committee pursue something along the lines of keeping intmax$_t$ with Uecker's changes from N2498?

Straw Poll: Should the committee pursue something along the lines of deprecating intmax$_t$, as suggested by Ballman?

Action Item: Seacord will submit a paper that proposes just %w (without the other contents of N2465)

- 6.16 Svoboda, Towards Integer Safety [N 2466]

  Krause: Do we really need all those types? Which types do people want to use? Every type needs a justification.
  Svoboda: I was aiming for maximum capability, so I did not assume some types should not have checked analogues.
  Pygott: Why does the constructor takes a boolean?
  Svoboda: That made the proposal academically complete. It is probably useful to indicate an error, perhaps buried in a conditional expression.
  Myers: The paper says "constant expression" a lot. Which kind of constant expression? Also, which functions must be functions (with external linkage) or macros?

Svoboda: Every function and macro is necessary although perhaps not all require external linkage.
Myers: For example, "Arithmetic constant expression" vs. "literal constant expressions". You need to be more specific in the text.
Svoboda: OK
Ballman: How does the checkecd$_{overflow}$ types work with extended integer types (N2472)?
Svoboda: I would guess we would have checked versions of extended integer types, but I need to study N2472.
???: Do these types have ranks? Also, divide can overflow, why not handle that?
Svoboda: Safe division is invention, not covered by GCC or SafeInt or Java.
There are no explicit ranks or sizes in the paper, although those can be implicitly derived from the standard integer types.
Our functions do not cover conversions between different integer types, just checked vs. unchecked.
Bhakta: Does this align with ISO 10967 naming & operations?
Svoboda: I do not know, I am unfamiliar with ISO 10967.
Svoboda: Should s3.4.3.p3 say "signed integer overflow". I will submit a clarification request.
Keaton: I agree.
Action Item: Svoboda will submit a clarity request for s3.4.3p3 (as outlined in N2466, bottom of page 1)
Bhakta: I would like an early draft (before next meeting) to make sure this is consistent with regard to constant expressions (in floating-point in another proposal).
Uecker: Requiring constant expressions is very limiting on how checked integers may be implemented. They could not be implemented as a struct.
Bhakta: I agree.
Gilding: It is not complicated to allow structs as constant expressions, even though the standard does not mandate it.
Svoboda: How big a rabbit hole is "structs as constant expressions"? It sounds like a simple obvious thing that WG14 could mull over for days.

- 6.17 Thomas, C2X proposal - NaN and infinity macros [N 2469]

  Ballman: s6.6p4 seems to contradict this paper.
  Keaton: No, it just cannot be a constant expression.
  Straw Poll: Does the committee wish to adopt N2469 into C2X? 7-2-7
  Bhakta: This does not help the implementation issue; it is still magic. This makes the paper harder & less clear.
  Keaton: Let us re-vote.
  Straw Poll: Does the committee wish to adopt N2469 into C2X? 0-8-7 (fails)

Friday

- 6.18 Meneide, Preprocessor embed - Binary Resource Inclusion [N 2470]

  Tabled until Meneide is here. Addressed Friday.

- 6.19 Douglas, Stackable, thread local, signal guards [N 2471]

  tabled until Douglas is here. Addressed Friday.

- 6.20 Blower, Adding Fundamental Type for N-bit Integers [N 2472], updated to [N 2501]

  Gilding: How do these interact with bit fields? Bit fields have rank & promotions which differ from n-bit-ints.
  Hoffner: Bit fields are extremely hard to optimize, at least when you do not have byte alignment.
  Gilding: Should you make extended ints non-addressable, just like bit fields?
  Hoffner: It is more interesting to use them with as few constraints as possible. I do not see the benefit of making them non-addressable.
  Blower: I wrote up a notation for n-bit int literals, but that did not make it into this proposal.
  Ballman: These should be regarded as normal integers. I disagree that signed ext-int of size is not consistent with bit fields. A signed 1-bit ext-int is nonsensical; I am glad the paper recognizes that.
  Krause: Since we standardize two's complement (signed ext 1-bit) 0b1 = -1.
  Ballman: Developers assume everything is two's complement. I have only seen it used incorrectly; I have never seen it used properly.
  Krause: Is there considered to be a maximum number of bits for n-bit integers?
  Hoffner: We have not considered a maximum number.
  Straw Poll: Is the committee interested in seeing something along the lines of N2501 in C2X? 13-0-1 passes

- 6.18 Meneide, Preprocessor embed - Binary Resource Inclusion, [N 2499], updates [N 2470]

  Hoffner: How is the support for the -E option? I dislike features that do not get along with compiler option -E.
  Meneide: I have an implementation that works with -E. It produces the same output whether or not you use -E.
  Hoffner: I am confused. Are you allowed to have different embedded types than the array you map it to?
  Meneide: You can use different types, although you may get truncation warnings. It

behaves as if it generated an initializer list, which can generate warnings.
Ballman: I have a nit: The grammar production for #embed has a production rule for headers.
With regard to the limits.h hook-in: does that really help? limits.h is about target limits, not the host limits (of the compiler's platform).
Meneide: The grammar works for target architecture. We want it to use values for the target architecture, not the host architecture. It reads values on host architecture. It would be a compiler implementation detail about how to convert to target architecture.
Ballman: How is endianness handled?
Meneide: The programmer is responsible for byte-swapping; this is not covered by #embed.
Ballman: I could embed a single int as a compile-time integer; it could be byte-swapped for the host or for the target architecture.
Meneide: OK, i will add something about that.
Gilding: Back to -E. Is this supported by compiler magic?
Meneide: It keeps the filename for diagnostic information, but afterwards the builtin is processed as what it is. After preprocessing, the file need no longer exist.
Gilding: I had implemented this a while back. It does massively improve performance.
Krause: It is not useful to me to specify type. It would be most useful to specify signed vs. unsigned + number of bits.
Meneide: It only has to support these token names, not user-defined types. If you pass in a type, it consults limits.h for type sizes.
Wiedijk: I really like this. But it should not be in the preprocessor. The hacking around the types suggests this should be done in the compiler proper so that it does understand the type system.
Meneide: I did propose an in-language version of this to another committee, which uses distributed builds. If it is a language feature, you have arbitrary computations, which makes it difficult for them to rectify dependencies. A preprocessor solution guarantees that dependency management does not need to wait until the front-end is done.
Wiedijk: What is "built-in with compilers" about?
Meneide: Those are details for my implementation. Builtins are generated by the preprocessor. I could put it in Clang & GCC, but not Microsoft Visual C++ which is not open-source.
Straw Poll: Would the committee prefer #embed using type tokens (yes) or #embed using a bit-based constant expression (no), (N2499)
2-10-3 (the committee prefers bit-based)

- 6.21 Thomas, C2X proposal - why no wide string strfrom functions, [N 2490], updates [N2475]

Straw Poll: Does the committee wish to adopt N2490 into C2X? 13-0-1

- 6.19 Douglas, Stackable, thread local, signal guards [N 2471]

  Douglas is still not here. and no one championed this paper.

  Bhakta: The phrase "broken-pipe", listed on page 4, is specific to POSIX. There might be other signals that do not belong here. "broken-pipe" does not apply to other implementations.
  Gilding: I like the general direction. Many developers end up building something vaguely like this.
  Bhakta: There was very week support for this in Ithaca.
  Straw Poll: Is the committee interested in seeing something along the lines of N2471 in C2X? 5-2-6 (weak consensus in favor)

- 6.16 Svoboda, Towards Integer Safety [N 2466] revisiting

  Gilding: Regarding compile-time constants, there is no problem with this for host vs. target architecture. This feature is not strictly necessary. Expanding what must be a constant expression could be done later.
  Svoboda: Yes, it is easy to take out. And it is hard to guarantee on multiple platforms.
  Ballman: I am uncomfortable due to extended int proposal, how does it interact with this one?
  Svoboda: My proposal does not address integers in the future; that is too hard. The proposal is focused on current integer types.
  Seacord: I do not see "compile-time constants" in your paper. Why is that in your straw-poll question?
  Svoboda: Right, the term should be "constant expressions" instead.
  Straw Poll: Is the committee interested in seeing the core and supplemental proposals in N2466 adopted into C2X as is? 2-7-4
  Straw Poll: Is the committee interested in seeing the core and supplemental proposals in N2466 adopted into C2X if we took out constant expression guarantees? 4-4-6
  Krause: I understand why this is useful for $size_t$, but why would I want this support for $intptr_t$ or $uintptr_t$? We do not need functions, generic interface would be enough. Namespace pollution by functions and supporting types. I would like to see use cases for every type you support.
  Ballman: I want to have concrete function definitions. My clients like addressable functions.
  Myers: I do not necessarily want addressable functions, I just wanted clarification.
  Hoffner: Why do you have this safe integer for integral pointer types? Pointer arithmetic requires more safety than overflow.

Straw Poll: Is the committee interested in seeing addressable functions that detect integer overflow, based off of N2466? 5-1-7 (no consensus)

- 6.19 Douglas, Stackable, thread local, signal guards [N 2471]

  Douglas is here now and championed the paper:

  Bhakta: Did all implementations had less than 31 bits, or signal values under 31?
  Douglas: Signal values under 31.
  Bhakta: My implementation has values that goes up to 0xc00f61412.
  Douglas: How to you go into signal sets?
  Bhakta: I would have to investigate. I think it uses bit masks.

7. Clarification Requests

All clarification requests have been processed.

8. Other Business

1.5: Approval of Previous Minutes [N 2451] (PL22.11 motion, WG 14 motion) (Ballman, Tydeman, approved)

Keaton: We have stubbed out PL22.11 business in old minutes
Blower: There was a little wordsmithing, but no big problems.

2.2.1 Document system

Keaton: Uecker is not here. I wanted to ask him if the C-number generator is separate from other things we could put on the proposed University of Goettingen site, like clarification requests? I will discuss with him offline.
Seacord: I have created an account on that site.
Seacord: Should we solicit N-numbers for papers from Dan for now?
Keaton: Yes, until instructed otherwise.
Ballman: Will we have a mixture of N- and C- numbered papers?
Keaton: Yes
Ballman: If we switch from N-numbers to C-numbers how can we produce a C-numbered paper that is an update from an N-numbered paper?
Keaton: Pick up a C number when they become available, and say it revises an N-numbered document.
Krause: Will the introduction of the C-number system be announced on the reflector?

Keaton: I recommend submitting your paper whenever it is ready, rather than wait for C-numbers. But do what you think best.

2.2.2 WG14 still needs a new project editor & backup editor

Keaton: Please consider nominating an editor from your organization, because Gustedt will not release the source code for the standard until we have a new editor.
Ballman: Why is one person able to withhold the source code?
Keaton: Having two editors was designed to prevent situations like this. Our backup editor resigned shortly followed by our editor. I have Jones' source, but not with Gustedt's latest changes.
Ballman: Do we have any processes in place for preventing this situation from recurring?
Keaton: I have re-distributed the source code from Jones to a few additional people, will do the same once I receive Gustedt's updates.
Barry: He cannot legally withhold the source code.
Keaton: Yes, but I prefer to resolve this with him amicably first.
Krause: Thanks for distributing the source. The latest should be made available to everyone on the committee, so that we can write proposals based on it.
Keaton: Once the source (to the standard) is on Uecker's site, everyone please create an account and "git clone" the source.
Bhakta: That copy is still under ISO non-disclosure, so we should not distribute it outside PL22.11 committee members, right?
Keaton: Right!
Gilding: Actually it should be restricted to WG14 members. Remember, PL22.11 members are a subset of WG14.

Deferral of papers

We have no time to review any more papers. These papers will be deferred until the August meeting.

- 8.2 Alepins, Const functions [N 2477]
- 8.4 Ballman, Querying attribute support (updates N2411) [N 2481]
- 8.5 Ballman, Minor attribute wording cleanups [N 2482]
- 8.6 Ballman, Unclear type relationship between a format specifier and its argument (updates N2420) [N 2483]
- 8.7 Bachmann, Make pointer type casting useful without negatively impacting performance [N 2484]
- 8.8 Bachmann, Add explicit$_{\text{memset}}$() as non-optional part of to C2X [N 2485]
- 8.9 Krause, register at file scope [N 2486]
- 8.10 Krause, short float [N 2487]

- 8.11 Thomas, C support for IEEE 754-2019 [N 2488]
- 8.12 Thomas, C2X proposal - min-max functions [N 2489]
- 8.13 Thomas, C2X proposal - powr justification, wording [N 2491]
- 8.14 Thomas, C2X proposal - note about preserving math function properties [N 2492]
- 8.15 Ballman, What we think we reserve (updates N2409) [N 2493]
- 8.16 Gustedt, a common C/C++ core specification [N 2494]
- 8.17 Tydeman, snprintf [N 2495]
- 8.18 Uecker, Compatibility of Pointers to Arrays with Qualifiers [N 2496]
- 8.19 Uecker, Free Positioning of Labels Inside Compound Statements [N 2497]
- 8.20 Meneide, Preprocessor embed - Binary Resource Inclusion, r2 [N 2499]
- 8.21 Meneide, Restartable and Non-Restartable Functions for Efficient Character Conversions, r2 [N 2500]
- 8.22 Tydeman, Range errors and math functions [N 2506]

8.24 Virtual Meeting Feedback

Keaton: How well did the virtual meeting work for everyone?
Seacord: The meeting was quite productive. Voting was more efficient.
Svoboda: The chat feature was useful, for collecting roll call info, background business, fixing audio & other technical issues, and blowing off steam.
Bhakta: It was less interactive then a face-to-face meeting, with less interruptions.
Svoboda: Yes, also there were less unplanned chats and gossip.
Wiedijk: This worked once, but I do not think we could sustain pure teleconferencing in the long-term.
Bhakta: I also do not want to expand how many meetings we have by using teleconferences.
Keaton: I could see 3 meetings per year, with 2 virtual. But we do not need to decide this now.
Krause: I still like one European meeting per year. Two meetings were not enough.
Keaton: We do not need more virtual meetings if we have two physical meetings per year.

9. Resolutions and Decisions reached

9.1 Review of Decisions Reached

Decision: We could invite WG21 to join our teleconference, as long as we are having them.
Decision: Start subsequent virtual meetings one hour later.
Straw Poll: Should N2444 be incorporated into the standard, modulo any changes for C11? 16-0-4

Straw Poll: Should N2446 be incorporated into the standard? 14-0-1
Straw Poll: Would the committee like something along the lines of N2447 in C2X: 16-0-1
Straw Poll: Does the committee want to make suggested change #1 as described in N2476? 14-0-1
Straw Poll: Does the committee want to make suggested change #2 as described in N2476? 9-1-7 (passes)
Straw Poll: Does the committee want to make suggested change #3 as described in N2476? 14-0-1
Straw Poll: Does the committee want to make suggested change #4 as described in N2476? 14-1-0
Straw Poll: Does the committee want to adopt N2464? 16-0-0
Straw Poll: Does the committee want to adopt N2480 with the footnote reading:
"A parameter that has no declared name is inaccessible by name within the function body."
? 14-2-1
Straw Poll: Does the committee want to adopt N2480 with the footnote removed? 12-2-3
(We had consensus, but weaker consensus than using the modified footnote, so we will not do this)
Straw Poll: Would the committee like to incorporate N2448 into C2X? 15-0-0
Straw Poll: Does the committee wish to adopt N2459 with $\text{clock}_{\text{getres}}$ changed to $\text{timespec}_{\text{getres}}$ in the proposed wording? 12-0-5. Passes.
Straw Poll: Do we want something along the lines of $\text{TIME}_{\text{MONOTONIC}}$, as proposed in N2460, in C2X? 12-0-3 passes
Straw Poll: Do we want something along the lines of TIME_{CPU}, as proposed in N2460, in C2X? 5-2-9 fails
Straw Poll: Do we want something along the lines of $\text{TIME}_{\text{THREADCPU}}$, as proposed in N2460, in C2X? 4-4-6 fails
Straw Poll: Should the committee pursue a new work item for a Technical Specification on provenance along the lines of N2362? 19-0-0
Straw Poll: Should the committee pursue something along the lines of N2461? 15-0-2
Straw Poll: Do we want to make any changes to intmax_t? 12-5-0 (passes)
Straw Poll: Should the committee pursue something along the lines of intbasicmax_t from N2465, strictly as a type declaration? 6-9-3 (fails)
Straw Poll: Should the committee pursue something along the lines of a growing type as described in N2465 and N2498? 6-9-3 (fails)
Straw Poll: Should the committee pursue something along the lines of deprecating intmax_t as a type? 7-3-5 (passes)
Straw Poll: Should the committee pursue something along the lines of %w from N2498 and N2465? 9-2-5 (passes)
Straw Poll: Does the committee wish to adopt N2469 into C2X? 0-8-7 (fails)
Straw Poll: Is the committee interested in seeing something along the lines of N2501 in

C2X? 13-0-1 (passes)
Straw Poll: Would the committee prefer #embed using type tokens (yes) or #embed using a bit-based constant expression (no), (N2499)
2-10-3 (fails, AKA the committee prefers bit-based)
Straw Poll: Does the committee wish to adopt N2490 into C2X? 13-0-1
Straw Poll: Is the committee interested in seeing something along the lines of N2471 in C2X? 5-2-6 (weak consensus in favor)
Straw Poll: Is the committee interested in seeing the core and supplemental proposals in N2466 adopted into C2X as is? 2-7-4
Straw Poll: Is the committee interested in seeing the core and supplemental proposals in N2466 adopted into C2X if we took out constant expression guarantees? 4-4-6 (no consensus)
Straw Poll: Is the committee interested in seeing addressable functions that detect integer overflow, based off of N2466? 5-1-7 (no consensus)

9.2 Review of Action Items

Ballman: Produce initial wording documenting WG14 procedures for new members. Once submitted to Keaton, Keaton will check for a procedural reason to manage procedure documentation, and post to the reflector.

Uecker: Investigate Gitlab-like hosting solution offered by the University of Goettingen. In particular, does it offer a solution for generating document numbers? And can we keep source code (in ISO C LaTeX format) group-readable, but clarification requests world-writable, with some gate-keeping to minimize spam?

Ballman: Study the character encoding type for string literals from #error, $\text{static}_{\text{assert}}$, and now the nodiscard attribute (via N2448) and write a paper.

Ballman: Submit a followup to N2480 with the updated footnote reading for our future project editor. N2480 already has committee approval.

Seacord: Submit a paper that proposes just %w conversion specifier from N2465 (without the other contents of N2465)

Svoboda: Submit a clarification request for C17 s3.4.3p3 (as outlined in N2466, bottom of page 1)

10. PL22.11 Business

PL22.11 Issues only

11. Thanks to Host

11.1 Thanks and apologies to Philipp Krause, the originally intended host

11.2 Thanks to ISO for supplying Zoom capabilities

12. Adjournment (PL22.11 motion) (Tydeman, Seacord, adjourned)