Draft Minutes for 12 – 16 October, 2020

MEETING OF ISO/IEC JTC 1/SC 22/WG 14 AND INCITS PL22.11

WG 14 / N 2605

Dates and Times

Each day will have a half-hour break from 14:30-15:00 UTC.

12 October, 2020	13:00 – 16:30 UTC
13 October, 2020	13:00 – 16:30 UTC
14 October, 2020	13:00 – 16:30 UTC
15 October, 2020	13:00 – 16:30 UTC
16 October, 2020	13:00 – 16:30 UTC

Meeting Location

Please note: Due to the global health emergency, this is no longer a face-to-face meeting.

This meeting is virtual via Zoom.

Meeting information

Please see the ISO Meetings platform (log into login.iso.org and click on Meetings) or contact the convener for the URL and password.

Local contact information

David Keaton <dmk@dmk.com>

1. Opening Activities

1.1 Opening Comments (Keaton)

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Notes
Aaron Ballman	Intel	USA	WG21 Liaison
Rajan Bhakta	IBM	USA, Canada	PL22.11 Chair
Lars Bjonnes	Cisco Systems	USA	

Name	Organization	NB	Notes
Melanie Blower	Intel	USA	
Alex Gilding	Perforce / Programming Research Ltd.	USA	
Barry Hedquist	Perennial	USA	PL22.11 IR
Tommy Hoffner	Intel	USA	
David Keaton	Keaton Consulting	USA	Convener
Will Klieber	CERT/SEI/CMU	USA	
Maged Michael	Facebook	USA	
Clive Pygott	LDRA Inc.	USA	WG23 liaison
Robert Seacord	NCC Group	USA	
David Svoboda	CERT/SEI/CMU	USA	Scribe
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Freek Wiedijk	Plum Hall	USA	
Bill Ash			SC22 CM
Aaron Bachmann	Austrian Standards	Austria	Austria NB
Roberto Bagnara	University of Parma	Italy	Italy NB, MISRA Liaison
Andrew Banks	LDRA Ltd.	UK	MISRA Liaison
Geoff Clare	The Open Group	UK	Invited Guest
Jens Gustedt	INRIA	France	
Philipp Krause	Albert-Ludwigs-Universit	Germany	
Kayvan Memarian	University of Cambridge	UK	
JeanHeyd Meneide	TomTom	Netherlands	
Joseph Myers	CodeSourcery / Siemens	UK	
Miguel Ojeda		Spain	Invited Guest
Peter Sewell	University of Cambridge	UK	Memory Model SG
Nick Stoughton	USENIX, ISO/IEC JTC 1	USA	Austin Group Liaison
Martin Uecker	University of Goettingen	Germany	
Jorden Verwer		USA	Invited Guest
Michael Wong	Codeplay	USA	WG21 Liaison
Joerg Wunsch		Germany	Invited Guest

1.3 Procedures for this Meeting (Keaton)

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 Approval of Previous Minutes [N 2581] (PL22.11 motion, WG 14 motion)

Gustedt: I prefer to defer minutes until the next meeting. There are some errors and I haven't gotten to review it.

1.6 Review of Action Items and Resolutions

Action Item: Keaton: To coordinate with Herb Sutter on establishing co-located study groups between WG14 and WG21.
Done
Action Item: Stoughton: To submit a proposal on the removal of the removal of asctime_r and ctime_r.
Done in N2566
Action Item: Stoughton: To write a POSIX response to n2526 before the next committee meeting.
Done in N2565
Action Item: Ballman: To produce wording documenting WG14 procedures for new members.
Done at http://www.open-std.org/jtc1/sc22/wg14/www/contributing
Action Item: Ballman: To study the character encoding type for string literals from #error, static_assert, and nodiscard.
Done in N2563

1.7 Approval of Agenda [N 2582r] as amended by adding 9.5 and 9.6 (PL22.11 motion, WG 14 motion)

Motion to Approve. seconded, passed

1.8 Identify National Bodies Sending Experts

Austria, Canada, France, Germany, Italy, Netherlands, UK, US

1.9 INCITS Antitrust Guidelines and Patent Policy

1.10 INCITS official designated member/alternate information

2. Reports on Liaison Activities

2.1 ISO, IEC, JTC 1, SC 22

- 2.1.1 Change in policy for WGs: C2x becomes C23

  Keaton: C2x now has a publishing deadline: August 2023

2.2 PL22.11/WG 14

- 2.2.1 Outreach

  Ballman: We need some means of communicating with users. If we need the bureaucracy of a study group, it will never get off the ground.
  Keaton: A study group needs a charter.
  Seacord: I am mainly concerned about attracting new membership to the committee. We should also include marketing the language.
  Meneide: The new document system will not only let the public browse papers but also leave comments on those papers.
  Stoughton: Recent changes in directives complicate outside attendance of these meetings
  Keaton: To join, Keaton must send out a convener's invitation, and notify invitee's national body, who can veto them (if they haven't paid membership dues.
  Gustedt: Could we operate like WG21, focusing on 'official' SG meetings with few formal WG14 meetings?
  Blower: How about a public bug report mechanism?
  Keaton: Ballman is working on that, would be useful.
  Stoughton: The Austin group has a public bug-reporting system, so we can reach out to bug reporters and invite them to meetings. Meetings do not operate under SC22 rules, so they are easy for guests to join.
  Keaton: Perhaps we should reserve 1 meeting day for outreach.

2.3 PL22.16/WG 21

- 2.3.1 C/C++ Collaboration

  Keaton: Someone has proposed to chair both groups for liaison-ing WG14 & WG21. We are awaiting his employer approval.

2.4 PL22

2.5 WG 23

2.6 MISRA C

2.7 Other Liaison Activities

3. Reports from Study Groups

3.1 C Floating Point activity report

3.2 C Safety and Security Rules Study Group

Keaton: The Average lifetime of a Technical Specification (TS) is 6 years, but TS17961 has been active for 7 years. The feedback i hear suggests that it is still actively sought. Perhaps we should make it a standard to prolong its life.
Pygott: It needs some work based on feedback. Could this group do this work?
Keaton: The remaining work could be done by the editor. They would need to fold in DRs and update

references (C11->C17).
Seacord: I am happy to work to move it to an international standard. Is there a list of DRs for TS17961?
Keaton: 2 DRs. I will send you a URL.
Straw Poll: Should TS17961 be promoted to an International Standard? 8-0-11. Passes
Action Item: Seacord: Update TS17961 (fold in DRs and update C11 references to C17). Keaton will send Seacord the URL for DR list.

3.3 C Memory Object Model Study Group

4. Future Meetings

4.1 Future Meeting Schedule

Please note that in-person meetings may be converted to virtual meetings due to coronavirus considerations.
30 November - 4 December, 2020 – Virtual, 14:30-18:00 UTC each day
Spring, 2021 – Strasbourg, France (tentative)
4-8 October, 2021 – Minneapolis, Minnesota, US (tentative)
31 January - 4 February, 2022 – Portland, Oregon, US (tentative)

Keaton:
The November meeting starts at a later time than this meeting, as conformant with daylight savings time. Any objections? (none)
The Spring 2021 meeting will now also become 2 virtual meetings, like both 2020 meetings had become. I will propose dates by our November meeting. The Portland meeting (in winter 2022) is still on.

4.2 Future Mailing Deadlines

Note: Please request document numbers by one week before these dates.
Post-Virtual-202010/Pre-Virtual-202011 (one mailing between the two meetings) – 30 October 2020
Post-Virtual-202011 – 18 December 2020

Stoughton: Pure virtual meetings (like this one) run more smoothly than meetings where some but not all are teleconferencing. I would like more 'pure virtual meetings'.
Bhakta: I want less virtual meetings. They are harder, because 2 week-long meetings is easier to schedule than 4 week-long half-day meetings.
Seacord: I prefer virtual half-day meetings. I spend the same amount of time on meetings and reviewing documents.

5. Document Review

Svoboda: Since there was no roll call, I need a way to verify that the participants list is correct. Can I share my screen with my incomplete list over tomorrow's break and people send me completions and updates?
Keaton: Agreed.

Monday

- 5.1 Blower, Adding Fundamental Type for N-bit Integers [N 2534]

  Seacord: This makes the standard complex since no previous integer types are removed.

Myers: The previous document proposed an ABI, which doesn't go into the standard. I want more information about the ABI.
Gustedt: This should be integrated and interact better with bit-fields, fixed-width types, and least-width types.
Hoffner: I have opinions on bit-fields, but need to add that to the document.
Svoboda: Are bit-fields just as powerful as extended integer types, with harder syntax and less predictable behavior? (That is, can two consecutive ExtInts affect each other or have race conditions like two consecutive bit-fields can?
Hoffner: There are less issues than bit-fields because bit-fields have no alignment rules. Making ExtInts regular int types should fix this.
Svoboda: One difficulty is how to write a function that applies to extended integers of any size…should it be a macro, or a C++-style template or…?
Hoffner: We serve different communities, not a one-size-fits-all approach.
Svoboda: I think we would need both approaches: (1) ability to get bit-width at compile time in order to write functions optimized for specific (arbitrary) widths, and (2) ability to get bit-width at runtime for a general (less-optimized) function that operates on all bit-widths.
Bachmann: Is this proposal intended to be mandatory or optional?
Blower: Mandatory
Tydeman: Why not use <stdint.h>?
Hoffner: You can write any of this using standard C but it's horribly messy
Bhakta: This is mandatory, but there is no prior art section. This makes me uncomfortable. Perhaps it should be optional, such as a conditionally normative TS or Annex.
Blower: Hmm, I did not consider that. Some history: we submitted a patch to LLVM, and they suggested proposing it to WG14. We did identify other compilers for FPGAs with similar features.
Hoffner: We can't recall seeing competing platform implementations in other proposals, so we did not consider it.
Keaton: ISO recommends two competing compiler implementations to standardize something. That would argue for optional feature (TS/Annex). I will update the charter when I update the release schedule, which speaks about existing implementations.
Ballman: I like theproposal because the useful types (fixed-width / fast / least) are all optional. We need something truly portable. But if we publish this as a TS, would implementations add it? If so, users would depend on TS behavior and then holler if we discover problems and "fix" the implementation.
Keaton: We have published TS's with the expectation of collecting feedback.
Ballman: Yes. The TS process is imperfect as a beta-testing solution.
Gustedt: <stdint.h> is under-specified because types are non-portable. We should fix that.
Ballman: Agreed, but specifying exact bit-widths is still very useful.
Bhakta: It would be better to add a TS than to add to the standard where there might be mistakes.
Hedquist: A TS is intentionally not ready for standardization.
Straw Poll: Is the committee in favor of adding special extended integer types, using a constant integer parameter that specifies the number of bits that are used to represent the type, as opposed to embedding the parameter into the type name, along the lines of N2534? 16-1-4 clear guidance
Straw Poll: Is the committee in favor of folding the special extended integer types into the C standard integer conversion ranks, along the lines described in N2534? 7-2-10 general guidance to go forward
Straw Poll: Is the committee in favor of adding the exception to the standard integer promotion for the special extended integer types, along the lines described in N2534? 14-0-6 clear guidance
Straw Poll: Is the committee in favor of adding support of integer literals for special extended integer types, using a suffix, along the lines described in N2534? (The values described by the literals are bounded by the intmax types). 13-4-3 clear guidance
Krause: The macro to support small values should be __STR_TO_EXTINT because it is referring to unsigned types

Blower: Agreed, noted.
Svoboda: Could we change to "…along the lines of __STR_TO_EXTINT…"?
Myers: It depends on the suffix.
Blower: Let's just not vote on this one.
Myers: We should allow arbitrarily long strings to specify the literal. Perhaps not any macro at all.

- 5.2 Blower, Allow Duplicate Attributes [N 2557]

  Bhakta: What about duplicates with different parenthesized values? [[deprecated(A)]] vs [[deprecated(B)]]
  Ballman: The same thing as if you have two attribute sequences: they both get stored, and the attribute parser decides what happens.
  Straw Poll: Is the committee in favor of adopting N2557 into C23? 19-0-0

- 5.3 Svoboda, Towards Integer Safety [N 2543]

  Seacord: Overflow is the only thing that can go wrong with operations
  Svoboda: Strictly speaking, yes, but most operations store their result in a type, and that's where truncation & misinterpretation of sign come in.

  Krause:
  Regarding the 7.22.6.3 mk_ckd_type() family, why not use a macro?
  WG14 doesn't care about namespace pollution? Why not use a new header?
  Do we need the functions or are the macros good enough?

  Svoboda: We used to use a new header, but got feedback to use <stdlib.h>. I do not know what wisdom WG14 would recommend…perhaps this should be a straw poll.
  Myers: In normative wording, signed vs. unsigned integer or character types is redundant. "signed or unsigned integer types" is sufficient, and what you want, because it includes all integer types except enums and 'plain char'.
  Svoboda: Agreed
  Myers: You should replace "generic macros" with "family of macros". The Atomic section uses "generic functions and generic macros" and there was a DR asking what that meant.
  Svoboda: Yes. I should avoid "generic" in my proposal….the definition of those terms is a rabbit hole that this proposal should steer clear from.

  Gustedt:
  I like the core proposal.
  But I want support for division & other operations.
  I'm unhappy about "ckd" as prefix. Dunno how to pronounce "ckd". Prefer "checked"
  These items should go into a new header, not <stdlib.h>
  I have lots of questions about this being a complete type. Can a checked type be initialized? What is its default initializer? Is it copy-able? memcpy()-able? Assignable?

  Svoboda:
  There should be a separate proposal for division & other operations. These operations are not supported by GCC or Clang, so it constitutes invention.
  I'm OK with 1- or 2- argument initializers (initialize by value or value + flag).
  A checked type should be assignable using assignment operator or memcpy().
  Reading an uninitialized checked type should work the same as reading an uninitialized unchecked type.

I'll add all this to the document.

Gilding: For reference, a complex type is laid out as an array of 2 elements.
Gilding: You should use the term "type-generic", it's the right term here. Several other sections, including Atomics, use the term and we should make sure they are consistent.
Action Item: Svoboda: Write a proposal to define "type-generic" (for functions and macros) and make it consistent. Consider N2558 as a possible usage.

Tuesday

- 5.3 Svoboda, Towards Integer Safety [N 2543], cont.

  Svoboda: I want more structured discussion. In particular, direction on these questions:
  Svoboda: Should the prefix be "checked" rather than "ckd"?
  Stoughton: I like consistency. What is Clang's practice? (None)
  Gilding: A short prefix used to be the convention, but newer packages like atomic spell it out.
  Bhakta: I still see lots of abbreviations (e.g.: 'thrd'). 'atomic' is the only exception I know. 99% of C standard prefixes are short. ("pri", "flt"). I prefer the short form for consistency.
  Krause: I suggest a straw poll.
  Bhakta: No consensus to add this paper, perhaps we should focus on big-picture issues?
  Svoboda: I assumed those were resolved by latest revision.
  Gilding: Like core + supplemental issue, they could have different prefixes, esp if supplemental is more user-friendly
  Svoboda: I hadn't intended that, but could do it. It would be a shift.
  Bhakta: ISO IEC 10967-1 had a similar question…how did it attack the prefix?
  Straw Poll: Would the committee be in favor of changing the prefix to "checked", rather than leaving it as "ckd" in N2543. 8-6-9 divided.
  Svoboda: Should the items defined in this proposal go into their own new header file or into <stdlib.h>?
  Klieber: Adding things to <stdlib.h> can cause clashes with code that includes it. But that is not a problem for a new header.
  Gilding: Regarding creating a new header, this proposal seems like a good candidate because it's very self-contained in its functionality and focused in its purpose. Unlike say `offsetof` or `tovoidptr` (proposed, n2522), it's not just a utility designed to be used as a helper for a bigger operation. It's a self-contained and complete API solution (operators like `div` aside - it is complete in what it intends to provide) that aims to address a whole class of operations.
  Krause: <math.h> is for floating-point, but this is integers. Maybe <ckdint.h>.
  Gustedt: I prefer headers that start with "std". Perhaps <stdckd.h>.
  Bhakta: I agree with Krause. I prefer a new header, <stdckd.h>.
  Banks: This shouldn't go in existing headers, as Klieber cited. I sympathize with "std".
  Gilding: Perhaps imitating GCC/Clang's use of <stdlib.h> is risky, because our behavior will be subtly different.
  Svoboda: Is anyone in favor of <stdlib.h>? (No) What about <stdckdint.h>?
  Krause: The header name and prefix should be the same. That's fine if you use "ckd" in prefix.
  Banks: C17 already has <stdnoreturn.h>. I agree that the header should match the prefix.
  Keaton: We have used up 30 minutes today; let's table the rest of this discussion to later.

- 5.4 Seacord, Specific bit-width conversion specifiers [N 2511]

  Tydeman: How about adding an * for the size?

Seacord: I hadn't considered that.
Krause: This wouldn't work with ExtInts because of promotion.
Seacord: OK
Gustedt: I am opposed to adding *, because the dynamic-ity doesn't make sense here.
Myers: I also think * does not make sense. There is no need to determine width at runtime. I think there are no conflicts with N2562, but we should verify.
Bachmann: this duplicates some types in <inttypes.h>, but it is still desirable, because it increases readability.
Bhakta: Section on existing practice not compelling. But we should only standardize existing practice. Is there other existing practice?
Stoughton: I like the principle, but want to see existing practice before standardizing this. I wanted to see more about b=binary format specifier.
Seacord: That part wasn't normative; it was our rationale for using 'w' instead of 'b'.
Hoffner: Why can't ExtInt fit into this? Also, will this be optional (as ExtInt would be?)
Seacord: I hadn't proposed this as an optional feature.
Keaton: Fixed-width types are optional.
Seacord: Would using "precision" be a better term than "width"?
Gusted: No, the printf() needs the size of data, not the number of bits.
Keaton: You could always elect to put it in a TS instead of the standard. That might alter a vote.
Seacord: I hadn't considered a TS for something this small.
Keaton: We once had a character TS that was 5 pages.
Straw Poll: Does the committee wish to include N2511 into C23 as is? 7-7-8 not sufficient consensus for standardization
Straw Poll: Does the committee wish to include something along the lines of N2511 into C23? 16-2-5 clear direction
Bhakta: Is it C2x or C23?
Keaton: August 2023 is the deadline. It is unlikely that we can publish earlier.
Stoughton: My "abstain" vote would become a yes if we had existing practice.
Bhakta: Given the last vote, do we not care about existing practice?
Krause: Given how small this is, I thought it was still good enough.
Keaton: A small, very targeted feature, could depend on related features for existing practice.

- 5.5 Seacord, Defer Mechanism for C [N 2542]

  Gilding: Every platform can quickly support this, because this is a library and compilers can re-order defer statements.
  Keaton: Given a break in 10 minutes, let's limit comments now to those that address Seacord's questions?
  Straw Poll: Does the committee wish a defer statement, along the lines of N2542, in C23? 15-3-6 (clear direction)
  Straw Poll: Should defer statements be static, rather than dynamic, along the lines of N2542? 12-8-3
  Straw Poll: Should object values be captured when the defer statement is encountered, rather than when the defer statement is executed, along the lines of N2542? 8-9-6 (no sentiment)
  Straw Poll: Should defer-ing be determined by scope along the lines of N2542? 10-4-7
  Straw Poll: Does the committee wish a panic / recover mechanism, along the lines of N2542, in C23? 4-11-9
  Keaton: Tabled til later

- 5.6 Banks, Return type of <ctype.h> and <wctype.h> character classification functions [N 2541]

Tabled

- 5.7 Stoughton, C23 Liaison: Rebuttal to N2526 [N 2565]

  Gustedt: I am horrified by the naming choice (prepending 'c' in functions). Or creating duplicate versions of these functions. POSIX could deprecate the non-const functions for several years before C23 comes out.
  Stoughton: We have to support code going back to 1990. It is unlikely that the Austin Group would make these changes.
  Clare: When the C standard was introduced (1989), they added const so as not to break existing code.
  Gilding: As getenv() allows the return value to be mutable, isn't making getenv() const actually over-specifying it?
  Stoughton: The majority of code cases never modifies the string returned by getenv(), but there are a few scenarios where it can be done safely.
  Bhakta: C++ liaison issues are high-priority. POSIX and other WGs should also have high impact.
  Ballman: Thanks to the Austin Group for bringing this up. Our charter says we should not break code.
  Wiedijk: How can this make compilation fail?
  Stoughton: A shell implements these functions under the hood, and expects to be able to modify getenv()'s output. Bash would not compile at all.
  Seacord: We need to have a broad evolution for things. We deprecated gets() in C99 and removed it in C11…can we do something like that here? Also, we are proposing new names for these functions, but if we create a new set of functions, we would want to give more thought to them.
  Banks: I am disturbed with creating a new set of functions. I don't see the problem here, but don't like the solution.
  Uecker: I would like to have the const on the functions. Is there a way to propose a schedule that plans the change for the future, which gives POSIX and us time to "soften the blow"?
  Keaton: Let's discuss the next paper and decide on both together.

- 5.8 Krause, use const for data from the library that shall not be modified [N 2526]

  Gustedt: musl is not a good example; it is a C library.
  Gilding: Regarding deprecating assignment to non-const pointer, many groups have recommendations not to use non-const pointers.
  Stoughton: We can't remove these functions in a future standard, because that breaks our contract, and API. I'm in favor of adding recommendations to treat these return values as const, but I am against changing the APIs. POSIX has a clause saying deviations from ISO C are unintentional and POSIX and ISO C disagree, ISO C overrides POSIX.
  Gustedt: It sounds like POSIX does allow getenv()'s return value to be modified. I would like to see a paper going into that, because perhaps that should be introduced to C23. I don't think this would apply to the other three functions.
  Keaton: Stoughton, could we revert getenv() but not the others?
  Stoughton: I am completely opposed to any future direction that says "we will add const here".
  Krause: I am opposed to adding new functions. The people who would use them don't need them.
  Gilding:If POSIX has extended getenv() by defining some cases that would be undefined behavior in ISO C. This is permissible because POSIX is defining this undefined behavior.
  Straw Poll: Should the committee withdraw our previous vote that was in favor of N2526? 15-3-4 clear direction
  Straw Poll: Should the committee invent new functions to improve safety and security in these areas, along the lines of N2565? 5-12-4 clear direction
  Gilding: Should the committee poll about recommended practice?

Bhakta: I would want to see a paper before I vote on anything.
Banks: How about a straw poll about writing a paper?
Keaton: Take an action item to write the paper because people are voting already.
Action Item: Banks: Write a paper to strengthen the recommended practice for using const in these areas, along the lines of N2565.

- 5.9 Stoughton, C23 Liaison: Removal of Deprecated Functions [N 2566]

  Gustedt: It is not necessary to implement functions using sprintf(). Removing it eliminated 10K code size. Recommending that we add a deprecated attribute to a new function will break lots of code that uses it (and that treats warnings as errors)
  Stoughton: Yes, this is a bit inconsistent with my previous stance. But I still stand by it.
  Bhakta: Adding in "deprecated" seems contrasting to what we had before. I dislike both proposals, but proposal 2 is better than 1.
  Stoughton: I'm willing to remove the "deprecated" attribute from the proposal.
  Gilding: I don't mind submitting two papers with different solutions. I do prefer calling them "deprecated".
  Straw Poll: Should the committee adopt the proposed version 1 of N2566 into C23? 2-10-9 No consensus
  Straw Poll: Should the committee adopt the proposed version 2 of N2566 into C23? 10-4-5 clear direction
  Straw Poll: Should the committee revert the asctime() and ctime() functions to the wording as in C17? 4-11-5 clear sentiment not to

Wednesday

- 5.10 Sewell, A Provenance-aware Memory Object Model for C [N 2577]

  Sewell: N2378 is a slide tutorial about the same material.
  Bhakta: But we can't talk about how we will vote, right? (Right!)
  Sewell: We are not specifying anything about when pointer lifetime ends. That is intended for a subsequent proposal
  Wiedijk: This is Very specific. Current compilers must change to comply, right? Was that intentional or do I misunderstand?
  Sewell: Compilers are not always consistent about provenance. So we did intend this as a consistent change. We are unaware of required changes with major impact. But this might affect some optimizations.
  Keaton: Some implementations thought they might need to change. So they requested this proposed as a TS, mainly to collect & organize the new specs.
  Wiedijk: Is this TS "complete" or an ongoing process?
  Sewell: Unknown. We might receive improvements from committee. Extensions or bug fixes.
  Gustedt: What is here is stable; it hasn't changed for over a year.
  Wiedijk: We don't want proliferation of incompatible C dialects. Is this TS a "pick & choose" of several items?
  Sewell: No, this is one model that we want universally adopted.
  Bhakta: Are there any changes to the test suite license?
  Sewell: We provided these under a BSD license, but we can take it offline if that is problematic.
  Bhakta: Does WG21 have plans to take this into their standard or as a TS?
  Sewell: We have not pushed on WG21, although that would be good to do.
  Keaton: We are out of time.

Blower: For the process, we need five NBs to approve.
Keaton: We can't discuss that in this meeting. Do you want time at our next meeting? Perhaps 90 minutes?
Sewell: We will go over the technical details and examples. We can work this out offline with Keaton.

- 5.11 Tydeman, Missing DEC_EVAL_METHOD [N 2546]

  Myers: Should there be a change to section 6.5.6.2 to also mention DEC_EVAL_METHOD?
  Tydeman: We'll look into it.
  Straw Poll: Does the committee wish to adopt N2546 into C23? 20-0-1 adopted

- 5.12 Tydeman, Missing 'const' in decimal getpayload functions [N 2547]

  Bhakta: This is editorial; we should just put this in, since we already voted on it. It wasn't added because we lacked an editor at the time.
  Meneide: I'm fine with adding it to the editorial queue.

- 5.13 Tydeman, intmax_t and math functions [N 2548]

  Svoboda: These functions converted from float to int, they now only convert float to float, right?
  Tydeman: Yes. They also truncate the decimal part and they report conversion errors.
  Straw Poll: Does the committee wish to adopt N2548 into C23? 17-0-6 adopted
  Wiedijk: This will generate criticism on forums. because people will get confused.
  Keaton: Yes, but only if you see the API and not the associated text.

- 5.14 Tydeman, Range errors and math functions [N 2564]

  Tydeman: No vote necessary; this paper summarizes what was already accepted.

- 5.15 Thomas, F.3 editorial cleanup for rounding macros [N 2552]

  Bhakta: No vote is necessary, but WG14 can object to these planned editorial changes.
  Krause: Can we discuss Banks' paper?
  Keaton: We will first discuss Wednesday's & Thursday's papers. If there is more time, we can discuss remaining papers, with Banks' paper as the first.

- 5.16 Thomas, C23 proposal - Annex B with prototype forms [N 2558]

  Gilding: Love!
  Gustedt: This is meant to summarize what is there, but it is not automatically generated. We should make this change to the standard itself.
  Bhakta: We did not consider auto-generation.
  Gustedt: We should apply this level of readability to the entire standard!
  Bhakta: That is more work than we had planned to do. It could be done later.
  Blower: Is the #pragma (FP_CONTRACT) a typo?
  Bhakta: That should not be there…it looks like a typo to me.
  Myers: There are some mistakes in the list of interfaces here. For example, "next_toward()" has no float type.
  Keaton: The pragma is in earlier versions of the standard; it was not introduced in this proposal. The surrounding text is changing.
  Ballman: MS is an acronym elsewhere that means other things than "macro suffix", because "M" is used as a 1-char symbol. Perhaps a better character is in order?

Bhakta: "M" & "N" was used in space 3, that's why we adopted them.
Tydeman: next_toward() has long double type too. Annex D is missing several things, so the generation process can not be totally automatic.
Svoboda: This looks like a case of the "generic functions/macros" that I ran into (in N2543). I'll consider this when writing a paper.
Gustedt: I am opposed to an Annex B that doesn't exactly describe what is in the Standard. Inconsistency with the standard reduces usability.
Myers: Perhaps these standard types (and other type categories should be defined in the proper standard?
Tydeman: I am not sure how you would handle the ifdefs for these types in the rest of the standard.
Myers: If decimal floats go into different headers, that would complicate things.
Gustedt: Do cleanup first.
Stoughton: I want "Annex B" in the vote.
Bhakta: But if our editor must do lots of extra work then forget it.
Svoboda: The straw poll gives you direction. But you can independently take direction from our editor. So leave editorial veto out of the straw poll.
Bhakta: This paper is only for Annex B for now. Changes to others can be addressed in a separate straw poll.
Straw Poll: Does the committee wish to adopt something, along the lines of N2558, in Annex B, into C23? 11-4-8 direction to continue.
Straw Poll: Does the committee wish to adopt something, along the lines of N2558, in the main body of the standard, into C23? (That is reworking the main body to follow type parameterization) 11-2-8 direction to continue

- 5.17 Thomas, C23 proposal - Update to IEC 60559 2020 [N 2559]

  Myers: What's the ISO convention for referencing multiple versions of another standard?
  Bhakta: You think we should keep references to original versions where C functions came from?
  Sewell: What benefit do we get from keeping those references?
  Myers: Historical purposes.
  Straw Poll: Does the committee wish to adopt N2559 into C23? 15-0-5 adopted

- 5.18 Thomas, C23 proposal - FP hex formatting precision [N 2560]

  Myers: The footnote has become very long. The example should go into an Examples section.
  Bhakta: That was discussed in CFP, and our vote yielded a 50-50 split. We couldn't figure out where to put the example.
  Gilding: We should keep the implementation-defined behavior example close to its definition.
  Straw Poll: Does the committee wish to adopt N2560 into C23? 10-1-9 adopted
  Bhakta: Why the abstentions? Do you care about footnote vs. example or is this not a worthwhile change?
  Gilding: i am Neutral. I just did not know enough about this field to vote.

- 5.19 Thomas, C23 proposal - TS 18661-3 annex update [N 2561]

  Myers: N2579 is newer so should we be using that?
  Bhakta: In N2579, we updated references, changed dates and adopted it to latest C23 working draft. N2579 was not ready for the mailing deadline.
  Myers: Further updates were announced on the reflector as needed to N2579.
  Meneide: The annex lists places where I have to look out for this.
  Myers: What changes are there for float-to-float macros?

Ballman: With attributes we use the meeting dates.
Meneide: If you expect people to keep track of this in real-time, you want meeting dates.
Bhakta: I want to vote on this as is. We can submit future changes on top of this.
Keaton: Agreed.
Myers: I don't want a draft with internally inconsistent state in the standard. Could we vote on a paper submitted Friday?
Bhakta: I'm not sure that we can get those changes done by Friday.
Keaton: OK, let's put this paper onto our queue of papers to revisit. If you can get those changes made when the queue is ready we'll discuss it. Tabled.
Bhakta: Does anyone object to discussing this paper if we update it? (Yes)
Seacord: We are not supposed to review modified papers not submitted 30 days.
Keaton: That is not an official rule…we've done it many times, but you can still object.
Straw Poll: Does the committee wish to incorporate N2561 in to C23, anticipating future changes will still need to be done?
10-3-7 adopted

Paper was subsequently made available as Part 3 of TS annex X: (http://www.open-std.org/jtc1/sc22/wg14/www/docs/tmp/cfp3-annex-20201014.pdf) is available

- 5.20 Thomas, C23 proposal - Feature and want macros for Annex F functions [N 2570]

  Krause: Don't these WANT macros pollute the namespace?
  Bhakta: Not talking about all FP functions. The WANT macros are not sufficient.
  Krause: I prefer that WANT stays. I want it to cover more functions, so more identifiers become available.
  Straw Poll: Does the committee wish to address the issue raised in N2570? 12-0-6 clear direction
  Straw Poll: Does the committee wish to keep the WANT macro as described in N2570? 4-6-8 not clear direction
  Straw Poll: Does the committee wish two frames to describe these functions in N2570? 4-2-12 underwhelming
  Straw Poll: Does the committee wish the single-frame solution for both binary and decimal, as described in the second example in N2570? 6-0-11 more consistent support
  Straw Poll: Does the committee wish to adopt, as is, the single-frame solution for both binary and decimal as described in the second example in N2570? 6-0-12
  Keaton: So many abstentions make this straw poll difficult to judge.
  ???: I abstained from Lack of domain knowledge
  Banks: So did I. Abstain was my way of not voting against it.
  Keaton: It sounds like we have sufficient consensus to adopt the single-frame solution.

Thursday

- 5.3 Svoboda, Towards Integer Safety [N 2543], cont.

  Svoboda: Should the "constructor" mk_ckd_* family of functions be a single macro rather than a function family?
  Gilding: I want both macros and functions. Please don't require the straw poll to be exclusive.
  Gustedt: Agreed. Also they should be called ckd_mk… not mk_ckd.
  Uecker: I like both. There should be a high-level plan or strategy for families of type-generic macros.
  Svoboda: If they remain functions, the mk_ckd_* functions should be [[nodiscard]].
  Gilding: Yes, need high-level plan.

Bhakta: CFP had to go through the type-generic problem many times. It is Long, complicated and unintuitive.
Krause: This paper is a drop in the bucket with regard to type-genericity
Gustedt: Initialization of these types could help. The mk_ckd macro might be superfluous if we get initialization right.
Uecker: Functions vs. function-like macros are hard to distinguish. Perhaps we should make them distinguishable. Generic macro families should be distinguishable.
Svoboda: Functions vs. function-like macros are hard to distinguish by design. getc() for example. This is not a problem for type-generic macros
Krause: Making a naming convention may be beneficial. I have never taken the address of a standard library function, so don't care about such beasts.
Uecker: We need to make sure we distinguish type-generic macros from functions in general.
Svoboda: Macros are conventionally uppercase, functions are lowercase (but I need to verify that my proposal complies with this)
Gustedt: If we distinguish type-generic things we should use another convention (besides capitalization)
Ballman: Any function can be implemented as a macro. The salient property is type-generic.
Bhakta: How much time do we have?
Keaton: We can spend another 30 min on this paper.
Bhakta: C is not a generic language; we have added generic programming only in cases where we thought users would not care. But users should know what is going on when they see what looks like a function call. This paper is not as complex as FP but not trivial either. I would want different names between functions & macros; I don't care about capitalization. I would want to see different tokens for function vs. macro.
Svoboda: Does anyone prefer having macros but not functions for mk_ckd?
Krause: Functions contribute to namespace pollution
Bhakta: Is this proposal for the library only, or is it language-based?
Svoboda: What's your definition of library-only?
Bhakta: library-only means "not part of C library", that is, it is a separate library. We could make this into a TS or international standard.
Keaton: Either of those could work. If we wanted to experiment, we make a TS. If we wanted it to be permanently separate from standard and we were happy with the specification, we make an IS.
Svoboda: I hadn't considered a TS or IS. Something integrated with the platform would be faster than a separate library.
Ballman: Type-generic macros are not necessarily a good thing. C is not that good for type-generic programming. It lacks anything like C++'s "typeof" operator. It also lacks format specifier support for generic types.
Bhakta: The proposal is correct in doing library-only but could be part of the compiler. This does not preclude TS or IS. TS would be more suitable. Clang and GCC count more like 1.5 compilers than 2 compilers when considering existing practice, because Clang tries to copy GCC.
Keaton: A TS is great because after we get experience, we can merge it into C standard or promote it to an IS.
Stoughton: I disagree that a library can't have good performance if it is tied to a platform.
Gilding: I assume these types do not promote to one another, do they?
Svoboda: The proposal assumes no implicit promotions or conversions. That could be a separate proposal.
Gilding: Perhaps you should add some wording to that part.
Krause: The exact flag might be better as an 'inexact' flag, because the convention is that no errors means 0 or False.
Svoboda: I had followed GCC's convention, but I'm receptive to changing the exact flag to inexact.

Klieber: Implementing inside the compiler allows for inlining. Implementing in a separate library makes a non-inline-able function call, which is slower.
Bhakta: No, library functions can be inlined.
Klieber: You can't jump to elsewhere in a calling function inside a library function, but you can in a compiler-supported function.
Bhakta: No, it is possible and easy to do.
Krause: A simple compiler can inline library functions. so they are no less efficient than compiler builtins.
Straw Poll: Does the committee want a function family for the mk_ckd services in N2543?? 12-1-8 clear direction
Straw Poll: Does the committee want a type-generic macro for the mk_ckd services in N2543?? 8-3-10 favor, but more undecided

- 5.6 Banks, Return type of <ctype.h> and <wctype.h> character classification functions [N 2541]

  Seacord: Would Stoughton and The Austin Group complain if we changed APIs here?
  Stoughton: This would be less controversial than adding const.
  Gustedt: We need a study to see if this breaks anything.
  Bhakta: Some implementations provide (nonzero) classification of characters.
  Ballman: I am worried about breaking stuff, especially user code. We could break generic selections. Are there any performance concerns?
  Gilding: I am only concerned if the function return value gets assigned to an lvalue. Such code would be intentionally broken by this proposal.
  Ballman: What's the value-add here, besides better function signatures?
  Gilding: The paper shows two examples of user errors, that occur under current conditions.
  Svoboda: The hazard is if anyone depends on the integer value (besides being nonzero). Jens brought this up last hour with the ability to use the exact flag to indicate error details. Has that ever happened in history? That is, has a boolean 0/nonzero return value ever evolved to integer? I think errno qualifies… anything else?
  Keaton: Bhakta suggested a table of bit-masked values. But that was not public.
  Stoughton: I think POSIX has some; I am checking.
  Bachmann: API breakage: The AL register gets the boolean return value, the EAX register gets an integer value.
  Bachmann: Many years ago these functions returned char or bool. Users complained that an int was more expensive.
  Tydeman: Should the committee look at other functions that returned bool or int? If we don't adopt this, can we recommend that these return values be cast to bool?
  Gilding: We did not find any other easy targets, so we just focused on char types.
  Stoughton: There are is*() functions that could be adopted. I am turning against this proposal. I find functions that return an index into a table or 0. Also, POSIX has added more functions in the same (ctype) family, which would need to be changed.
  Gilding: Does POSIX communicate more info with nonzero values?
  Stoughton: Implementation-defined
  Seacord: Changing the ABI seems to be extremely difficult. It's easier to introduce functions with a slightly different name. Could we do something like bool is_alpha(char c)? In contrast deprecating gets() was fast and easy.
  Gilding: const getenv() and similar functions was discussed earlier.
  Seacord: Could we just have an include file that overwrites the old functions with the newer functions?
  Banks: That would be invisible to anybody, due to anyone distinguishing the nonzero result.
  Ojeda: Recommending a cast to bool would be easier to add than accepting this paper. And we could

require a warning to see if these functions are being misused.
Bhakta: Conversion to bool may be a performance penalty (sometimes but not always eliminate-able). The performance penalty can be nontrivial.
Svoboda: gets() wasn't replaced with anything so it incurred no breakage. I'll second Tydeman's suggestion that if we nix this paper, we should add some clarifying wording. I'll suggest we allow implementation-defined particular non-zero values.
Tydeman: fenv() function got changed in C99/TC1.

- 5.5 Seacord, Defer Mechanism for C [N 2542]

Gilding: If we request capture-by-name we get capture-by-value for free.
Seacord: Agreed
Krause: Defer will be difficult and come with some runtime cost.
Seacord: The reference implementation binds these questions to a specific implementation.
Keaton: We should interpret those first votes knowing they were rushed before there was any discussion.
Gustedt: We definitely must be more informed on static vs. dynamic. In Go, defer is bound to a function body.
Seacord: I'm comfortable with moving away from how Go implements defer. We can possibly move closer to C++, due to compatibility with that committee.
Bhakta: Implementations were in C but were in common with a C++ compiler.
Gustedt: These were specializations. The implementation worked with setjmp()/longjmp().
Seacord: We could do a TS, it's up to the overall committee.
Gilding: I would side with C++ expectations rather than Go expectations.
Ojeda: Consider whether we want scope-like approaches. Also I would like an appendix about Rust.
Seacord: The paper was developed as a Google doc with multiple authors…Ojedo, please join us and add a Rust appendix. I'll share the link.
Ballman: C++ solves these problems in different ways already. C++ uses constructors, destructors, and RAII. and panic/recover == try/catch. We should try to solve problems in a way compatible with C++. Capture might be more interesting for C users. You can always copy errno and defer it. But people will forget that. So there is lots of value capturing things by default when encountering defer, rather than executing deferred statements.
Seacord: Panic/recover is stupider than try/catch/throw, because there is no exception hierarchy, just an integer value.
Gustedt: We could augment the implementation to make panic/recover compatible with try/catch. This would allow us to call C++ function from C.
Hoffner: This contains new keywords. If it becomes a TS, does the keyword go in the standard?
Keaton: There are no problems reserving keywords in a TS.
Uecker: We should constrain things in the initial proposal, and open it up later.
Seacord: Our last straw poll indicated a preference to not do panic/recover now. If we make a TS, we try to gain experience with things like panic/recover.
Bhakta: CFP made a TS, and from that brought in all required stuff but only some recommended stuff.
Keaton: By the way, Microsoft is now up-to-date with the current C standard. I mentioned this paper to Herb Sutter, who said paper is interesting.
Bhakta: I am happy with this going forward into a TS, but not happy with it going into C23.
Wiedijk: Traditionally the C return statement was not extensible, and this would change that. I still really like defer.
Straw Poll: Does the committee wish a defer statement, along the lines of N2542, in some way? 12-3-6 clear direction
Straw Poll: Does the committee wish a defer statement, along the lines of N2542, as a TS? 14-1-7 clear

direction
Svoboda: The next question is: Should defer statements be static, rather than dynamic, along the lines of N2542? 12-8-3
Gustedt: I would rather like the question to be conditional. I really care about this case about defer being in a conditional.
Seacord: A static conditional defer statement could have a compile-time bit to indicate if it should be executed. But a static defer statement in a for loop would be executed either 0 or 1 times.
Ojeda: Scope matters. There are combinations of "static" or "dynamic" that makes sense.
Krause: Even if defer is not in any conditional statement, it could be conditional, perhaps by being after a return statement inside a conditional.
Gustedt: The point is it should be easy to program with.
Seacord: A defer in an if could limit the defer's scope to the scope if the if's then-block.
Meneide: It would be better to present this as a table, with columns describing design choice, and each row is a choice of decisions. That would make it clear what we would be voting for.
Gustedt: This is the only option for which the reference implementation does not work in the library. It would require compiler magic.
Straw Poll: Should defer-ing be determined by scope? 4-8-9 no real sentiment
Ballman: Perhaps we should ask about an explicit guard keyword?
Wiedijk: If I have many defers, can I have just one guard?
Seacord: You can have multiple defer statements inside a single guarded block.
Straw Poll: Should defer-ing be determined by an explicit guarded keyword? 4-4-11 topic needs more exploring
Straw Poll: Does the committee want to allow dynamic execution of the defer statement, rather than static? 4-13-4 clear direction away from dynamic.

- 5.21 Verwer, Request for definition of the term "multi-threaded program" [N 2555]

  Gustedt: This is undefined behavior because platforms can further define the behavior, as POSIX does.
  Krause: There are three possibilities: (1) A program has multiple threads at any point in its lifetime (2) A program had only one thread until now, and (3) The program had multiple threads in the past but only one now.
  Tydeman: My impression is option (1).
  Uecker: We could rephrase this in terms of multi-threaded execution. That is: "Use of this function during multi-threaded execution is undefined behavior".
  Ballman: The definition is contextual. So let's fix the signals wording rather than define "multi-threaded program".
  Bhakta: Section 5.1.2.4 speaks of "multi-threaded execution" (not "multi-threaded programs"). I suggest option 1.
  Keaton: It seems like there is sentiment to mirror POSIX.
  Straw Poll: Do we wish option 1 with regard to specifying multi-threaded execution? 20-0-2 clear direction that our response to this request be option 1.

- Use of Chat

  Keaton: The chat function is useful. But chat was scrolling faster than people were talking. This is not respectful to the speakers. Chat should not replace conversation.
  Bhakta: Seconded. Chat should be supplementary.
  Seacord: I would prefer the whole discussion to be in the chat. This bypasses accents, audio problems, and an inadequate record.
  Svoboda: First, I would like to save the chat, it contains useful info that is hard to speak, such as URLs.

But the reflector has all the advantages of chat without the disadvantage of disrespecting the speaker.
Keaton: Chat is great as a backup when verbal response does not work.
Gustedt: The chat is also gone for good if you get disconnected.

Friday

- 5.22 Wunsch, Proposal for C23 [N 2549]

  Bachmann: This was in the provisional for C99. it was rejected because of no prior art, but now that is fixed. The proposal is useful but only if we have the means to group these numbers. We could allow underscores in binary literals to allow some form of grouping.
  Wunsch: A form of grouping would be useful. Why not add them to other number literals?
  Svoboda: How compatible with C++ is this?
  Wunsch: I am not familiar with C++, but we aimed to be compatible.
  Ballman: This is fully compatible with C++. I'm happy to write a subsequent paper that supports digit separators. C++ uses single quotes as separator. My paper would cover all numeric bases, so it would be orthogonal to your paper.
  Stoughton: I want to see binary I/O (that is, for printf() and scanf()) for binary numbers.
  Wunsch: I never had such a request. If WG14 would find it useful, I'll write a separate paper.
  Gilding: We commonly have underscores in the middle of numbers. They get handled by the C preprocessor.
  Gustedt: I think editors are capable of fixing the missing table.
  Krause: We can vote on this paper as is. Both separators and scanf() are separate issues.
  Seacord: The Intel N-bit integer paper might also be a good use case for these.
  Meneide: I can fix the table header, so this is editorial.
  Straw Poll: Does the committee wish to adopt N2549 into C23 with the editorial change to update the table in Paragraph 5? 23-0-0 adopted
  Action Item: Ballman: Write a paper on adding separators to C integer literals, along the lines of C++
  Action Item: Wunsch: Write a paper on adding binary I/O for printf() and scanf().
  Svoboda: Perhaps a straw poll to see if the committee would accept binary I/O?
  Myers: If you are doing I/O you should also consider functions like strtol()
  Gilding: We have configuration options to understand those format specifiers.
  Bhakta: Please don't spend committee time discussing that potential paper.
  Wunsch: Gilding, do you know what formatting has been used?
  Gilding: I don't know offhand. But I can send you what I find.

- 5.23 Ballman, Unclear type relationship between a format specifier and its argument (updates N2483) [N 2562]

  Straw Poll: Does the committee wish to adopt N2562 into C23 as is? 22-0-1 adopted

- 5.24 Ballman, Character encoding of diagnostic text [N 2563]

  Bhakta: A diagnostic character set is a 3rd character set, distinct from source and execution. So I'm in favor of it.
  Svoboda: Will C++ produce something more substantial? How permanent is their current advice?
  Ballman: It is possible WG21 will come up with something different. But there are no current expectations.
  Gilding: Issue of 3 or more character sets is a real problem in practice. Often in Japanese encodings with English language systems.

Straw Poll: Does the committee wish to adopt N2563 into C23 as is? 23-0-0 adopted
Bhakta: Ballman, you are promoting a similar paper to WG21, right?
Action Item: Ballman: Write a version of N2563 for C++.

- 5.25 Working draft updates
- Meneide, C23 Working Draft [N 2573]
- Meneide, C23 Working Draft with diff-marks from N 2478 [N 2583]
- Meneide, C23 Editor's Report [N 2574]

Bhakta: CFP group found that papers going in work well, and we are getting things done a lot faster. Good job!
Keaton: Everyone please look at the diff-marks draft to make sure changes look good.

6. Clarification Requests

The previous queue of clarification requests has been processed.

7. Other Business

The following papers had been deferred to this meeting.

7.1 Tydeman, snprintf() [N 2571]

Bhakta: Same objections as last time. I don't see the problem here.
Svoboda: Didn't we discuss this ambiguity in snprintf() before?
Bhakta: When we last discussed this, Martin Sebor and Larry Wagonner and I didn't see any issue with the text. At that time there was no confusion on what snprintf() was intended to do.
Seacord: The sentence in the standard was incorrect; it needs "both"; e.g.: "the value is **both** non-negative and less than n". That is entirely an editorial manner.
Gilding: I am not seeing the ambiguity, but there could still be one. Doesn't harm the meaning of the sentence.
Bhakta: I disagree that we should change something if someone sees an ambiguity. Also "nothing is written" is nonsense, perhaps a 0-length string is written.
Tydeman: You are asking for nothing to be written.
Bhakta: No, I are asking for a 0-length string to be written. Not nothing.
Krause: "A null character is written after at the end of the characters actually written…" But if no characters are actually written, then what? This is bad wording.
Gilding: That doesn't seem contradictory to me. 'Characters written' could include nothing.
Straw Poll: Does the committee wish to make any changes, along the lines suggested in N2571? 11-3-7 We do need other votes
13-3-6
Straw Poll: Does the committee want to editorially add the word "both" before "non-negative and less than n" in 7.21.6.5p3: 13-3-6 adopted
Straw Poll: Does the committee wish to make the first suggested change from N2571 to C23? 6-7-6 no consensus
Ballman: Since this is an editorial change, do we need another paper?
Keaton: Meneide, please note this editorial change.

7.2 What to do about YYYYMM?

Keaton: This is an issue that spans multiple papers
Bhakta: We could add a meeting number as Ballman suggested. It makes sense that each published draft should have a YYYYMM date number.
Ballman: Doesn't need to be consistent for all proposals. We assume people are tracking the standard and using features as they are added. Either way we go, well-written code should be based on macro values.
Myers: We have added various STDC macros, using YYYYMML. But we haven't done it consistently to all features.
Bhakta: Most of our external community does not follow committee meetings, just draft standards. If we want an early jump on a new feature, we use a preprocessing directive whose macro date is greater than the last publishing date.
Ballman: That is what I see as well: last published standard + 1 month. The default should be meetings because that gives us the most resolution.
Tydeman: I use meeting date in my testing.
Keaton: If we substitute actual dates for each meeting, what placeholder do we use to substitute another time?
Ballman: We can use a LaTeX marker for the editor to search for.
Bhakta: What most users see is publication date + 1 month. Back around C99, date changes were confusing. Multiple date macros would cause more churn. They were publication dates not meeting numbers.
Keaton: If you submit proposal with YYYYMM. We can decide how to handle past proposals that don't specify dates.
Keaton: Future proposals that have YYYYMM macros should specify whether the date set on each version of the proposal or publication of the standard.

8. Resolutions and Decisions reached

8.1 Review of Decisions Reached

Should TS 17961 be promoted to an International Standard? 8-0-11. Passes
Is the committee in favor of adding special extended integer types, using a constant integer parameter that specifies the number of bits that are used to represent the type, as opposed to embedding the parameter into the type name, along the lines of N2534? 16-1-4 clear guidance
Is the committee in favor of folding the special extended integer types into the C standard integer conversion ranks, along the lines described in N2534? 7-2-10 general guidance to go forward
Is the committee in favor of adding the exception to the standard integer promotion for the special extended integer types, along the lines described in N2534? 14-0-6 clear guidance
Is the committee in favor of adding support of integer literals for special extended integer types, using a suffix, along the lines described in N2534? (The values described by the literals are bounded by the intmax types). 13-4-3 clear guidance
Is the committee in favor of adopting N2557 into C23? 19-0-0
Would the committee be in favor of changing the prefix to "checked", rather than leaving it as "ckd" in N2543. 8-6-9 divided.
Does the committee wish to include N2511 into C23 as is? 7-7-8 not sufficient consensus for standardization
Does the committee wish to include something along the lines of N2511 into C23? 16-2-5 clear direction
Does the committee wish a defer statement, along the lines of N2542, in C23? 15-3-6 (clear direction)
Should defer statements be static, rather than dynamic, along the lines of N2542? 12-8-3
Should object values be captured when the defer statement is encountered, rather than when the defer statement is executed, along the lines of N2542? 8-9-6 (no sentiment)
Should defer-ing be determined by scope along the lines of N2542? 10-4-7
Does the committee wish a panic / recover mechanism, along the lines of N2542, in C23? 4-11-9
Should the committee withdraw our previous vote that was in favor of N2526? 15-3-4 clear direction

Should the committee invent new functions to improve safety and security in these areas, along the lines of N2565? 5-12-4 clear direction
Should the committee adopt the proposed version 1 of N2566 into C23? 2-10-9 No consensus
Should the committee adopt the proposed version 2 of N2566 into C23? 10-4-5 clear direction
Should the committee revert the asctime() and ctime() functions to the wording as in C17? 4-11-5 clear sentiment not to
Does the committee wish to adopt N2546 into C23? 20-0-1 adopted
Does the committee wish to adopt N2548 into C23? 17-0-6 adopted
Does the committee wish to adopt something, along the lines of N2558, in Annex B, into C23? 11-4-8 direction to continue.
Does the committee wish to adopt something, along the lines of N2558, in the main body of the standard, into C23? (That is reworking the main body to follow type parameterization) 11-2-8 direction to continue
Does the committee wish to adopt N2559 into C23? 15-0-5 adopted
Does the committee wish to adopt N2560 into C23? 10-1-9 adopted
Does the committee wish to incorporate N2561 in to C23, anticipating future changes will still need to be done?
Does the committee wish to address the issue raised in N2570? 12-0-6 clear direction
Does the committee wish to keep the WANT macro as described in N2570? 4-6-8 not clear direction
Does the committee wish two frames to describe these functions in N2570? 4-2-12 underwhelming
Does the committee wish the single-frame solution for both binary and decimal, as described in the second example in N2570? 6-0-11 more consistent support
Does the committee wish to adopt, as is, the single-frame solution for both binary and decimal as described in the second example in N2570? 6-0-12
Does the committee want a function family for the mk_ckd services in N2543?? 12-1-8 clear direction
Does the committee want a type-generic macro for the mk_ckd services in N2543?? 8-3-10 favor, but more undecided
Does the committee wish a defer statement, along the lines of N2542, in some way? 12-3-6 clear direction
Does the committee wish a defer statement, along the lines of N2542, as a TS? 14-1-7 clear direction
Should defer-ing be determined by scope? 4-8-9 no real sentiment
Should defer-ing be determined by an explicit guarded keyword? 4-4-11 topic needs more exploring
Does the committee want to allow dynamic execution of the defer statement, rather than static? 4-13-4 clear direction away from dynamic.
Do we wish option 1 with regard to specifying multi-threaded execution? 20-0-2 clear direction that our response to this request be option 1.
Does the committee wish to adopt N2549 into C23 with the editorial change to update the table in Paragraph 5? 23-0-0 adopted
Does the committee wish to adopt N2562 into C23 as is? 22-0-1 adopted
Does the committee wish to adopt N2563 into C23 as is? 23-0-0 adopted
Does the committee wish to make any changes, along the lines suggested in N2571? 11-3-7 We do need other votes
Does the committee want to editorially add the word "both" before "non-negative and less than n" in 7.21.6.5p3: 13-3-6 adopted
Does the committee wish to make the first suggested change from N2571 to C23? 6-7-6 no consensus

8.2 Review of Action Items

Seacord: Update TS17961 (fold in DRs and update C11 references to C17). Keaton will send Seacord the URL for DR list.
Svoboda: Write a proposal to define "type-generic" (for functions and macros) and make it consistent.

Consider N2558 as a possible usage.
Banks: Write a paper to strengthen the recommended practice for using const in these areas, along the lines of N2565.
Ballman: Write a paper on adding separators to C integer literals, along the lines of C++
Wunsch: Write a paper on adding binary I/O for printf() and scanf().
Ballman: Write a version of N2563 for C++.

9. PL22.11 Business

PL22.11 Issues only

10. Thanks to Host

10.1 Thanks and apologies to Alex Gilding, the originally intended host

10.2 Thanks to ISO for supplying Zoom capabilities

11. Adjournment (PL22.11 motion)