Final Minutes 30 November – 4 December, 2020

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

WG 14 / N 2691

Dates and Times

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

30 November, 2020	14:30 – 18:00 UTC
1 December, 2020	14:00 – 17:30 UTC
2 December, 2020	14:00 – 17:30 UTC
3 December, 2020	14:00 – 17:30 UTC
4 December, 2020	14:00 – 17:30 UTC

Meeting Location

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
Bill Ash			SC22 CM
Aaron Bachmann	Austrian Standards	Austria	Austria NB
Roberto Bagnara	University of Parma	Italy	Italy NB, MISRA Liaison
Aaron Ballman	Intel	USA	WG21 Liaison
Dave Banham	BlackBerry QNX	UK	MISRA Liaison
Andrew Banks	LDRA Ltd.	UK	MISRA Liaison
Rajan Bhakta	IBM	USA, Canada	PL22.11 Chair
Lars Gullik Bjønnes	Cisco Systems	USA	
Melanie Blower	Intel	USA	
Alex Gilding	Perforce / Programming Research Ltd.	USA	
Jens Gustedt	INRIA	France	

Name	Organization	NB	Notes
Barry Hedquist	Perennial	USA	PL22.11 IR
Tommy Hoffner	Intel	USA	
David Keaton	Keaton Consulting	USA	Convener
Philipp Krause	Albert-Ludwigs-Universität Freiburg	Germany	
Kayvan Memarian	University of Cambridge	UK	
JeanHeyd Meneide	NEN	Netherlands	
Maged Michael	Facebook	USA	
Joseph Myers	CodeSourcery / Siemens	UK	
Miguel Ojeda	UNE	Spain	Spain NB
Thomas Plum	Plum Hall	USA	
Clive Pygott	LDRA Inc.	USA	WG23 liaison
Robert Seacord	NCC Group	USA	
Peter Sewell	University of Cambridge	UK	Memory Model SG
Nick Stoughton	USENIX, ISO/IEC JTC 1	USA	Austin Group Liaison
David Svoboda	CERT/SEI/CMU	USA	Scribe

Name	Organization	NB	Notes
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Martin Uecker	University of Goettingen	Germany	
Freek Wiedijk	Plum Hall	USA	
Michael Wong	Codeplay	Canada, UK	WG21 Liaison
Jörg 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 JTC 1 Summary of Key Points [N 2613]
- 1.4.4 INCITS Code of Conduct

1.5 Approval of Previous WG 14 Minutes

- 1.5.1 August, 2020 [N 2588] (WG 14 motion)

  Moved by Seacord, seconded by Tydeman, no objections

- 1.5.2 October, 2020 [N 2605] (WG 14 motion)

  Moved by Ballman, seconded by Bhakta, no objections

1.6 Review of Action Items and Resolutions

- *Seacord*: Update TS 17961 (fold in Defect Reports and update C11 references to C17).

  Seacord: In progress

- *Svoboda*: Write a proposal to define "type-generic" (for functions and macros) and make it consistent. Consider N2558 as a possible usage.

  Svoboda: No progress yet

- *Banks*: Write a paper to strengthen the recommended practice for using const in these areas, along the lines of N2565.

  Banks: Not much progress

- *Ballman*: Write a paper on adding separators to C integer literals, along the lines of C++

  Ballman: Done, on the agenda: N2606

- *Wunsch*: Write a paper on adding binary I/O for printf() and scanf().

  Wunsch: Working on first revision on paper, with feedback from Myers. Interim draft paper submitted; it is N 2618

- *Ballman*: Write a version of N2563 for C++.

  Ballman: Done, it is P2246R0.

1.7 Approval of Agenda [N 2614] (PL22.11 motion, WG 14 motion)

Keaton: 2 agendas: proposed: Let's discuss on Friday this update: http://www.open-std.org/jtc1/sc22/wg14/www/docs/tmp/n2618.pdf. No objections.
Bhakta: Can I object on Friday if I have not studied this paper yet?
Keaton: Yes. It is tentatively accepted for now. Let's change future meeting

times this week from 14:30 to 14:00.
Bhakta: Any objections to the half-hour offset?
Keaton: Yes, Europe already objected. Modulo those, do we want to approve the agenda? Moved by Nick, seconded by Gilding. PL22.11 agenda moved by Ballman. seconded by Tydeman. No objections.

1.8 Identify National Bodies Sending Experts

Austria, Canada, France, Germany, Italy, Netherlands, Spain, UK, USA

1.9 INCITS Antitrust Guidelines and Patent Policy

1.10 INCITS official designated member/alternate information

Ojeda: I am working with my national body (Spain). I do not know when I will be approved.
Update: Miguel was approved Thursday morning.

2. Reports on Liaison Activities

2.1 ISO, IEC, JTC 1, SC 22

- 2.1.1 An ISO/IEC proposal to revoke public document access was defeated.

  Keaton: There is no change, and documents will remain public.
  Gustedt: What are the reasons for this proposal?
  Keaton: ISO/IEC misinterpreted that all documents must be closed. (ISO does require this, but JTC1 takes some rules from ISO and others from IEC.)

2.2 PL22.11/WG 14

Bhakta: Are they looking for volunteers for the rationale?
Keaton: We do not have any volunteers yet. We are seeking volunteers.
Meneide: Are there any recent rationales?
Keaton: We have not had a rationale since C99. We would encourage a volunteer to produce rationale about the new C23 features. The rationale is an unofficial document, not published by ISO.

- 2.2.1 Convener's Report and Business Plan [N 2609]

  Ballman: Is the Free Standards Group the same as the Austin Group?
  Stoughton: Yes, the Free Standards Group was the predecessor of the Linux Foundation. The Linux Foundation has no projects in ISO arena. I am one of SC22's liaisons to the Austin Group. Updating "Free Standards Group" to "Austin Group" would be good.
  Action Item: *Keaton*: Change Stoughton to represent Austin Group rather than Free Standards Group.

- 2.2.2 Proposed C23 Schedule [N 2610] and Charter Revised with Proposed Schedule [N 2611]

  Keaton: Are there any objections to the schedule?
  Ojeda: When is the last meeting for new features in C23??
  Keaton: Aug 31, 2021 is the mailing deadline for the Fall meeting (in Minneapolis)
  Ballman: Are we now switching to 3-year schedules for C, instead of 10-year schedules?
  Keaton: In ISO, a Working Group (WG) is normally destroyed when it creates its document. But in JTC1, the WG stayed indefinitely to maintain the document. But in Dec 2018, ISO and JTC1 changed: Now a WG stays until it has published all documents in progress and the SC containing it disbands the WG. WG14 counters by submitting another C revision now to prevent disbandment when C23 is published. So we do not have to revise every 3 years, but we must have revisions in place.
  Gilding: Do we want to stay coincidental with WG21's 3-year

publication cycle?
Keaton: It was coincidental. But it is a good topic for the C/C++ liaison group.
Pygott: I can not think of a proposal that has been agreed upon within 6 months.
Ballman: I am concerned about trying to push WG14 to a 3-year schedule. Our customers (mostly large government with long contracts) view C++'s standard as not stabilized. The 3-year schedule is tolerable in C++ only because they are the big "experimental" language.
Keaton: We can escalate this to SC22 and JTC1 if a 3-year revision schedule proves harmful to customers.
Gustedt: Can we still discuss outstanding Technical Specification's after WG23 without a new revision published?
Keaton: We can discuss anything until we get disbanded.
Keaton: The rule says that the SC "shall" disband a WG with no further publications.
Tydeman: What does this do with regard to Defect Reports?
Keaton: If a WG is disbanded, the editor is requested (but not required) to act on any incoming Defect Reports.
Ballman: Do we still plan to alternate bugfix releases with feature releases?
Keaton: I had not planned that far ahead.
Gustedt: I expect we will have a backlog of new features after C23 is published.
Keaton: Yes, including from the Memory Object Model group.

- 2.2.3 Outreach

2.3 PL22.16/WG 21

- 2.3.1 Create a C/C++ Collaboration Study Group

  Keaton: Propose a new C/C++ Study Group, chaired by Ballman, assist. chair by Meneide. WG21 will create their half too.

Ballman: WG21 already created their half at their last meeting.
Keaton: Are there any Objections? (None)

2.4 PL22

2.5 WG 23

Keaton: We have completed the first 3 revisions of its TR. They were turned down because it was a TR. WG23 is converting those Technical Reports into standards. It is free because it is a catalog of material from other standards. (So ISO C cannot be free because it contains original material).

2.6 MISRA C

Pygott: Bagnara, you are on MISRA, can you report?
Keaton: Bagnara's audio is not working.
Bagnara: We are proceeding with the accommodation of C11 and C17 features. There is not much else to report.

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 charter was to add safety to TS 17961. Due to conflicts with MISRA, we decided not to add safety. At the previous meeting, we decided to promote TS 17961 as is to a standard. There is nothing left for the group to do under its original charter now, so we should disband the SG. The chair may contact people to form a new SG.

Pygott: I am intrigued on what the chair has in mind with alternative tasks?
Seacord: I am still technical editor for TS 17961, right?
Keaton: Yes. Seacord is responsible for editing the document.
Bagnara: There is a similar study group in WG21.
Ballman: WG21's SG studies undefined behavior, security, and vulnerabilities.
Pygott: I was in Belfast for WG21, and got invited to the safety study group proposal. It is much more like the scope of MISRA.
Ballman: But WG21's SG was not trying to incorporate MISRA into TS 17961.
Keaton: Are there any objections to disbanding this SG? (None)

3.3 C Memory Object Model Study Group

Gustedt: We can not talk about the TS until tomorrow night, but can discuss on Wednesday.

4. Future Meetings

4.1 Future Meeting Schedule

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

8-12 March, 2021	Virtual, 14:30-18:00 UTC each day (proposed)
17-21 May, 2021	Virtual, 13:30-17:00 UTC each day (proposed)
4-8 October, 2021	Minneapolis, Minnesota, US (tentative)
31 January - 4 February, 2022	Portland, Oregon, US (tentative)
11-15 July, 2022	Strasbourg, France (tentative)

Keaton: The May meeting is 1 hour earlier because of Daylight Savings Time. Comments?
Pygott: No problem with that.
Seacord: The meetings should be evenly spaced out so we can get more done between them.
Keaton: We could move the second meeting later, to June or July. Remember that we do not want this meeting to be too late because October 2021 will be a full meeting. I will propose June 14-18. Comments?
Gilding: I was thinking that same time, so that we can have two mailings, and a rush of proposals for new features.
Ballman: Are we still planning on having a virtual component on our face-to-face meetings?
Keaton: Yes, WG14 has a policy of allowing people to dial in.
Keaton: WG21 has separate policies. Sometimes their hosts imposed expensive cancellation fees. So WG21 asked SC22 to make it impossible, to avoid paying cancellation fees. This does not affect WG14 because we have less than 100 people. We can decide three months ahead of time if we must convert the Oct 2021 meeting to another virtual meeting.
Svoboda: Dialing in to a face-to-face meeting has lots of technical problems due to bad microphone coverage.
Keaton: That varies depending on what our hosts provide. I try to make it good every time.
Keaton: I will update the May 2021 meeting to June, and update the mailings, to go into the next agenda.

4.2 Future Mailing Deadlines

Note: Please request document numbers by one week before these dates.

Post-Virtual-202011	18 December 2020
Pre-Virtual-202103	5 February 2021
Post-Virtual-202103 / Pre-Virtual-202105	9 April 2021
Post-Virtual-202105	11 June 2021

Pre-Minneapolis	3 September 2021
Post-Minneapolis	29 October 2021
Pre-Portland	31 December 2022
Post-Portland	25 February 2022
Pre-Strasbourg	10 June 2022
Post-Strasbourg	12 August 2022

5. Document Review

Monday

- 5.1 Ballman, What we think we reserve [N 2572]

  Keaton: The reason for the C/C++ difference with double underscores was name mangling. The "Cfront" C++ preprocessor would turn "::" into "__".
  Gustedt: I wish to ask the chair of the C/C++ SG about the double-underscore question. Perhaps it should be removed from C++?
  Ballman: If an implementation wants to experiment with a proposal, this allows them to capture potentially reserved identifiers and turn them into reserved identifiers on a trial basis.
  Gilding: Regarding name mangling, I prefer that this not be reserved in C. "Gambit" is another compiler that mangles names, so that problem is not unique to C++.
  Ballman: If I understand correctly EDG still supports Cfront's implementation, but perhaps they do not care.
  Myers: Doing "complex" separately might be safer.
  Bhakta: Do not hold off on this paper because of hypothetical issues (such as "complex"). Put this into C23.
  Ballman: Tydeman pointed out two editorial fixes in the paper.
  Straw Poll: Does the committee wish to adopt N 2572 with editorial changes into C23? 21-0-0 adopted.

- 5.2 Ballman, Digit separators [N 2606]

  Bhakta: Thanks for picking a digit separator that works with EBCDIC!
  Gustedt: One editorial nit: You could remove the optional wording.
  Gilding: Could the proposed wording be improved by adding non-normative examples?
  Ballman: I did not think they were necessary, but I have no problems adding examples.
  Wunsch: Such examples could also go into a "C23 Rationale" document.
  Ballman: Does anyone want to keep "optional"? (no) There are enough editorial changes that I do not want to vote on this immediately.
  Straw Poll: Does the committee want something along the lines of N 2606 to be added to C23? 21-0-1 looking good

- 5.3 Pygott, Proposed enhancement for C23: Allowing the programmer to define the type to be used to represent an enum [N 2575]

  Krause: Is there a use case for having a type with qualifiers?
  Pygott: How compatible is this with C++?
  Uecker: How compatible is type-specific enum with the underlying enum type?
  Pygott: From our discussion in August, how would a generic function treat two enumerated types with the same underlying type? Are they the same or not?
  Gilding: As Myers chatted: Non-transitivity of type compatibility already exists for enums and implementation-defined type compatible with enums. We would also have a way forward that is compatible with C++.
  Bhakta: Regarding the new constraint in section 6.7.2, paragraph 2, why is that a constraint, and not semantic?
  Pygott: I can move it down to semantics.
  Gustedt: The next paragraph also reads like semantics, not a constraint.
  Pygott: I also have comments from Seacord.

- 5.17 Meneide, Preprocessor embed [N 2592]

  Keaton: This item was planned for Friday, but we have time to discuss it now.
  Gilding: I saw an example of a function call with "#embed ,0", suggesting this list could be used as an initializer-list. Was that intentional? It would complicate parsing, and function argument lists could lose performance.
  Meneide: I do not have an example of that in the paper. #embed could go anywhere an initializer-list could go. I did not add a constraint saying it must be used to initialize an array because you would have to mix the language and preprocessor together to enforce the constraint.
  Myers: Function arguments allows an initializer-list.
  Gustedt: Allowing #embed everywhere that comma-separate list of numbers is allowed is too hard. So is forcing people to include limits.h before they use #embed.
  Meneide: I kept that because you do not know the size of unsigned char in semantics, so you need to include limits.h.
  Meneide: In Clang, I generated a make-dependency format. This plugs into the #include section.
  Myers: Does anyone still have a separate preprocessor from the compiler that does not know things like sizeof(char)?
  Bhakta: Yes
  Meneide: I found several tools that did have that behavior. Not up-to-date tools though. Some static analyzers have preprocessor steps like that.
  Ballman: This does not provide semantic constraints as written…the implementation already had the freedom to do this. I would like to not require limits.h before using a preprocessor directive.
  Gustedt: Since this is normative, the preprocessor should be able to mine limits.h itself (without requiring an #include). This feature may include limits.h.
  Meneide: Can we also include "unsigned char UCHAR_MAX" in the preprocessor the way we smuggle in INT_MAX?
  Gilding: It is necessary configuration information. so it could be

overridden by a command-line option or configuration file.
Bhakta: If you need to pack multiple octets, the ordering of octets in a 16- or 32-bit word can be an issue.
Meneide: I left that up to the implementation.
Meneide: I would like to know if WG14 wants #embed-produced things inside function argument lists, along the lines of N 2592?
Ballman: Are we allowed to mention things like endianness?
Keaton: Yes that would work.
Straw Poll: Do we want to allow #embed to appear in any context that is different from an initialization of a character array? 5-8-6 leaning in the direction of no but not clear
Wiedijk: What does "allow" mean (in the straw poll) here? Do we require or just not forbid it?
Meneide: We can leave it as unspecified behavior.
Ballman: I do not like the lack of portability around trying to make this unspecified. I prefer to leave that in the compiler implementation space.

Tuesday

- 5.4 Seacord, Specific bit-width length modifier [N 2587]

  Myers: The text should say "width" not "bit-width".
  Seacord: OK
  Gustedt: The specification is not enough because specifiers are meant to distinguish multiple types with same widths. These functions need sizes not widths. So specifying just a width may not be sufficient.
  Seacord: The specification covers the least and exact types. I thought if least and exact width types are defined, they would both be the same. Do you have a specific example where least and exact types disagree?
  Gustedt: There are architectures where int_least16 is actually 32 bits? Sorry, I don't know of a more specific example.
  Keaton: "Least" means "Least number of bits that big".
  Gilding: The "%w" does not control how many bits are read, so why is

this an issue?
Seacord: Agreed. So 16 versus 32 bits is not a problem because of integer promotions.
Keaton: "exact" means "exact number of bits". So int32 cannot be 64 bits.
Myers: If the exact-width types exist the least type must be the same type as the exact-width type.
Seacord: Yes, you have to tie these two things together.
Svoboda: A least48 type might be 64 bits long. How does printf() know to read 64 bits for the "%w48" integer?
Seacord: It would have to infer the type.
Ballman: There are two proposals proposing to add bit-specific modifiers for formatted I/O. I am concerned that these will confuse users. Will this solution clash with extended integers (N 2590)?
Svoboda: There are two purposes here: How many bits to read versus how many to print? We should have this clarification.
Seacord: There is not length of promoted size, we assume that the implementation can determine the size. That why this paper does not support ExtInt, because those types do not undergo promotion.
Gustedt: Ballman, Seacord is coming from intmax_t and extended integer types, regardless of how they are classified. The tool to print is ugly with these macros, so this is a way forward. Adding other exact-bit-width types that do not promote do not fit into our type system.
Krause: Implementations are not disallowed from using extended integer types, only from types that exceed intmax_t.
Gilding: I think users would prefer this over other length modifiers like "h". To me, this is a big simplifying factor for all types. It is useful even if it only supported standard type sizes (16, 32, 64, etc).
Myers: I like the concept of multiple letter modifiers: e.g.: "wx" for exact-width, "wl" for least width, etc.
Ballman: This should be "default integer promotions" in the "wN" normative text.
Seacord: Agreed, but I figured we should correct that globally throughout the standard.
Straw Poll: Does the committee wish to adopt N 2587 into C23 as is?

8-11-3 no consensus
Straw Poll: Does the committee wish to require that a least integer type matches an exact integer type when both are defined in C23? 19-1-3
Straw Poll: Does the committee wish to add a second letter to indicate "exact" or "least" to the width modifier in N 2587? 8-5-8 Leaning yes, but a lot of abstains

- 5.5 Krause, Character handling for freestanding implementations [N 2576]

  Svoboda: Why is ctype.h not currently required in freestanding implementations?
  Krause: I do not know.
  Keaton: I do not have any historical recollection either.
  Bhakta: Not all freestanding implementations are on small devices; large devices can also be freestanding. It is also not necessarily easy to do for everyone, thanks to multiple code-pages and support for multiple locales. Perhaps a preprocessor-query like "has-include" would be sufficiently portable here?
  Wunsch: Even with multiple code pages, a system still has to provide a unified means of classifying characters.
  Bachmann: There is no big advantage of having ctype.h be optional.
  Bhakta: If you have something like that, say for a kernel, then users should have the option to not require allocated memory for character handling for things like OS kernels.
  Bachmann: Memory should not be an issue, if we do not link in ctype.h we do not use resources.
  Gilding: Something like a kernel should allow you to control whether ctype.h is linked in or not. People seem to assume it will have a large memory footprint even when unused.
  Bhakta: Here is one example: If you have a number of systems linked together with a hypervisor, you have a small chip with limited memory dedicated for various purposes. So memory is critical on the hypervisor chip and we need the ability to leave out extra memory, which ctype.h would require. On that level, you have only static linking (but no

dynamic linking) and a linker that is not necessarily smart enough to identify unused memory.
Ojeda: You do not include ctype.h if you do not want to use it, so linker does not include it. What we want is the ability to control linking in the memory used by ctype.h, and that is outside WG14's scope.
Krause: In Bhakta's example, a hypervisor would be similar to tiny devices, but also have multiple locales.
Bhakta: In my example, people who write implementation for the hypervisor do not want the memory usage of ctype.h to be required. We do not want to force the overhead on anyone who does not need it.
Krause: But could your hypervisor implementation have some switch to not link in locales?
Bhakta: Right now, C only has "freestanding" or "hosted" implementations.
Krause: Why prevent multiple locales from including ctype.h?
Bhakta: There is either local support (with everything) or none. There is currently no single-locale option.
Stoughton: Required includes affect the language, not the library. Adding ctype.h is very different, it affects the library, not the language.
Krause: We added some string functions at the previous meeting.
Straw Poll: Does the committee wish to adopt N 2576 into C23 as is?
4-8-10 no sentiment to add

- 5.6 Blower, Adding Fundamental Type for N-bit Integers [N 2590]

  Gilding: In interaction with generics, this is symmetrical with how arrays are handled. _Generic does not support array dimensions or ExtInt widths.
  Gustedt: I am still horrified by this. This is a real constraint for implementors. It excludes implementations where people could use a struct with a bit-field. It is much too much to impose on everyone's implementation for a marginal feature. We have cleaned up integer types. This will bring us years of difficulty teaching and more cleanup. I do not like having ExtInt be a keyword in the language (rather than as

a macro). See my paper (N 2522) about how to use a keyword.
Ballman: We do not want implementations to do whatever they want for ExtInt as a macro. It must be a keyword because it is exempt from integer promotions.
Bhakta: Atomic ext-int lock-free. Did you want this to not be suitable in a preprocessing directive?
Ballman: The intent is you can use ExtInt as an atomic lock-free at a particular bit-width. We might have problems scaling for all potential bit-widths. Do you think ExtInt should be integrated into atomics now?
Bhakta: I thought it was better to not be in atomics for now. But this is not a big concern.
Bhakta: This still sounds like invention, as it is not part of the market yet. Perhaps this should be a Technical Specification (TS)?
Ballman: We know of 4 different implementations. We are not opposed to a TS. I want some concrete questions from the committee regarding TS versus language extensions. Its hard to get implementors to support a TS.
Meneide: I also want to match more than one exact width for ExtInt. Perhaps we should improve generic selection?
Myers: This could be 2 papers: one for the new types, and one for the width-of operator.
Ballman: I have no problem splitting this into two papers.
Seacord: Among my cryptography co-workers, they could not see the usefulness of this paper. I agree a TS is not the right direction because of prior art. Perhaps an optional annex. Perhaps we should target some hardware to use this feature?
Ballman: Perhaps your co-workers are used to the way things are already done? I have heard that there were cryptographers (concerned with AES) who would be interested in using this. ExtInt does not extend the design space that allows you to solve new problems. With regard to an annex, we are not opposed, but we do not know how we would specify this. They are more additive than transformative. Integer promotion is important to us, but I do not understand how we would avoid default integer promotion in an annex.
Bhakta: Decimal floating-point types are already in an annex.

Krause: Also complex numbers.
Ballman: I can study those.
Krause: I prefer that the type be "BitInt" because they are not extended integers anymore.
Straw Poll: Does the committee prefer changing the name from ExtInt to BitInt in N 2590? 12-0-8 there is sentiment to do that.
Straw Poll: Does the committee prefer splitting N 2590 into a "bare-minimum" paper and an ancillary paper? 11-0-8 there is sentiment to do that.

- 5.7 Ojeda, secure_clear [N 2599]

  Svoboda: One editorial change in the normative text of all three proposed functions: The implementation may not clear other copies of the data (e.g. intermediate values, cache lines, spilled registers, etc.). Change "may" to "might", because "may not" is interpreted as normative in English while "might not" is not. I wonder if alternative #1 is viable (e.g.: does any platform prefer to write data other than 0?)
  Keaton: The ISO usage has changed recently, they do not like "may not" anymore.
  Ojeda: I do not know of any platforms for which option #1 (indeterminate values)?
  Krause: Some platforms might be faster using the accumulator value. I think memset_explicit() is most elegant, because it is similar to memset(). I asked about this on the BSD mailing list. It is useful to overwrite with 0, not useful to overwrite with different or indeterminate values. Their function is named bzero_explicit().
  Bachmann: Using "Recommended practice" is not a good idea. It is useful to have a memory barrier after writing, which impacts performance. I prefer weaker wording like "should not have significantly worse performance".
  Ojeda: Some people wanted a function to be "as secure as possible", while others wanted current practice with only the "not optimized away" part being new.
  Gilding: I do not think "Recommended practice" is necessary.

Performance is less important than security. I think there is great value in the option #1 (indeterminate values).
Uecker: I am concerned about 1st alternative wording; what "indeterminate values" means here. I prefer the other two alternatives, which are more clear.
Seacord: I dislike trap representations (with regard to 1st alternative), because they are an excuse to make a subsequent read into undefined behavior. We did not address whether values can wobble. Alex's idea that we could add all three functions is decent. memset_explicit() is the most general. We could also pull memset_s() into main standard (from Annex K), and modify it to not use runtime constraint handlers. We left a lot of work in Annex K…perhaps we should eliminate memset_s() when adding this proposal? I would be happy with "indeterminate values that are not allowed to be trap representations."
Keaton: The memory object issue (wobbly values) has not been dropped; it is addressed in the Memory Provenance Model, which we will discuss tomorrow. But it will not get added to C23.
Banham: This is an instance of a broader problem; we want more control over what the optimizer is doing.
Ojeda: Some have suggested this for C++. We can explore that, but it would be a major proposal. We can standardize this and then explore the broader optimization problem.
Bhakta: Filing with 0's is faster on zed machines. We really should remove memset_s() if we add one of these functions. It does make sense to have a more general approach to optimizer suppression. Perhaps we can also use or revise nodiscard attribute to prevent optimization?
Ballman: No, attributes are not appropriate because program must remain correct in if its attributes are ignored. A keyword would be more appropriate. I am uneasy with memerase(), because it does not indicate if resulting memory are initialized.
Krause: Adding all three is a big cost to users, and makes the standard less readable. I want there to be only one function.
Svoboda: Trap representations are unlikely to help (because the exact trap values depends on the array type). We can promote or delete

memset_s() but let's do it separately. 3rd alternative memset_explicit() is the same as memset_s() with the runtime-constraint-handling removed.
Wong: WG21 wants to also address safety and security, and is also reviewing this paper. We are creating a Safety Security Review Group, which reviews material without generating new material. There will be an SG14 call on December 9 to discuss charter goals and chairs for this group. We will review this paper, but not on December 9.
Ojeda: So should the function write 0 or another value?
Straw Poll: Does the committee wish to adopt something along the lines of alternative 3 of N 2599 into C23? 16-1-6 clear direction.
Ojeda: I need direction on how to improve wording?
Keaton: We recommend that you have a discussion on the reflector and then submit a new paper. Also you should request 1 hour, rather than 30 minutes, to discuss this issue at the next meeting..

- 5.8 Gilding, Qualifier-preserving standard library functions [N 2603]

  Krause: I like this proposal, but it will break code that used to compile.
  Gilding: I felt that here the user has control over what is const; this is a difference from strerror() (N 2526)
  Svoboda: Would this apply to volatile objects as well as const objects? How about atomic objects?
  Gilding: I have ruled out volatile and atomic explicitly in the normative text. Neither are appropriate. I do want this to be generic to address other future qualifiers, such as a memory-space qualifier.
  Bhakta: Generic functions have a drawback: They are not address-able. This would break more than just user-controllable code.
  Gilding: Because these functions currently have only one address each, I would expect them to still be preserved as external and address-able objects. My hope is that this does not break anything.
  Bhakta: I do not see much gain in doing this, for the functions in the paper.
  Gilding: I do not have numbers of people making these errors, in particular losing constness.

Ballman: Some static-analysis-tool users miss bugs performing a typecast which is legal by the typing rules, but not legal by the rest of the language.
Banham: I am not sure about the syntax. The user might want to write functions with similar properties (e.g.: preserve constness in return values).
Myers: More work is needed in wording, specifically when const is preserved, and similarly for volatile and other qualifiers.
Seacord: _Generic seems to be the C solution for generic programming. Is it time to deprecate "auto"? We should probably not say these are generic.
Gilding: These functions are parametrically polymorphic; they do not use _Generic.
Wiedijk: What is "Q"? How hard is it to parse this?
Gilding: "Q" is a qualifier variable. I lifted this convention from the atomic library. They are parsed like the atomic functions.
Straw Poll: Does the committee wish to preserve "const" across the functions listed in N 2603? 13-1-8 leaning to yes
Straw Poll: Does the committee wish to adopt something along the lines of the first syntax option in N 2603 into C23? 10-4-7 leaning to yes

Wednesday

- 5.10 A Provenance-aware Memory Object Model for C
- Tutorial Slides [N 2378]

Wiedijk: Can you please summarize what exposed pointers is about?
Sewell: If a pointer which points past its item points to the start of next item is cloned, then the clone cast to integer. If both pointers have been exposed, then casting the integer back to a pointer. This becomes ambiguous as to which provenance the cast pointer should pick up. Option #1: this is forbidden. Option #2: the pointer gets provenance of the region of memory it points into. This means casting a one-past

pointer to int and back is still forbidden. The solution is to be symbolic: If cast int-to-pointer is ambiguous with regard to provenance, it gets symbolic provenance until one performs an access or arithmetic that disambiguates the pointer.
Svoboda: How is provenance related to the overhead stored in fat pointers?
Sewell: Overhead is key…provenance is not something that a program keeps track of at run time. Provenance is only in the abstract machine and alias analysis. Basic provenance cases are isomorphic to fat pointer overhead. At any one point in time pointer provenance is the same fat pointer overhead. But fat-pointer-overhead info can be foiled by freeing and allocating memory which might duplicate a previously-existing object. In that case pointer bits might be identical but provenance would be different.
Gilding: Are there any new optimizations found or disabled by provenance model? Does this enable people to do more things?
Sewell: We are only focused on what disables optimizations.
Michael: I would like clarification on lifetime end-pointer zap. What are the implications of this TS with regard to undefined behavior?
Sewell: This TS does not attempt to solve issues of pointer zap.
Ballman: Forming a pointer with bad provenance is not undefined behavior, but dereferencing it is, right?
Sewell: Some of those things were undefined behavior before. We are not trying to change that in this draft.

- Working Draft Technical Specification [N 2577]

  Seacord: Is it possible for pointers p and q to have identical bit-pattern but p==q = False?
  Sewell: Yes!
  Seacord: Is this considered an unfix-able bug?
  Sewell: Our intent has been to not change things. We might need to change this though. See slide #67 for details on pointer equality.
  Gilding: Does assigning pointer to point to an external variable constitute exposure?

Gustedt: Assigning to pointer carries provenance, so this is not a problem for us.
Seacord: There is a lot of history with C with regard to books, training, etc. Is it not possible to keep the terminology everyone is familiar with? In particular, "Memory Management Functions".
Gustedt: The term "memory" is ambiguous in the standard. Malloc & similar functions manage storage without talking about real physical memory underneath. But we could change headline back to "Memory Management" if people wish.
Seacord: In your document, section 7.22.3, paragraph 2 says "Memory Allocation Functions".
Gustedt: Thanks, we should fix that.
Krause: Why do we have aligned_alloc() if malloc() aligns things?
Gustedt: This is off-topic, but the point is to allow larger alignment than necessary.
Svoboda: Is there still no well-defined way to tell if realloc() moved an object in memory?
Gustedt: There is no semantic change here. The new object and old object from realloc() are considered to be two different objects. Old and new pointer values might be equivalent but their objects are different. So comparing these pointers is still undefined behavior.
Sewell: We might want to re-visit this problem.
Wong: What does it mean when an allocation is marked as exposed?
Sewell: It means if a pointer is cast to an integer, and when cast back to pointer it regains its original provenance.
Wiedijk: Using "%p" is another way to expose a pointer. Also examining bit representations, using unions. or fwrite().
Gustedt: Yes. In the abstract machine if integer-to-pointer cast points to one past storage instance of exposed pointer, the pointer regains its previous provenance.
Gustedt: Conversion between pointers and integers is outside our scope, we just manage the provenance..
Seacord: We say "the pointer / address / storage instance is exposed". So informal conversation is actually incorrect.
Sewell: Yes, what is exposed is the address to storage instance…e.g.:

the bit-pattern, rather than the pointer value, which includes provenance.
Wiedijk: Why are not all addresses exposed all the time?
Sewell: Personally, I would prefer that. The expose mechanism means you can observe a desirable undefined behavior by examining a single execution. Otherwise one relies more heavily on non-determinism to observe undefined behaviors. It seems that WG14 and WG21 prefer the concrete view.
Keaton: If everything is exposed, some optimizations will be disallowed.
Sewell: Yes, but surprisingly few.
Gustedt: It is easier to argue about unexposed addresses.
Bagnara: In how many pages can you explain what can and cannot be done, to C programmers?
Gustedt: We tried to do this today. Provenance, storage instance, exposed, and synthesized. These are the concepts you must clearly explain. The rest are standardese.
Sewell: If one wants to understand all cases, one needs a bit more. Not all that difficult, compared to something like arithmetic semantics in C.
Gilding: In many places it is a lot simpler.
Seacord: Is this not lower-priority since it cannot get added to C23? We have also farmed out other things to this study group that got deprioritized because of provenance. Should we address these other problem areas?
Gustedt: These other things are more difficult to reach consensus on. So they would also never make it into C23 even if we prioritized them now.
Seacord: So are we not concerned about uninitialized reads or wobbly values?
Sewell: In the list of papers, several talk about those things, like wobbly values. We focused on provenance because we could obtain consensus.
Krause: What would stop us from voting this into C23 then?
Keaton: Many people objected to that, because of unintended consequences with regard to optimizations.

Sewell: That would also assume we have serious compiler feedback, which we currently lack.
Gustedt: This is more of a certification that compilers abide by these constraints. No one has currently certified that.
Gilding: Would it be acceptable to extract papers out of this? Like one on storage instances, without provenance, going into C23?
Gustedt: There is a paper about storage instances. We still could discuss that. Not very difficult to do. It would not reduce the size of the TS, which is based on C17. A C23-based TS would be smaller though.
Wiedijk: How can one participate in the Memory Object group? Can you publish how to get involved?
Sewell: There is a mailing list among the authors.
Ballman: I can add info to the C study group website if you publish info on getting on the mailing list.
Sewell: We suggest people read the draft TS diffs offline and submit comments. It is unclear how long our feedback window should be. Perhaps we can aim for consensus on TS diffs by next WG14 meeting in March.
Wiedijk: One way to do this is by giving weird corner cases where compilers behave "oddly". But compiler writers will want to prevent changing their behavior. How do you see that?
Sewell: It is good to identify those examples; we have done many of those in the TS. Many compilers have "won't-fix" bugs, so that info may come with a big pinch of salt. It is hard for us to identify causes for these discrepancies.
Keaton: This is the first day we have been able to discuss the TS, so I understand people being unfamiliar with content. Please read and be prepared to discuss in the March meeting.
Svoboda: Since our virtual meetings alternate between different sets of topics, would we discuss this in March meeting or June meeting?
Keaton: Both.
Sewell: it depends on what feedback we get.
Bachmann: Do not talk about this in a single day. Perhaps Monday and Thursday and allow thoughts to settle?
Keaton: Good idea; we could do that in the March meeting.

Seacord: The 2-3 years we have on this; how fungible is that?
Keaton: Only as a last resort. It is bad form to build that into our planning.
Sewell: Again, I encourage feedback from compiler implementors.
Keaton: Blower, how long does Intel need to evaluate this TS?
Bhakta: Like Blower, I cannot say how much time I would need. But having a published TS will make it a magnitude more likely to spend resources and provide an estimate.
Gilding: Would not take my group long, now that it is final. Weeks rather than months to study this TS and modify and fix our model.
Keaton: Blower has audio problems, but she chatted that LLVM has an analysis round-table. It would be good to reach out to that group.
Sewell: I think we got them in the WG21 meeting.
Keaton: So some people can analyze this quickly, while others need this published as a TS so they can gain resources to analyze it. We can get some good feedback by March.
Gustedt: Can we also get feedback on the points that come after provenance. What other points are important? (uninitialized values , end-life zap)? What is the most urgent?
Ballman: Uninitialized values come up surprisingly often.
Gustedt: Because they are clearly related to padding.
Bachmann: Another version of the proposal to undermine type system to access storage instance. This is important for performance reasons.
Gilding: Did we not reach consensus on end-zap in the London meeting?
Uecker: End-zap somehow depends on provenance.
Sewell: We should distinguish end-zap pointers that need to be compared against versus dereferenced.
Bachmann: We can do these topics incrementally. Later on we can allow more situations.
Gilding: The consensus in London was removing end-zap completely. This seems simplistic to me.
Uecker: There is no improvement if we allow comparisons without fixing other things.
Keaton: The advice I hear is to keep the Big Picture in mind when

working on any of these. Then the priority for the Memory Object Model SG would be: first the current TS on pointer provenance, then uninitialized values, then pointer lifetime/end-zap.
Sewell: Thanks to everyone for a constructive discussion! Again, please read the diff and provide feedback, preferably to our mailing list as soon as possible.
Keaton: Sewell, please publish the slides you presented, since they are slightly different than N 2378.

Thursday

- 5.11 Thomas, C23 proposal - TS 18661-3 annex update 2 [N 2579] (slide deck [N 2578])

  Superseded

- 5.12 Thomas, Footnote about sufficient formatting precision [N 2586]

  Championed by Bhakta
  Straw Poll: Does the committee wish to adopt N 2586 into C23 as is?
  13-0-3 accepted

- 5.13 Thomas, Revised N2559 update for IEC 60559 2020 [N 2600]

  Championed by Bhakta
  Krause: In section 17, IEEE 854 was approved in 1987, not 1988.
  Bhakta: Yes, it should be 1987.
  Straw Poll: Does the committee wish to adopt N 2600 into C23 as is?
  16-0-3 accepted

- 5.14 Thomas, C23 proposal - TS 18661-3 annex update 3 [N 2601]

  Championed by Tydeman and Bhakta
  This covers the material in 5.11 (N 2579)
  Tydeman: Should the draft be meeting-based, draft-based or standard-

based?
Ballman: We prefer meeting-based.
Bhakta: Keep this C-standard-based since it has been translated to an annex.
Bhakta: No vote needed, but we can re-vote it in if necessary.
Keaton: We voted it in pending changes. A Re-vote would only be necessary if someone wants it.

- 5.15 Thomas, C23 proposal - edits for infinity and NaN macros [N 2602]

  Championed by Bhakta
  Myers: We do have a more recent paper (N 2617) to fix NAN macros.
  Keaton: This paper is not on the agenda; it was not submitted in time.
  Bachmann: Why do we have to prefix D32/64?
  Keaton: That is what N 2617 was about. When this material moved from math.h to float.h, these prefixes changed to match the convention of the new header. I told Jim to post the document but the change was editorial.
  Straw Poll: Does the committee wish to adopt N 2602 into C23 with changes to the prefixes that are appropriate to the new header? 12-0-5 accepted

- 5.16 Tydeman, DFP triples [N 2580]

  Svoboda: Is the C convention documented in the standard?
  Tydeman: Yes, in section 5.2.4.2, paragraph 2.
  Gilding: This is clarifying the C convention, not changing anything, right?
  Tydeman: Right!
  Straw Poll: Does the committee wish to adopt N 2580 into C23 as is? 13-0-4 accepted
  Bhakta: Why did anyone abstain?
  Pygott: I know nothing about floating-point, so I had no position to express.

Ballman: Same. Also I trust the C floating-point group.
Gilding: Same, but that is why I asked my question.
Svoboda: I usually abstain from floating-point items but I actually understood this one so did not abstain.

- 5.18 Meneide, Not-So-Magic: typeof() for C [N 2593]

  Bhakta: Can typeof apply to a function type? What is the intent?
  Meneide: It is intended to be applicable to function types. Bit-fields are the only forbidden types.
  Wiedijk: If I cast to something that involves a variable-length array where size calls a function with a side effect, is that forbidden or will it be evaluated even if not on the execution path?
  Meneide: The wording should say if the type of operand is a variably modified type, such as a variable-length array, it is evaluated at run time. Otherwise nothing happens at run time.
  Wiedijk: And is this required, even in dead code?
  Meneide: An implementation could choose not to do any evaluation if it can determine proper type at compile time.
  Svoboda: Why has not this been standardized before? It must have some history.
  Meneide: A form of this paper showed up before: N 1229 (by Stoughton), which suggests extensions including typeof. But it did not try to standardize typeof.
  Stoughton: In N 1229, we were looking around for features to consider for standardization.
  Keaton: Right. But each topic needs a champion, and no one volunteered for some topics.
  Gilding: With regard to variable-length array size, it is unspecified if this is evaluated if the size expression does not affect the result. During discussion of _Generic I recall that people shied away from things that were "too close" to C++ templates.
  Meneide: It is not apparent if we can have a computation for imaginary I and have typeof produce const imaginary float…that is what unqual_typeof is for.

Myers: With typeof you could consider whether an lvalue is const or not.
Meneide: We could add that in a subsequent paper.
Seacord: I read the name as "unequal_typeof" not "unqual_typeof". I do not know how to fix this awkward name.
Meneide: I went with "unqual_typeof" for shortness. But we could change typeof to qual_typeof and unqual_typeof to typeof. I do not know if that would help.
Seacord: That would be worse. Next naming comment: _Typeof is inconsistent with _ExtInt. Capitalization is weird.
Meneide: I was trying to follow the practice of _Static_assert. Gustedt suggested taking the keyword ("typeof", "unqual_typeof").
Ballman: How should paper handle type attributes? Are those treated like qualifiers? Can we have examples or make behavior explicit?
Meneide: We are not sure if attributes should stay, but we are leaning in favor of them staying. We can add non-normative wording.
Ballman: there are no standard type-based attributes yet, so this is currently academic. A footnote would be helpful. Perhaps also recommended practice. We both agree that "typeof" should leave attributes in and "unqual_typeof" should strip them.
Gilding: How much implementation divergence is there?
Meneide: People do various things to strip qualifiers. They prefer unqual_typeof. It is a mixed bag of what implementors were trying to do. In corner cases and bug reports involving qualifiers, there is no consensus on what people wanted.
Gilding: Stealing the keyword would be consistent with what we vote into C23 already.
Meneide: I thought we fully accepted promoting everything to keywords.
Uecker: I agree with Alex, we should steal the keyword. Compilers almost always preserve qualifiers, except GCC.
Bhakta: I have seen some divergence, with regard to keeping qualifiers and attributes. Keep _Typeof name because "typeof" has non-portable semantics. I suggest we leave existing platform-specific "typeof" keywords unchanged.

Meneide: Hmm, maybe we should vote on the keyword.
Krause: Introduce _Keyword now, and perhaps in 10-15 years add "keyword". Keep both qualifier-preserving versus stripping.
Myers: If you are concerned with compatibility, perhaps use qual_typeof and unqual_typeof, and do not use "typeof".
Ballman: Is there any expectation that C++ will also pick up this functionality? Or will they have the same problem with _Typeof versus typeof?
Meneide: The header file is for C's usage. C++ could take this, perhaps and get an empty header.
Ballman: Myers' suggestion would solve C++-compatibility too.
Bhakta: For our compilers, we support "__typeof", because it is strictly internal. We also have "typeof" in non-strict mode, which supports some qualifiers and attributes but not others.
Gilding: The divergences seem to be unintentional. I wonder to what extent code breakage would be our problem? This would be an opportunity for us to declare the keyword for a standard truth. Our compiler always strips qualifiers.
Meneide: The main purpose is creating temporary variables, and either strip or choose their own qualifiers. People use casts to tweak qualifiers.
Meneide: I would like a vote about whether we should use a footnote or recommended practice to describe the usage of implementation-defined attributes?
Bhakta: You are looking for something that is non-normative then?
Meneide: We do not need votes on what keywords to use until I work more on the paper.
Keaton: This is opening a can of worms. Perhaps this is a better topic for next time? I recommend you ask for one hour instead of 30 minutes.
Seacord: For attributes for functions, we should make the text normative, as there is no existing practice we would squash.
Myers: One more thing about compatibility: The one area I might like different semantics is for _Noreturn, which is not part of the type.
Meneide: OK, I will consider that for the next iteration of the paper.

- 5.9 Uecker, Compatibility of Pointers to Arrays with Qualifiers (updates N2497) [N 2607]

  Ballman: C++ is trying to get _Atomic to work in C++ as it works in C. Since this paper excludes the _Atomic qualifier, will this introduce C++ versus C incompatibilities.
  Uecker: I do not know what C++ does here. In C _Atomic is technically not a qualifier. Nothing here changes _Atomic (in C).
  Ballman: Is this for "atomic type" or "Atomic(type)"?
  Uecker: For here it is the qualifier, without parentheses.
  Ballman: If this is the one without parentheses, it does not affect C++ at all.
  Straw Poll: Does the committee wish to adopt N 2607 into C23 as is? 17-1-2 accepted
  Seacord: Does it make sense to clean up the Standard language so that atomic is not a qualifier?
  Uecker: I would like to change this about atomic, it is not clear now.

- 5.19 Meneide, Mixed String Literal Concatenation [N 2594]

  Krause: This was implementation-defined in the first place. I do not know any users of our implementation. Perhaps there was a potential use that never came up.
  Meneide: I was hoping someone would catch this paper and claim that string literal mixing was useful. I could not find uses of mixed string literals in open-source code.
  Gilding: I asked our customer base if anyone was using this. It produces compiler warnings on our platform. No one has seen this warning in the wild.
  Ballman: The type confusion from mixed literal concatenation is a loss of information. So closing this hole for security purposes is a good thing.
  Bhakta: We do not support doing this in our implementations either, so no contrary use cases. Hypothetically, would it make sense to have a "u8" prefix followed by a wide string literal with characters present in

a keyboard for people who use that locale?
Meneide: This could make sense, as a way of adding in non-ASCII characters to string literals. If people tried this, they got burned by inconsistent compiler support. We could write a follow-on paper that provides such use cases.
Gilding: Would this also forbid wide literals, non-prefixed literals, and another wide literals?
Meneide: Yes, by transitivity.
Krause: The wording with regard to "adjacent" is confusing.
Meneide: We could add working along the lines of: "UTF8 string literal, nor shall they have a qualified string literal with a different prefix". Also, "Adjacent wide string literal" could be written better.
Bhakta: Why can narrow string literals not concatenate? They currently can. I think the current words are good.
Krause: Plain ones without prefix, wide ones with prefix, utf8 prefixes.
Keaton: Perhaps we should vote on the document as is, and then revise later, or if vote fails.
Straw Poll: Does the committee wish to adopt N 2594 into C23 as is?
18-0-2 passes

Friday

- 5.20 Wunsch, C23 proposal: formatted input/output of binary integer numbers [N 2612]

  The paper has been updated here: http://www.open-std.org/jtc1/sc22/wg14/www/docs/tmp/n2618.pdf
  Keaton: Are there any objections to covering this paper? (None)
  Krause: There is precedent in other languages but not in C. Do C programmers really need this?
  Wunsch: We believe that the presence in other languages indicates people want this feature.
  Ojeda: It is a useful feature to add. It is also easy to implement, so perhaps C programmers currently roll their own.

Ballman: It is strange that we currently support hex and octal but not binary. We could not find any C platforms that support %B. So I prefer option #2.
Gilding: Lack of prior art means a vicious cycle; it does not mean no one will use this. I suspect uptake would be immediate.
Krause: I do not see option #1 as advantage over option #3. In our source code "b" is used, I do not know what for.
Bhakta: "B" or "b"?
Krause: "B".
Ballman: I am uncomfortable with a recommended practice. Recommending the practice does not buy programmers anything. If "B" is implementation-defined, that does not increase portability.
Bhakta: I view recommended practice as advice for implementers not programmers.
Gilding: Option #1 is scary, because if "%B" consumes some large differently-sized type (than int), this introduces subtle portability error.
Wunsch: That is why options #2 and #3 exist.
Bachmann: In the future, we will probably be forced to use uppercase letters (as format specifiers) because we are running out of lowercase letters.
Wunsch: In that case, we should deprecate the use of capital letters in the standard (reserving them for platform-specific extensions)
Gilding: Leaving "%B" out now gives us the choice to add it later.
Myers: We are already using new uppercase letters in C23 ("D" and "H"). C99 also used new uppercase letters ("A" and "F"). Also, we could use lowercase letters with an escape should we run out.
Krause: "H" has similar problems. We say we reserve capital letters and then break this promise.
Myers: I think everything outside the basic character set should be left to extensions.
Bhakta: Agreed. But I also agree with Krause that we do not take away from implementations (e.g.: standardize capital letters) So option #2 is out. And option #1 would reduce confidence in C due to portability.
Meneide: I figure that the best way to standardize identifiers is not the best way forward. Instead, we should spend time on a better printing

function that is more type-safe and extensible. This is a good way out of our current problem. For now, I would choose option #2 or #3.
Gilding: I agree that printf() and format strings should be superseded. But a new function could never replace printf(), so this problem would outlive a replacement function.
Myers: With regard to alternatives: printf() does work better than many alternatives with internationalization because arguments can be reordered for different format strings.
Keaton: What if we did option #1, with fair notice that a future standard would go to option #2?
Bhakta: I disagree. It would be better to focus on current implementations. A printf() alternative is a sidetrack.
Gilding: Leaving it implementation-defined forces platforms to specify their behavior, and makes it unreliable and non-portable.
Myers: If we want option #1 over #2, we should use "recommended practice" to specify #1.
Svoboda: When we do straw polls, please only do binary voting. We spend more time discussing how to do trinary or more voting properly than we spend doing the actual voting. So do something like option #1 versus #2 or #3, please.
Ballman: Are we voting this into the standard right now?
Keaton: No, this poll is option #3 versus #1 or #2.
Straw Poll: Does the committee prefer something along the lines of option #3, instead of #1 or #2, for N 2618? 15-2-3 clear direction to pursue option #3
Bhakta: We can talk, but I do not want this into C23 without time to go through other peoples' feedback.
Keaton: You are right, this paper is not officially on the agenda. Give Wunsch time to update the paper and strip out options #1 and #2.
Myers: Should we do an "along the lines of" straw poll?
Svoboda: Let Wunsch decide if he needs more clear direction, in the form of a straw poll.
Wunsch: I have direction, I will submit a new paper.
Keaton: Do people want a non-normative mention of "B"?
Wunsch: Perhaps for the rationale?

Keaton: We do not have any volunteers to write the rationale.
Straw Poll: With regard to N 2618, does the committee want non-normative wording mentioning capital "B" for this purpose? 6-3-11 no clear direction
Keaton: Wunsch, it is up to you if you want to add wording.

6. Clarification Requests

The previous queue of clarification requests has been processed.

7. Other Business

7.2 Meneide, Restart-able and Non-Restart-able Functions for Efficient Character Conversions [N 2595]

Banham: Why change from multi-byte to multi-character and multi-wide-character?
Meneide: One problem with the previous design assumed only one character (or wide character). HKCS has to return chars outside the standard which hurts portability. The new interface is fashioned after the style of "iconv()".
Banham: A single wide character might map to more than one regular character.
Meneide: It was outputting two UTF32 code points, you could use either in a render-er. We wanted a new set of functions where we could modify inputs, outputs, and return values. And we could convert from multi-character inputs that map to one or multi-character outputs. For now, I am only proposing the conversion functions.
Myers: I cannot follow what "code unit" means at all.
Meneide: We need a better definition for things like "code unit". Every function call works on indivisible units of work. A code unit is the type (char16_t, char32_t, char, wchar_t) of input. Input is typically a sequence of code units.
Myers: I am unclear about the current maximum minimum values for

output.
Meneide: I was trying to preempt an implementation from defining these things to be too small. If they have to increase these numbers, they will be overwriting buffers that were not big enough. So I set these minimums to be rather high.
Krause: We do not want to allocate so many bytes because the standard is being cautious. Also, the C to UTF conversion functions have been left out.
Meneide: I could perpend STDC_ to standard C function names. But I am concerned about some platforms' 32-char limit on unique function names.
Ojeda: Are you planning to bring UTF to UTF once?
Meneide: That depends on if people want it. I wanted minimum changes to fix narrow versus wide encodings and Unicode to narrow versus wide encodings. It might be better to provide UTF functions anyway.
Ojeda: Have you considered Windows functions to have an error when a character cannot be converted. Or substitute with a replacement character?
Meneide: The function stops if there is an error. It increments input pointers and decrements sizes so you can see how much of the buffers were produced and consumed. This is the lowest-level API you can design to handle these sorts of things. I did not want to impose any specific error-handling behavior.
Myers: What about the UTF to UTF conversions?
Meneide: I could add it if people want. Perhaps we should vote to see if people want them.
Bhakta: I thought we were discussing a future paper, not an update.
Keaton: Good point. The concepts are separable so paper should be separable too.
Straw Poll: In the context of character conversions, would the committee like to see a future paper that includes UTF-to-UTF conversion functions?
14-0-6 clear direction in favor
Action Item: *Meneide*: Write a proposal to write a paper suggesting UTF-to-UTF conversion functions.
Banham: The standard, including this paper has to be a lot clearer than it is on the definition of a character, especially the width (that is, range of values) of a character.
Meneide: C++ went on a Unicode tear to better address character sets. But

that will be a separate paper. The phrase "Wide character set" is in the standard, but the term "narrow" is not.
Seacord: For some functions we were saying "character" instead of "code unit". After committee discussion, there were three different definitions of "character" in the standard, but you had to know which definition each use of "character" employed. I was told you have to fix the entire standard to address this. This is a big problem with a lot of effort required.
Gilding: In Ithaca, the question was asked for a precise definition of "character", and the answer was that it was ambiguous and followed the English meaning.
Keaton: There is a tool, "wkhtml2pdf", that preserves search-ability and links in the resulting PDF. I have had problems with "pandoc". The new ISO document system forces conversion to PDF. But Plakosh has found an option to preserve HTML. So you can send Dan raw HTML, but let Dan know when you request document numbers if you plan to submit your document as HTML.
Krause: What is the plan on ending N-space pollution?
Keaton: It is on hold, because Plakosh is overwhelmed, so he wants to delay the conversion to C documents. Also Meneide is working on software to allow document submission.
Meneide: Yes, I hope to have that software ready by February.

The following papers have not come in yet, so we will not discuss them

- 7.1 Seacord, Defer Mechanism for C [N 2591] (slide deck [N 2589])
- 7.3 Working draft updates
- Meneide, C23 Working Draft [N 2596]
- Meneide, C23 Working Draft - Diffmarks [N 2597]
- Meneide, C23 Working Draft - Editor is Report [N 2598]
- 7.4 Gilding, Review and comparison of existing extensions and practice for functional programming in C [N 2604]

8. Resolutions and Decisions reached

8.1 Review of Decisions Reached

Does the committee wish to adopt N 2572 with editorial changes into C23? 21-0-0 adopted.
Does the committee want something along the lines of N 2606 to be added to C23? 21-0-1 looking good
Do we want to allow #embed to appear in any context that is different from an initialization of a character array? 5-8-6 leaning in the direction of no but not clear
Does the committee wish to adopt N 2587 into C23 as is? 8-11-3 no consensus
Does the committee wish to require that a least integer type matches an exact integer type when both are defined in C23? 19-1-3
Does the committee wish to add a second letter to indicate "exact" or "least" to the width modifier in N 2587? 8-5-8
Does the committee wish to adopt N 2576 into C23 as is? 4-8-10 no sentiment to add
Does the committee prefer changing the name from ExtInt to BitInt in N 2590? 12-0-8 there is sentiment to do that.
Does the committee prefer splitting N 2590 into a "bare-minimum" paper and an ancillary paper? 11-0-8 there is sentiment to do that.
Does the committee wish to adopt something along the lines of alternative 3 of N 2599 into C23? 16-1-6 clear direction.
Does the committee wish to preserve "const" across the functions listed in N 2603? 13-1-8 leaning to yes
Does the committee wish to adopt something along the lines of the first syntax option in N 2603 into C23? 10-4-7 leaning to yes
Does the committee wish to adopt N 2586 into C23 as is? 13-0-3 accepted
Does the committee wish to adopt N 2600 into C23 as is? 16-0-3 accepted
Does the committee wish to adopt N 2602 into C23 with changes to the prefixes that are appropriate to the new header? 12-0-5 accepted
Does the committee wish to adopt N 2580 into C23 as is? 13-0-4 accepted
Does the committee wish to adopt N 2607 into C23 as is? 17-1-2 accepted
Does the committee wish to adopt N 2594 into C23 as is? 18-0-2 passes

Does the committee prefer something along the lines of option #3, instead of #1 or #2, for N 2618? 15-2-3 clear direction to pursue option #3
With regard to N 2618, does the committee want non-normative wording mentioning capital "B" for this purpose? 6-3-11 no clear direction
In the context of character conversions, would the committee like to see a future paper that includes UTF-to-UTF conversion functions? 14-0-6 clear direction in favor

8.2 Review of Action Items

Keaton: Change Stoughton to represent Austin Group rather than Free Standards Group.
Meneide: Write a proposal to write a paper suggesting UTF-to-UTF conversion functions.

10. Thanks to Host

10.1 Thanks to ISO for supplying Zoom capabilities

11. Adjournment (PL22.11 motion)

Keaton: WG14 does not require a motion to adjourn.