Draft Minutes for 23 Jan – 28 Jan, 2023

MEETING OF ISO/IEC JTC 1/SC 22/WG 14
WG 14 / N 3116

Dates and Times:

Monday,	23	January,	2023	14:30 – 18:00 UTC
Tuesday,	24	January,	2023	14:30 – 18:00 UTC
Wednesday,	25	January,	2023	14:30 – 18:00 UTC
Thursday,	26	January,	2023	14:30 – 18:00 UTC
Friday,	27	January,	2023	14:30 – 18:00 UTC

Meeting Location

Teleconference

1. Opening Activities

1.1 Opening Comments (Keaton)

Keaton: due to ongoing difficulties caused by Long Covid, will be withdrawing as Convenor after this term. The new appointee will take over in September. Keaton will remain a member and remain as SC22 chair.

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Notes
Aaron Ballman	Intel	USA	C++ Compatibility SG Chair
Alex Gilding	Perforce	USA	
Barry Hedquist	Perennial	USA	INCITS/C IR
Clive Pygott	LDRA Inc.	USA	WG23 liaison
David Keaton	Keaton Consulting	USA	Convener
David Svoboda	SEI/CERT/CMU	USA	Undefined Behavior SG Chair
David Vitek	Grammatech	USA	
Douglas Teeple	Plum Hall	USA	
Elizabeth Andrews	Intel	USA	
Fred Tydeman	Keaton Consulting	USA	INCITS/C Vice Chair
Freek Wiedijk	Plum Hall	USA	
Paul McKenney	Meta/Facebook	USA	
Rajan Bhakta	IBM	USA, Canada	INCITS/C Chair
Robert Seacord	NCC Group	USA	
Aaron Bachmann	Austrian Standards	Austria	Austria NB
Dave Banham	BlackBerry QNX	UK	MISRA Liaison
Debbie Spittle	INCITS	USA	INCITS Secretariat
Eskil Steenberg	Quel Solaar	Sweden	Sweden NB
Geoff Clare	The Open Group	UK	Invited Guest
Jakub Lukasiewicz	Motorola Solutions Systems Polska	Poland	Poland NB
JeanHeyd Meneide	NEN	Netherlands	Netherlands NB
Jens Gustedt	INRIA	France	France NB
Joseph Myers	CodeSourcery / Siemens	UK	UK NB
Luigi Liquori	INRIA	France	Invitied Guest
Martin Uecker	University of Goettingen	Germany	
Michael Wong	Codeplay	Canada, UK	WG21 Liaison
Miguel Ojeda	UNE	Spain	Spain NB
Niall Douglas	Ireland ned Productions Ltd	Ireland	guest
Nick Dunn	NCC Group	USA	
Nick Stoughton	Logitech	USA	SC22 Austin Group Liaison
Ori Bernstein	PingThings	Canada	
Peter Sewell	University of Cambridge	UK	Memory Model SG Chair
Rajan Bhakta	IBM	USA, Canada	INCITS/C Chair
Roberto Bagnara	BUGSENG	Italy	Italy NB, MISRA Liaison
Ville Voutilainen	The Qt Company	Finland	Finland NB

1.3 Procedures for this Meeting (Keaton)

As usual, we hold straw polls instead of formal votes; guests may vote. All we do is make recommendations to our parent group.

We do *not* want multiple conversation threads; do not split the audio and chat. Do not have more than one conversation at once; respect the floor. The chat is an additional medium for backup and enhancements (such as URLs) only. The chat is not minuted unless being used as a substitute for audio.

1.4 Required Reading

1.4.1 ISO Code of Conduct

1.4.2 IEC Code of Conduct

1.4.3 JTC 1 Summary of Key Points [N 2613]

1.4.4 INCITS Code of Conduct

Reviewed without comments.

1.5 Approval of Previous Minutes

WG 14 Minutes [N 3044] (WG 14 motion): Tydeman moves, Ballman seconds. No objections.

INCITS/C Minutes [pl22.11-2022-00007] (INCITS/C motion): Ballman moves, Pygott seconds. No objections.

1.6 Review of Action Items and Resolutions

No outstanding Action Items.

1.7 Approval of Agenda [N 3087]

(INCITS/C motion, WG 14 motion): Tydeman moves, Ballman seconds. No objections.

1.8 Identify National Bodies Sending Experts

Austria, Canada, Finland, France, Germany, Ireland, Italy, Netherlands, Poland, Spain, Sweden, United Kingdom, United States.

The Linux Foundation and the Austin Group are also represented.

1.9 INCITS Antitrust Guidelines and Patent Policy

Guidelines were observed without discussion.

1.10 INCITS official designated member/alternate information

Check with Bhakta if unsure.

1.11 Note where we are in the C23 schedule [N 2984]

It is advisable to have a second CD ballot due to the volume of comments. The schedule is in flux; we will resolve this ballot and update afterward. Any objections to a second CD ballot?

Gustedt: well, try it first, maybe there will be nothing to resolve?

Gilding: will this push publication to 2024?]

Keaton: we are within the allowed schedule which ends in mid-2024. Failing a DIS ballot would be much worse for the schedule than adding an extra CD ballot.

DECISION: a second CD ballot will be held for ISO 9899:2023.

2. Reports on Liaison and Collaboration Activities

2.1 ISO, IEC, JTC 1, SC 22

No report

2.2 INCITS/C / WG 14

Bhakta: nothing from INCITS.

2.3 INCITS/C++ / WG 21

Ballman: a C++ meeting will take place in February; we are on schedule for C++23.

2.4 INCITS/Programming Languages

No report

2.5 WG 23

Pygott: nothing from WG23

2.6 MISRA C

Gilding: Amendment 4 has been finished and will be published in March.

2.7 Austin Group

Stoughton: the next version of POSIX is now feature-complete and in CRB ballot. The DIS ballot will take place this year for a revision based on C17.

2.8 Other Liaison or Collaboration Activities

No other reports

3. Study Groups

3.1 C Floating Point Study Group activity report

Bhakta: handling comments from the CD and questions from the community. We also have work in the pipeline for after the ballot.

3.2 C Memory Object Model Study Group activity report

Gustedt: have worked on replies to the draft TS; it is debatable whether they are editorial or not.

Keaton: A CD is optional for a TS, but so many changes were demanded by ISO that Jens didn’t want to go straight to DTS.

Gustedt: there is a document based on the diffs; ISO want a normative, real document. I personally want WG14 to review it.

Sewell: two questions – should ISO review it? Yes. Does this need a CD to happen? It is a non-trivial review task.

Seacord: which version is this based on?

Sewell: C17. The review is smaller than it would be for C23.

Keaton: a CD ballot costs two months, but there is time in the schedule.

Sewell: latency costs in industry enthusiasm.

Keaton: any objections to a CD ballot for TS 6010?

(none, unanimous consent)

DECISION: Gustedt's document will go to SC22 and start the two-month ballot process this week. One month available if needed for ballot resolution.

Sewell: can we start with ISO working in parallel?

Keaton: yes, ISO has volunteered to start its review early.

ACTION: Keaton to submit TS 6010 to ISO early.

ACTION: Gustedt to make up an N-document for TS 6010.

3.3 C and C++ Compatibility Study Group activity report

No report

3.4 Undefined Behavior Study Group activity report

Svoboda: still proceeding, with fewer meetings. Expect some proposals and a TR for October, or later if the C23 schedule requires.

4. Future Meetings

4.1 Future Meeting Schedule

13-17 February, 2023 (proposed virtual continuation of CD 9899 ballot resolution, if needed)

17-21 April, 2023 (proposed virtual CD 6010 ballot resolution meeting)

Sewell: really don't expect to need a week for this.

Gustedt: a day, more like. Do people prefer Tuesday-Wednesday? The whole week is still reserved.

October, 2023 (proposed, depending on DIS schedule) – DIS Ballot resolution meeting

Keaton: we can't confirm this until we have a new schedule.

4.2 Future Mailing Deadlines

Post-Virtual-202301/202302, Pre-Virtual-202304 – 17 March, 2023

Post-Virtual-202304 – 12 May, 2023

These mailing deadlines remain in force.

5. Document Review

Keaton: the purpose of this meeting is CD 9899 ballot resolution. We will review N3067 sequentially, except for the Austin Group comments, and additional (unofficial) comments in N3073. Editorial comments will be added to a mass-approval list to vote on at the end.

The following comments will be covered at the beginning of the Tuesday session, to receive assistance from the Austin Group:

> GB-190, GB-209, GB-232, GB-238 (NL-239), GB-240, GB-245

We cannot finish in two weeks without limiting each comment to 15 minutes.

(Comments with no discussion are not included in this section)

Monday

SE-001 (unnamed parameters)

Steenberg: this is really two comments, more concerned about N2510. Do not believe the solution works, only applies to the definition; with no link to the declaration and not communicated to the caller, it is broken. Proposal predates attributes and should use that instead. With no performance impact there are thin advantages. Removes an opportunity for a warning.

Ballman: this standardizes existing practice.

Gilding: but parameters in declarations are already often unnamed?

Steenberg: that's not meaningful – the caller can do something different upon seeing the definition. There is asymmetry on the inlining impact, an attribute would carry this.

Ballman: this mostly impacts callbacks for which the signature and ABI are not user-controlled. Users don't put the information in the signature, it is local to the callback.

Steenberg: the signal is local to the function; the pointer doesn't know which function is called, so there is no optimization, so this is only useful to functions called directly. There are better options available.

Bernstein: this is purely syntactic. There is no implication it *should* affect ABI.

Voutilainen: this isn't mentioned anywhere as being optimization-oriented – the caller can't elide arguments.

Straw poll: (decision) Should WG14 accept SE-001, first part referring to n2510 only?

Result: 3-16-5 (no consensus)

Steenberg: I am happy to wait to talk about `auto` later alongside other comments.

GB-004 (normative reference requirements)

Banham: this is a question about whether the references comply? Are we allowed to *normatively* reference non-ISO Standards (i.e. Unicode)?

Keaton: would submit it as-is, it doesn't affect understanding. There are no complaints from the ISO editors and this is easy to fix in DIS.

Banham: we must have a *record* of permission to cite.

ACTION: Keaton to get permission from the Unicode Consortium to cite their Standard.

CA-006 and other trigraph comments

Meneide: this is part of a collection of related comments with multiple solutions we can discuss all together. Canada's unconditionally reverts the removal of trigraphs; Netherlands deprecates them; US would restore with digraph semantics.

Ballman: removal doesn't make the `#pragma` impossible, because phase 1 is completely implementation-defined. This is well-discussed.

Seacord: same observation – recent approval, very popular change.

Gustedt: surprised by *this* comment because C23 will definitely have character sets that can do this! And it still has digraphs.

Bhakta: these points are addressed – digraph characters aren't invariant, can't be used this way. Only phase 1 can do it. Theoretically the # is the only one we need, to engage the `pragma`.

Gustedt: the digraph for # is non-moving in all known codepages.

Bhakta: no it's not.

Voutilainen: is the implementation materially different between C and C++s? IBM did not object to removal from C++.

Bhakta: it's not more difficult, there were objections in C++ but at the time the belief was that C++17 was just more important. In practice compilers simply left trigraphs enabled and chose to be non-conforming.

Gustedt: `:%` appears to be an EBCDIC invariant.

Ballman: but phase 1 is implementation-defined, so why is support problematic? At the moment support is forced on all compilers.

Bhakta: phase 1 reads and interprets the bytes – so when `??=` is seen, the byte representation for a # is produced. This changes the meaning within strings, it would not be possible to print `"#"`. The Standard mandates the internal representation.

Ballman: the C part is far away and separated from phase 1? It maps bytes to a character set and applies within strings. WG21 believe this is permissive and still allows the byte sequence `??=` to be "a funny multibyte character".

Seacord: the proposal aimed to fix a defect. These don't fix it.

Tydeman: phase 1 doesn't know the difference between a trigraph in a directive or in a string.

Seacord: the Standard allows the reading of a binary and the mpping to characters to be implementation-defined. While reading `??=` off the disk, it can be the binary representation of #. It *will* still work in a string, this is not an obstacle.

Ballman: confused – *only* digraphs are distinct inside and outside strings. A straight replacement in phase 1 replicates the exact semantics of trigraphs.

Steenberg: trigraphs change the meaning of string literals, you can't trust the literal to know what it should mean.

Meneide: you can use them accidentally, but this already only happens sometimes. The implementation-defined phase 1 makes this unreliable in strings. This is a portability issue not just for tokens but for the source text itself. The US comment evades this by at least guaranteeing consistent string behaviour.

Keaton: (we are looking at all trigraph comments together)

Bernstein: code including the ASCII # needs translation anyway. Treating it as an encoding detail is fair for portability.

Ballman: *no* versions of Clang, GCC or MSVC treat trigraphs as on-by-default. The portability is not real. Clang will continue to treat them as optional but this boils down to adding implementation burden.

Bhakta: the arguments that it is a problem and that it is not used are contradictory. The general idea is that portability of code should be promoted by the Standard. "Ancient" is a pejorative; stability is a benefit of C, not a problem. A lot of things aren't in Standard mode by default, which is fine, but many vendors do use this, and we do want to take the broadest possible view. The local burden is never zero but not major to change.

Ojeda: given the digraph semantics option, could an implementation define a mapping that the file determines for phase 1?

Bhakta: we did investigate this, choosing automatically. Performance was not a huge overhead, but has to be heuristic and the ambiguous cases are pathological.

Ballman: agree that this is possible, allowed, and not practical.

Straw poll: (opinion) Does WG14 want to include trigraphs in C23, in any form?

Result: 7-16-3 (no consensus)

Straw poll: (decision) Should WG14 REJECT comments CA-006, US15-048, NEN/NL1-049, US18-050, US17-051, US16-052?

Result: 20-4-3 (rejected)

GB-007 (definition of single and double precision)

Myers: prefer to use the CFP wording.

Straw poll: (decision) Should WG14 accept GB-007 with the changes from N3082 applied?

Result: 21-0-2 (accepted)

US 3-011, GB-039 (dollar sign)

Seacord: this is part of a three-part change as an action item on $. Would require changes by section. We have consensus to make a change but not on the exact wording.

Straw poll: (opinion) Does WG14 prefer the wording from Alternative 1 in n3046, without the word "defined"?

Result: 14-0-10 (yes)

Straw poll: (decision) Should WG14 accept US3-011, without the word "defined"?

Result: 18-0-5 (accepted)

Straw poll: (decision) Should WG14 accept GB-039 with the proposed resolution?

Result: 20-0-2

GB-012 (six digit Unicode identifiers)

Gustedt: object, we only have the eight/four digit syntax. This is confusing.

Banham: we'll need to explain the discrepancy.

Straw poll: (decision) Should WG14 accept GB-012?

Result: 3-7-12 (no consensus)

Myers: the present wording is not consistent with the normative reference.

Bhakta: resolve the clash in a footnote.

Seacord: the link isn't even valid for the reference; the normative reference is therefore to the "2000" version. This feels like a defect that needs another fix. The syntax may be wrong.

Ballman: at some point stuff like this will go in the issue tracker.

ACTION: Seacord to write a paper on the issue highlighted by GB-012 as a second-round comment.

Gustedt: the solution can't be to use six-digit codes – this breaks code using the eight-digit syntax, can't have the capital `U`.

US 4-013 (split implementation)

Ballman: filed as implementor feedback. The Standard considers the whole toolchain, but Clang's relationship with Windows and with GCC means it doesn't usually control the library, so the existence of macros is not reliable.

Gustedt: the goal was to allow this divergence and have the library set the rules.

Ballman: given that integer types may be different, functions like `printf` can have different argument types – not a new problem.

Gustedt: a library is built with a compiler and takes definitions from it. For bit-precise integers, both the compiler and the library must support them.

Ballman: the value from the macro isn't wrong, but we can't correctly answer for all libraries. Compiler-inserted shim headers fix this but aren't practical.

Gustedt: it's not precise enough to revert.

Myers: N2359 does two things – adds version macros and removes `WANT` macros. This change would do both. There are two use cases for version macros – the language version supported and the headers shipped separately, for use against multiple versions. Cannot know how it is answered.

Straw poll: (decision) Should WG14 accept US4-013?

Result: 2-9-10 (no consensus)

Gustedt: there were other features too, can't pull it out just like that.

Ballman: it's only the feature tests we care about. Implementations can live with it, that's not a new problem, only an exacerbated one.

GB-028 ("plain" `char`)

Seacord: prefer to remove the quotes.

Bhakta: unclear comment? But I want them to match what we do for `int`.

Banham: prefer quotes for idiomatic use.

Seacord: I consider this incorrect grammar and would remove from all uses.

Bhakta: do not like the expansion of comment scope.

Straw poll: (opinion) Does WG14 prefer to quote "plain" in "plain char" etc.?

Result: 8-10-2 (no consensus)

Meneide: as the editor, emphasis helps but the text is not worse or lose meaning if the quotes are removed.

Tuesday

Seacord: I like quotes because "plain" is a placeholder for "non-specified"; without quotes it appears there is an associated concept.

Svoboda: it could also mean a non-wide `char`, we're not going to solve all cases.

GB-190 (excessively restrictive signal handlers)

Defect report to the Austin Group. Programs getting UB by reading objects defined as constant and similar things that aren't specified as safe, but logically should be well-defined.

Straw poll: (decision) Should WG14 accept GB-190?

Result: 3-9-9 (no consensus)

Stoughton: the proposed change is towards expected behaviour.

Seacord: we tried to improve this before – the conclusion was that this is all *so* un-portable that we regretted the inclusion of signals at all. No interest in fixing it, even the guarantees are non-portable.

Ballman: this is an incompatible change with C++, does not allow reads of `const` objects that may not be the definition. C++ *does* allow calls to Standard Library functions marked as signal-safe.

Gustedt: `const` is not adequate, also needs to be non-`volatile`, even that may not be enough. “Happens before” is not the right relation. This wording is too fragile to alter this way.

Myers: reasonable wording exists, although not clear what to do about pointed-to values. C++ excludes `thread_local` objects, the comment is more restrictive.

Ballman: we should track the issue though.

GB-209 (unintentional stdio deadlock)

Clare: this unintentionally allows a deadlock.

Gustedt: in favour in principle but this is too late, needs a full paper and discussion.

Keaton: these are all things for our issue tracker.

Bachmann: can we defer this so someone can write a full paper?

Keaton: we can only accept or reject the comment; our permission is not needed to write up a paper.

Stoughton: would this fit in the second CD?

Keaton: yes.

Straw poll: (decision) Should WG14 accept GB-209?

Result: 3-11-4 (no consensus)

GB-232 (two’s complement vs. `strtol`)

Straw poll: (decision) Should WG14 accept GB-232?

Result: 15-1-1

Bhakta: what does “negation” mean here? Negation is `!`.

Keaton: existing wording does not specify “arithmetic” negation.

Myers: ISO 2382 uses booleans for the definition of negation.

Gustedt: I agree with Joseph but we can omit the sentence anyway, it’s implicit.

Bhakta: I see this change as making the problem worse.

Gilding: don’t think it is worse, but would like an action to fix all cases in a new comment.

ACTION: Bhakta to propose alternative wording to GB-232 in an n-document.

ACTION: Gilding to review all uses of “negation” in the Standard.

GB-238 (data races in `mblen` etc.)

GB feel that aligning with POSIX is more helpful, POSIX will *not* align with C17.

Keaton: what are the consequences?

Seacord: working functions will stay working, so can’t imagine any.

Ballman: this should be added to the Papers of Interest list.

Bhakta: the impact is that some users will need to add locks?

Clare: users need this because that's not current practice.

Bhakta: are there any known examples?

Clare: no but this was all written before the introduction of threads, to use statics internally.

Steenberg: this is better than `realloc` because it's not a breaking change. Thread-safe implementations will continue to be fine.

Gustedt: we put effort into wording this; what is the procedure to change any function POSIX also specifies?

Keaton: we aim to standardize practice. We don't have to follow POSIX, but if practice diverges, we have a choice.

Seacord: would adding "implementation-defined" change the meaning?

Stoughton: it doesn't help users write POSIX-portable code, but is acceptable.

Bhakta: if we do that, you need the sync anyway. This adds the cost for all users, not just for POSIX-portable users.

Clare, Stoughton: we are not opposed to implementation-defined.

Keaton: a feature-test macro would help users, but needs a paper for wording. *(silence)*

Straw poll: (decision) Should WG14 accept GB-238, substituting "it is implementation-defined whether these functions avoid data races with other calls to the same function" as the proposed text?

Result: 17-3-1

NL-239 (`nsec_t`)

Meneide: needed Austin Group approval, now caught up. All implementations currently use `long`. Just makes the type implementation-defined.

Myers: `long` can always do this, but there is an ABI issue with 32-bit vs 64-bit. The motivating example in N3066 is not a problem and was fixed as a Linux kernel bug five years ago. Other cases need examples, not possibilities.

Bhakta: strongly in favour – AIX had similar incompatibility moving to `long` and hit an ABI issue.

Meneide: swapping the order of padding is needed when the endianness changes across the ABI. Linux is illustrative but other platforms still benefit.

Seacord: a lot of `_t` types never say they're implementation-defined, just "a type capable of representing…". Implementation-defined is wrong.

Meneide: `nsec_t` is no longer there and the comment notation seen elsewhere is used instead.

Straw poll: (decision) Should WG14 accept NL-239 using the wording proposed in n3066?

Result: 13-2-7

Gustedt: why is it signed?

Meneide: it was already and we were told to leave it.

Gilding: the Austin Group votes No?

Stoughton: our approval was an "...I guess", not enthusiasm. It will still be a `long`, which is conforming.

GB-240 (`(time_t)-1`)

Clare: this dates to a DR from 1994! POSIX has always forbidden this, but one BSD does it anyway. Every certified system is tested and doesn't do this. FreeBSD behaves correctly, NetBSD doesn't, overflowing instead. We couldn't find any other interpretation.

Banham: it's unclear if `time_t` is signed; is it a cardinal count of seconds? Casting -1 usually appears with `size_t` etc.

Clare: it can also be a real floating type, or even a bitfield.

Gustedt: would prefer a full paper.

Clare: this is the full proposed text.

Bhakta: I favour a change, but am not sure which. We do have the wording we need.

Stoughton: the options are in preference order, so start with no. 1.

Myers: is there a difference between 1 and 2?

Stoughton: only in their interpretation of C17.

Straw poll: (decision) Should WG14 accept GB-240 using wording option 1?

Result: 10-0-10

GB-245 (year format conversion)

Myers: obvious that `%F` claims untrue conformance – it is only true for 1000-9999, otherwise the digits are incorrect.

Gilding: there is no year zero in the Common Era, don't accidentally require it.

Gustedt: options 2 and 3 are normative; option 1 is OK.

Clare: option 1 is the weakest, can we fallback?

Straw poll: (decision) Should WG14 accept GB-245 using wording option 1?

Result: 12-2-7

Bhakta: what does this mean for years before 1000?

Stoughton: you will print a number that might not conform to ISO 8601.

GB-028 continued

Bhakta: according to the 1995 Merriam-Webster, quotation marks enclose borrowed, special, or informal words; the last two cases fit. Therefore, quoting is correct.

Seacord: Grammarly says in idiomatic usage, quotes indicate opposed usage, i.e. "not plain".

Lukasiewicz: as a non-native English speaker, any of the four usages are clear.

Ballman: this is purely editorial and not worth Committee time.

Straw poll: (decision) Should WG14 accept GB-028, by adding quotes around "plain" where they are missing before "char"?

Result: 15-3-4

Keaton: I would like to know the editor's preference and vote on that.

Meneide: I would add them where they are missing and come back to reconsider this later.

GB-029 (definition of "binary")

Gustedt: prefer to keep the NOTE as a reminder.

Gilding: is this observable at all? These are all specified purely arithmetically.

Bhakta: correct, but they are intended to be usable by control blocks.

Seacord: the second sentence is not part of the definition.

Banham: the sentence about bytes is fine, just the first sentence is superfluous. But happy to keep it.

Stoughton: this repeats paragraph 5 anyway, no need for the second sentence.

Straw poll: (decision) Should WG14 accept GB-029, by deleting the first sentence of NOTE 1 in 6.2.6.1?

Result: 9-5-9

DE-031 (composite type rules)

Uecker: the algorithm for composite types is not complete for all inputs. Missing rules for the newly-compatible structure and union types. This uses "same type" without defining that term.

Gustedt: the wording is not fully usable – in all cases so far the composite is an existing type – for the `enum` case, it doesn't say if it's one of the two, or some third type. Some implementations choose the integer type, do we constrain this?

Uecker: for the first question, "same" is unclear. The new type may have more information than either side. Merged information gets some type we can write the name of. Such arrays can appear in structures, and composing two `struct`s may produce a third `struct` type with information from both members merged. On enums, I expect an equivalent to either the first or the second input type, there is no observable difference except "same type" which is poorly defined.

Gustedt: we should just say it's one of them. What do implementations do?

Uecker: would prefer to force the `enum` type. We could make it implementation-defined, right now it is unspecified.

Myers: we also ought to specify the ordering of composite union members.

Bachmann: what happens when composing explicitly-sized enums with different sizes?

Gustedt: not compatible anyway.

Straw poll: (decision) Should WG14 accept DE-031?

Result: 0-17-3 (no consensus)

Seacord: now that we have typed and untyped enumerators, we need to be clear on how untyped `enum` types are handled. There is always exactly one compatible integer type, but it is not known which.

Gustedt: the source of the rule is that the same object may be defined with an integer type and `enum` across different TUs. This only makes sense for a compatible underlying type.

Keaton: we need a paper to work through this wording and apply it to the second CD.

Straw poll: (opinion) Would WG14 like to see a paper along the lines of DE-031?

Result: 20-0-2

US 9-034 and other nullptr comments

Gustedt: the omission of `(void *)nullptr` was intentional.

Ballman: I agree and withdraw the NB comment.

Uecker: what is the impact on C++ compatibility?

Ballman: this isn't a problem.

Straw poll: (decision) Should WG14 accept US9-034?

Result: 0-15-5 (no consensus)

Ballman: thank you Jens, this is appreciate and we support the proposed resolutions to US10 and US24. US22 is not worthwhile.

Straw poll: (decision) Does WG14 want to resolve US10-035 and US24-061 using the text in n3077 sections 2.1 and 2.2, and by removing footnote 126 in CD ballot document N5777?

Result: 17-0-6

Straw poll: (decision) Should WG14 accept US22-058?

Result: 0-17-4 (no consensus)

Meneide: so there was pushback from implementers about `nullptr` being necessary at all. Some were still using plain `0`. Although people prefer `void *`, the industry wasn't forced to move. Many implementers from TI through to Oracle responded with their definitions of `NULL`. `0` and `0L` are in use; we concluded that `nullptr` is needed but must follow up. Some implementers switched to a `void *` definition as a result of taking the poll. We should consider forcing the typed version of `NULL`.

Uecker: definitely want to fix `NULL`, have had proposals to deprecate it in the past. Once it is fixed, it is not less useful.

Ballman: Clang is considering a move to `nullptr`, but it's slow work. Doesn't mean we don't want it.

Bernstein: `nullptr` is harmless, and we want the fix separately. However, `#define NULL nullptr` may violate POSIX.

Ballman: be very careful if changing `NULL` to `void *`, as this is the opposite of C++. Do not want to silently change the behaviour of code by NB comment.

Myers: if we discourage `0` and `0L`, it would go in Future Library Directions. We should start by making the integer definition obsolescent and allow `nullptr` silently. This is better handled by a later comment or paper.

Steenberg: fail to see how adding more possibilities fixes that problem.

Straw poll: (decision) Should WG14 accept NEN/NL5-038 by removing the nullptr changes in n3042 from C23?

Result: 3-11-8 (no consensus)

Gustedt: in C, any scalar converts to `bool`, this is not specific to `nullptr`.

Ballman: implicitness is problematic because it allows misuse.

Gustedt: no conversion is needed for `if` etc.

Ballman: I agree that it would be inconsistent but that's the benefit.

Tydeman: what is the benefit of allowing it?

Gustedt: could be the result of a macro.

Gilding: maybe it shouldn't be silent; analogous to real floating conversion.

Steenberg: `NULL` being false is essential, and shouldn't change.

Bernstein: this would break code using `nullptr` for `NULL`.

Straw poll: (decision) Should WG14 accept US23-062?

Result: 3-11-7 (no consensus)

Wednesday

FR-036 (`_BitInt` as a macro)

Ballman: this lacks implementation experience, motivation, or evidence that the specifier would combine correctly with qualifiers.

Gilding: also you can do it anyway, if really needed.

Myers: this is the *easiest* part of implementing `_BitInt`; a pointless change.

Ballman: this was not hard in Clang and can't really imagine the problem.

Gustedt: it would allow for predefined simple identifiers that bake-in the size argument.

Ballman: will that combine with `unsigned`?

Straw poll: (decision) Should WG14 accept FR-036?

Result: 2-4-12 (no consensus)

Bhakta: why so many abstentions?

Gilding: total neutrality. Would implement it fully, but wouldn't stop someone else doing less.

US11-037 (`thread_local`)

Gustedt: we thoroughly discussed this and it provides no new information.

Uecker: so in C++ sometimes constructors run at init-time, but this doesn't affect C at all.

Ballman: I agree with Jens that this is not new, but we wanted a chance to resolve it. New ideas suggested in the study group need time to propagate.

Gilding: I recall that there was disagreement over whether there even *was* any incompatibility.

Bhakta: there is an issue but it's C++'s problem. Maybe that's unfair.

Uecker: not so much the problem but that we didn't realize we created one. C++ types have a different ABI from the C types. But as long as `extern "C"` exists there is a way out, so not really understanding it.

Ballman: I don't have other information, but this will bite users from shared headers. If it's C++'s problem, OK, but it exacerbates an issue.

Steenberg: adding a second way of doing something which we recommend against, that will stick.

Gustedt: this isn't made any worse by changing the macro to a keyword.

Ballman: agreed this is not a new problem, it affects only new code.

Straw poll: (decision) Should WG14 accept US11-037?

Result: 7-10-4 (no consensus)

FR-044 (clarify optional identifiers)

Gustedt: this isn't my idea, it is a compromise position. Nobody will be hurt and no real implementations are affected.

Ballman: does this include all Annexes?

Gustedt: yes. Still only optional.

Bhakta: understand it to mean if an independent library implementation includes identifiers and the compiler doesn't, the user may get a link error but not a compile error. Like a feature test macro could be usable by the user but missing.

Gustedt: no, it's still potentially-reserved.

Ballman: so that's on the user because the implementation did reserve a potentially-reserved identifier.

Bhakta: so an impact may exist.

Straw poll: (decision) Should WG14 accept the resolution to FR-044 in n3072 section 6.1?

Result: 7-2-9

GB-056 (let conditionals use compatible `struct` types)

Myers: if we accept this now it can stil be changed by the second paper to talk about composite type. We can set a direction now and improve the current state.

Straw poll: (decision) Should WG14 accept GB-056?

Result: 14-0-5

CA3-064 (trigraphs again)

Gilding: no proposed wording?

Bhakta: it would just undo the change.

Straw poll: (decision) Should WG14 accept CA3-064?

Result: 3-11-6 (no consensus)

Seacord: ...and fill in the grave with concrete.

CA4-114 (non-prototype declarations)

Uecker: the GCC forward-declared parameters extension could help here, we could bring it back.

Gilding: both seem to be about name declarations, not types?

Bhakta: this one is definitely about forward declaration.

Uecker: the paper was rejected and not revised, is there any interest?

Straw poll: (opinion) Would WG14 like something along the lines of N2780 to return for CD2?

Result: 9-4-7

Bachmann, Lukasiewicz: changed mind since then

Straw poll: (decision) Should WG14 accept CA4-114?

Result: 4-13-3 (no consensus)

GB-065 (rvalue qualification), US31-118

Gustedt: is this a preference vote? I prefer the second option.

Myers: the alternatives only affect function calls. Casts is an unconditional change.

Keaton: combine with US31-118.

Myers: `sig_atomic_t` can be `volatile`-qualified. Typedefs are the only risk here.

Seacord: function returns are an rvalue, so cv-qualification is pointless, it only matters what you assign to. Eliminating qualified types is harmless.

Gilding: definitely don’t want qualifiers in types to be deprecated! Especially `const`.

Gustedt: “explicitly” qualified.

Meneide: so, Rust-like immutability-by-default?

Seacord: I am not convinced hiding information inside `typedef` is good style.

Bernstein: maybe consider it for the next ballot.

Gustedt: in favour but want better wording. Maybe bring it for CD2?

Straw poll: (decision) Should WG14 accept GB-065, with the alternative wording for the first part?

Result: 19-0-1

Straw poll: (decision) Should WG14 accept US31-118?

Result: 5-5-9 (no consensus)

Seacord: the discussion indicates the wording wasn't strong enough.

Bachmann: we should discourage something useless.

Bhakta: this only makes it obsolescent and doesn't break code. What wording change is wanted?

Gustedt: first need to consider the `typedef` case separately from explicit. It breaks code. Wording changes are not easy. Another option is a Recommended Practice.

Uecker: I would suggest that – it is easier to just recommend the implementation warns. Macros are an issue as well.

Ballman: would also want to assess the C++ compatibility impact. I think it strips the qualifier as well, but it's definitely allowed.

Voutilainen: it is ignored but always allowed, mostly because references are in the same syntactic position; it is *not* discoverable by `type_traits`.

Seacord: `volatile` deprecation was adopted in C++, gets rid of `volatile` on return types. Was brought to C in N2743, so we almost have language that would align with C++.

Voutilainen: it was suggested and given partial direction – we are trying to get rid of `volatile` on types or variables and replace with a special access form.

Gustedt: some parts were accepted but not all.

Seacord: the part that was *un*accepted was `+=` which is used in the wild. We did not argue for all qualifiers, only `volatile`.

Straw poll: (decision) Should WG14 accept US31-118?

Result: 7-7-6 (no consensus)

US19-066 (compound literal scope)

Gilding: is there wording?

Gustedt: N3080 has it. Second part is more important.

Myers: there is an issue with ignoring storage class because the parser doesn't know if we're in a definition yet, and `constexpr` storage class changes the type, maybe changing other parameters. Need a new paper to address that.

Ballman: last sentence of paragraph 5 – in a function this would have no linkage and is therefore not covered by the wording which uses linkage.

Gustedt: disagree that that applies. Agree with Joseph about needing more wording.

Bhakta: “execution”? Elsewhere we use “startup”.

Gustedt: Agreed. There are three possible times for the initialization: at translation; prior to startup; during runtime. The text is trying to specify which.

Tydeman: nothing about `thread_local` being initialized at a distinct time?

Gustedt: the initializer is evaluated once and copied on thread startup only. The evaluation itself must be a constant expression.

Ballman: mostly filed this because Clang is non-conforming; this resolves the intent of the comments.

Straw poll: (opinion) Does WG14 want something along the lines of N3080?

Result: 15-0-2

Keaton: this can be homework, we can defer these comments (066 and 067).

US26-075 (`constexpr` evaluation time)

Seacord: no wording but this needs a fix.

Gustedt: partially solved by N3078 and the rest by the CFP report. There is a problem of understanding; `constexpr` objects *are not* re-evaluated at runtime, even if they have automatic duration. Trying to capture that in the resolution. This also captures `thread_local` with the same rules.

Ballman: not sure this addresses 075 – how do we match the semantics to runtime, when there is no runtime and no environment? WG21 added a Recommended Practice that evaluations are consistent.

Seacord: share Aaron’s concern about a different direction from C++, does this no longer allow implementations to defer evaluation to runtime?

Gustedt: they can, but with the environment constraint. This is not unique to `constexpr`.

Seacord: have we defined the translation-time environment?

Tydeman: yes, in Annex F, which is conditionally normative.

Myers: N3082 also has wording for this; no conflict, should apply both.

Steenberg: the text says “if … is” – it gives no room to defer to execution. If the intent is to allow evaluation at runtime, it should say so. Would not want to use this if it produced different results.

Gustedt: always the “as-if” rule. This is exactly the same problem as faced by `static`, except for the storage duration.

Bhakta: this is not a new problem and doesn't change the expected behaviour, only a different declaration. The words here and those provided by CFP both solve the problem, either is good. The original problem was legitimate, and the C23 draft is unimplementable, but both wordings fix it.

Tydeman: the translation-time environment may use extra precision as specified by Annex F.

Gustedt: the intent is to do exactly what is done for `static`. No new behaviour.

Ballman: can we see the CFP wording before the vote?

Straw poll: (decision) Should WG14 resolve US26-075 and GB-279 by adopting the wording in N3078 section 6.4, first set of changes, and N3082 sections US26-075 and GB-279?

Result: 17-0-3

Tydeman: issue with the word "must".

Gustedt: right, this is the wrong term.

ACTION: Gustedt to make a comment for CD2 to point at this.

GB-076 (declaration contexts)

Myers: noticed in the context of underspecified declarations that it is unclear what is part of the declaration. Related to existing places, do the contents of `typeof` etc. count? No wording because we need to establish a principle and will bring wording for CD2.

Gilding: case 1 e1 is implemented in practice; `typeof` is clearly supposed to work as a drop-in replacement for any type name and would be very surprising if it could not be used in some contexts.

Ballman: `auto` with a compound is useful for the Memento Pattern, given adequate generic function support.

Steenberg: skeptical of anonymous-anything, it is just hard to debug.

Ballman: Clang has supported this since forever.

Keaton: is this in C++? (yes)

Steenberg: not all existing practice is good.

Myers: interesting but explicitly invalid since C90, not suggesting we change that here.

Keaton: we decided not to import this from C++ at that time.

Straw poll: (opinion) Does WG14 want future wording to err towards disallowing the questionable cases illustrated in N3070?

Result: 13-1-5

(comment itself rejected pending wording)

GB-078 (unclear example of invalid string)

Myers: reflector indicated confusion.

Seacord: shouldn't it be the other way around? `0xFF` is OK in the `unsigned char` array, but maybe not in a `char` array?

Gustedt: agree, this seems wrong way around.

Gilding: it's a typo.

Gustedt: I will bring it as a homework change.

Myers: we will need normative wording if the homework paper concludes something different.

Bhakta: unclear on the problem with the resolution?

Keaton: `0xFF` is a good unsigned constant and a bad signed constant. The example is backwards.

Bhakta: so the example is talking about `string`…?

Gustedt: about the values of the characters within the strings.

Myers: the resolution must not depend on the way the character was written, only on the value within the buffer.

Steenberg: the example should say `signed` to clarify where the problem lies, rather than assuming a representation.

Meneide: I am wondering if the example *was* wrong – "possible violation" is next to the unsigned character string, which would get `-1` if the representation of `char` is signed. Why is one OK and not the other?

Thursday

DE-069 (when `sizeof` array is evaluated)

Uecker: we are missing wording for evaluation in type names inside cast and `typeof` expressions. The wording should not imply evaluation in unevaluated branches.

Myers: the wording is correct but confusing – evaluation of `typeof` and of the operand aren't the same, without examples you need to look at the specification of `typeof`.

Uecker: it could be improved more, want to fix at least this part.

Gustedt: it's a definite improvement even without examples.

Ballman: I am confused by the edits to 6.8 removing the VLA-declaration text.

Uecker: "any size expressions" is intended to cover this case. Examples and tweaks can either be editorial or brought for CD2.

Straw poll: (decision) Should WG14 accept DE-069?

Result: 13-0-4

GB-080 (`constexpr` null pointers)

Gilding: we missed this one, but there's no ambiguity in meaning – it can only ever be null even if not a null-pointer-constant.

Gustedt: “null pointer constant” in the Standard is a confusing and complex term. It’s not just “constants that are null”.

Myers: fine with this approach; note that GB-079 also suggests adopting the first part of this text.

Straw poll: (decision) Should WG14 resolve GB-080 using the text from N3078 section 2.1 and the first sentence of the GB-080 change?

Result: 15-0-3

Uecker: is it clear that something with the ”value null” is a null pointer? Zero isn’t a pointer.

Bhakta: it’s easier if we write where the change applies, not in the text.

Myers: now GB-079 only needs an example.

Straw poll: (decision) Should WG14 resolve GB-079 by adding the suggested example?

Result: 17-0-1

GB-081 (“no change of value”)

Ballman: we’re not mandating signaling-NaN in `constexpr`?

Bhakta: matching the behaviour of `static` should get the signal at execution-time.

Ballman: so it falls out naturally from the bit pattern.

Seacord: didn’t we change this yesterday? Is there a conflict?

Bhakta: this follows that change exactly and is consistent.

Myers: need clarity on whether the editorial comment is needed.

Bhakta: certainly it’s wanted, but it’s only clarifying.

Straw poll: (decision) Should WG14 resolve GB-081 by adopting the wording from the N3082 section named "about N3081", including the editorial comment?

Result: 15-0-3

CA1-084 (attribute syntax errors)

Ballman: this is a compatibility topic and I would like to reject it. Implementers are refusing to do it. Say `deprecated` isn’t supported – we want to be able to ignore an unsupported attribute entirely, not examine it for errors even though it’s unhandled.

Bhakta: this was always intended to be an error, and enforcing is useful to portability.

Ballman: intent is hard to derive because this is ten years old in C++. C++11 seemed to imply that all attributes are completely ignorable. Practically, we are not going to do it. Portability isn’t that bad, we have a feature test. Issues are inherently there with a semantic implementation.

Steenberg: the whole feature rests on ignorability. Being able to add and change is much more useful and gives experimental latitude. Essential to not change what the code does.

Bhakta: nothing in this aims to make attributes required and you are still allowed to ignore it.

Steenberg: you have to understand it to be able to parse it.

Gilding: it's an asymmetric concern that messaging attributes being ignored themselves need messages. Compiling is not proof of conformance.

Gustedt: this is just asking for noise. An implementation can just spit out a warning on every ignored attribute.

Bhakta: diagnosis of errors is good.

Ballman: conformance only requires the message to say the attribute was ignored. We shouldn't limit implementations that cannot deal with this. For parsing, we explicitly wanted these to be token soup. This also requires you to do appertainment, but we don't want ignored content to be in the AST at all.

Seacord: I have never encountered an implementation saying "can't do that" before.

Straw poll: (decision) Should WG14 accept CA1-084 with the revised resolution in N3073?

Result: 2-14-2 (no consensus)

Bhakta: phew! IBM didn't want this either!

US36-087 (attribute support)

Ballman: clarifying that `__has_c_attribute` returns `0` if the attribute is ignored.

Bhakta: what do we do if the behaviour changes? Such as adding the string to `nodiscard`.

Ballman: the returned value is a date, an earlier date indicates an earlier state of support.

Straw poll: (decision) Should WG14 accept US36-087?

Result: 17-0-2

GB-089 (fallthrough items)

Myers: just use the wording from the C++ issue?

Gilding: need to avoid C++-isms we don't have wording for. Is it OK to refer to "error"?

Ballman: I'll produce a WG14 document as a homework item.

Bhakta: "substatement" is also not a C term.

Gustedt: "block item" fits.

(deferred)

GB-098 (`enum : bool`)

Gustedt: is this a normative change? Can we do this at the moment?

Myers: it is allowed but it is a bad idea.

Gilding: what is the motivating example?

Straw poll: (decision) Should WG14 accept GB-098?

Result: 14-1-3

Meneide: so I can't use `bool` as the underlying type when I only have two values?

Myers: that produces a type whose behaviour is unexpected. These comments appear to be out of the submitted order.

GB-103 (unqualified underlying type)

Bhakta: was this change intended?

Meneide: this was accidental, wasn't meant to impact it.

Straw poll: (decision) Should WG14 accept GB-103?

Result: 15-0-2

GB-164 (`errno` and exception requirements)

Myers: CFP disagree with this, but it's controversial. We don't believe the change is correct. Fine with the version in N3082.

Straw poll: (decision) Should WG14 resolve GB-164 by applying the change in N3082 section GB-164?

Result: 11-0-7

CA-N3073-006 (all numbers representable)

(this is a different comment from CA-006)

Bhakta: CFP recommends no change; difficult to accommodate because not consistent with the model and needs even more macros; model is for normalized numbers only.

Tydeman: there is no standard definition for `double double` and implementations differ in the wild.

Bhakta: I don't know of other models, but they may exist. The lack of a standard is an issue, but it brings them closer to standardization.

Straw poll: (decision) Should WG14 accept CA-N3073-006?

Result: 1-5-8 (no consensus)

US 42-169 (`nexttoward`)

Myers: this cites the wrong section!

(deferred)

ACTION: CFP to examine US42-169.

GB-286 (`wcstofN`)

Straw poll: (opinion) Should the `wcstofN` and `wcstodN` functions be fully defined and included in C23 Annex H?

Result: 9-0-8

Bhakta: this is a huge change for a CD ballot comment.

Gustedt: this is strange wording, seems rather like invention.

Bhakta: we did reserve the prefix and did something similar for `cr` (correctly rounded)

Meneide: would prefer to add the specification.

ACTION: CFP to provide specification for `wcstofN` and `wcstodN` in Annex H.

GB-287 (`strtodN`)

Myers: if we can ask CFP about functions we can also ask for this.

Bhakta: we didn't think it was practical, and we didn't want to prohibit it either, hence putting it in Annex H. But the main body and Annex H should be consistent.

Myers: why is it impractical? It has similar complexity to converting decimal to binary, and is a smaller change.

Bhakta: it is predicated on the idea that adding functions was too much for a CD.

Bernstein: adding functions isn't huge, we have analogous functions, this adds functionality.

Straw poll: (decision) Should WG14 resolve GB-287 using the wording in N3082 section GB-287?

Result: 6-4-8 (not consensus)

Tydeman: this is easy to add, need CFP to write the text.

Steenberg: is this describing bits, or an integer?

Bhakta: hexadecimal bits, with a decimal exponent.

Banham: what is the application of this? To force very specific numbers with no conversion?

Bhakta: yes, specifying exact bits is very useful.

Banham: that implies a deep understanding of the underlying representation.

Bhakta: hence its positioning in Annex H, which defines the representation.

Myers: in Annex H some conversions go from non-arithmetic formats.

Keaton: this is clearly more complex than we expected, let's poll again.

Straw poll: (decision, repeated) Should WG14 resolve GB-287 using the wording in N3082 section GB-287?

Result: 5-3-9 (not consensus)

Banham: I don't feel I grasp the nuances of this.

Gilding: I just don't want to demand more workload for CFP.

Banham: so does this force bits to be set exactly? Forcing a meaning?

Bhakta: no, you either support it or you don't. The format is well-defined.

Banham: in that case is a macro proposed to mark the optional feature?

Bhakta: we could, but don't want to. It's in the source code, being compiled; so these functions take an exact bit input and spit back a human-readable representation.

Gustedt: I don't like adding implementation-defined behaviour without a macro to guard it. If you have this feature though, why would you do that?

Bhakta: right now it's contradictory – prohibited in the main text, mandatory in Annex H. This is the most minimal fix to the Standard.

Bernstein: there are use cases outside tests – time series code cares about the "fiddly bits" of floats. We can tighten the spec over time.

Gustedt: this rounds, though.

Ballman: what's the return value when it's unsupported, if we make it implementation-defined?

Bhakta: if no conversion is specified, the result is zero.

Myers: Like the `0x` prefix produces 0.

Steenberg: if it's required in Annex H, can't we depend on Annex H? Non-floating-point implementations don't need to care.

Bhakta: that's an option. Implementation-defined is easier to specify.

Keaton: with at least two misunderstandings of what we were voting for here, let's defer this.

Friday

Gustedt: so this is only intended for *constant* strings – dynamic input is much more complex.

Steenberg: this isn't breaking anyone, let's do the right thing from the start, which means making it required if Annex H is available.

Bhakta: I can't imagine why an implementation would *not* provide it if it provided Annex H.

Straw poll: (opinion) Does WG14 want to require support for hexadecimal floating constants as arguments to strtodN outside Annex H?

Result: 15-0-2

`fopen` ("x" and "a") [N3059] (many comments)

Douglas: Microsoft have agreed to match this, because it's a good idea.

Gustedt: there is value in talking about "processes", but it isn't a term of art – we had an alternative word on the reflector?

Keaton: multiple comments are resolved by this.

Bachmann: single bytes are not normally useful, but might be needed.

Douglas: atomic append for various buffer sizes – 2GiB atomic append on POSIX, can be made arbitrary size and forced through to syscall though. `PIPE_BUF` provides a minimum guarantee for atomicity. Systems need to be tolerant of this because of logfiles.

Seacord: it wasn't me.

Bachmann: if it works, can we ask for a guarantee?

Keaton: this is an incremental improvement. Further guarantees require a new paper.

Douglas: writes don’t interleave, but only the increment is itself atomic, this is more portable. There is very wide support since the 1990s. Early FAT drives might appear to interleave writes, but were still consistent. NFSv3 didn’t do this, but NFSv4 fixes it.

Straw poll: (decision) Should WG14 resolve US60-212, US61-214, IE-211, IE-213, IE-326, and US87-327 by adopting N3059 with the chnge of "processes" to "other concurrent program executions"?

Result: 18-0-1

Douglas: Thanks! The second half of the improvement will come after C23.

US 19-066 and US20-067 (compound literals) [N3090]

Gustedt: this ensures the type matches but no object is ever evaluated.

Uecker: in the past only identifiers had scope, now we’re being more generic and maybe misusing the term? Things at “file scope” are normally `extern` declarations and the like. Should we fix this?

Gustedt: we could, but file scope is consistent and this *is* about lexical scope.

Straw poll: (decision) Should WG14 resolve US19-066 and US20-067 by adopting the wording in N3090?

Result: 16-0-4

US 27-110 (`typeof` in `main`)

Myers: this comment is explicitly wrong.

Ballman: I withdraw this comment.

Straw poll: (decision) Should WG14 accept US27-110?

Result: 1-16-1 (rejected)

US 30-115 (scope of declaration specifiers)

Myers: there is no wording provided.

Straw poll: (opinion) Would WG14 like wording to resolve US30-115?

Result: 9-0-7

ACTION: Ballman to provide wording for the US30-115 change.

DE-117 (`sizeof` in an unevaluated subexpression)

Uecker: found this by chance – evaluation of `sizeof` expressions isn’t something you would normally write, but we should define a useful behaviour. The unevaluated side becoming unspecified makes the other side take precedence according to the composition rules.

Ballman: thanks, this is currently confusing. Is there experience on the new parts?

Uecker: have patched GCC, but at the moment compilers miscompile or crash anyway so there is no existing behaviour to alter.

Myers: the proposal makes a significant change to the evaluation of variably-modified types, requiring complicated work, which may have other consequences. Would like to see this as an issue and paper for the next version, there is wider impact to consider. Please resubmit the limited `?:` case for CD2, and let's leave the rest undefined for now.

Bhakta: this is just too complicated for the short term mostly, especially with implementation divergence.

Seacord: I want to reject this and bring it back for CD2 as there is clearly not consensus now.

Straw poll: (opinion) Would WG14 like to see something along the lines of DE-117 in a future paper?

Result: 15-0-4

Straw poll: (decision) Should WG14 accept DE-117?

Result: 0-12-6 (no consensus)

US 32-121, US 34-122, US 33-123 (`auto` declarators)

Gilding: although we provided N3076 which dramatically reworks the feature to be more like C++, the goal is not to bait-and-switch and we prefer to standardize the feature as currently provided in GNU C. The paper should give direction so that implementers have an idea of what the expected practice is for the implementation defined areas.

Ballman: N3076 gives adequate direction for Clang and confirms that we will be conforming.

Myers: if we make `auto` a type specifier, I don't want declarations invalid in C++ to be allowed. Otherwise, behave according to the as-if rule.

Ballman: I just want a document I can point my users at when they ask questions.

Straw poll: (opinion) Would WG14 like to see something along the lines of N3079 in a future version of C?

Result: 10-0-7

Gustedt: so I propose we reject US 32-121 for now.

Ballman: I have what I need from this.

Straw poll: (decision) Should WG14 accept US32-121?

Result: 1-11-5 (no consensus)

Ballman: so the issue is that `auto * p` is undefined, and I don't like that. I want it nailed down.

Gustedt: for 123, this picks up the braces unintentionally and shouldn't do so.

Bernstein: does this interact with the type specifier?

Gustedt: no, it leaves that open for the future, especially if we remove the braces.

Gilding: braces are not in the GNU C feature, so this avoid confusion.

Ballman: this is helpful.

Straw poll: (decision) Should WG14 resolve US34-122 and US33-123 by adopting N3079 with editorial changes?

Result: 15-3-1

Ballman: what about the C++ entry in the bibliography? No year future-proofs against latest C++ versions.

Meneide: normally the year is left off anyway, indicating to use the latest available version.

Steenberg: the reflector argument is that SE-001 isn't a technical comment and we have now effectively voted down the second part.

GB-125 (`for` loop storage class)

Gustedt: nobody seemed to be sure why this is the way that it is anyway – it was expected to simplify the rules, but it adds a special case. I ran into it with generated code.

Bachmann: allowing it in `for` is just a static initialization. It is not reset on re-entry of the loop; there is no use for this.

Gustedt: there is a use case – this defines a loop with a finite number of executions in the program.

Gilding: it's a weird use case, but simpler than any alternative way to express it. It's easier to remove special cases and leave groups like MISRA to decide whether well-defined code is a good idea to write.

Uecker: note that we are also now allowing `typedef` here.

Steenberg: having the constraint also violates the equivalence to `while`.

Myers: note that `static_assert` is already valid here.

Straw poll: (decision) Should WG14 resolve GB-125 by adopting the wording in N3078 section 4.3?

Result: 14-1-1

GB-126 (tentative `thread_local`)

Myers: not really sure what the practice is here.

Straw poll: (opinion) Should `thread_local` declarations without an initializer at file scope be tentative definitions?

Result: 3-1-11

Bhakta: this really doesn't make sense, `thread_local` is like `static`.

Straw poll: (opinion, repeated) Should `thread_local` declarations without an initializer at file scope be tentative definitions?

Result: 1-5-10

Straw poll: (decision) Should WG14 accept GB-126 by adopting alternative wording 2?

Result: 9-0-6

FR-130 (__has_embed symbolic constants)

Bhakta: we introduced magic numbers? We shouldn’t keep those if they’re new. This seems more like a feature request than a CD fix though.

Meneide: C++ wanted them to be named, so we should add this now. Also I think it’s a good idea.

Ballman: this is a sibling group request, and an oversight in the feature rather than something new.

Myers: “mandatory” should be “predefined”.

Straw poll: (decision) Should WG14 accept FR-130 by adopting the wording in N3072 section 4, substituting "predefined" for "mandatory"?

Result: 14-1-2

6. Clarification Requests

The previous queue of clarification requests has been processed.

7. Other Business

No other business

8. Recommendations and Decisions reached

8.1 Review of Decisions Reached

That there will be a second CD ballot on ISO 9899:2023. (unanimous consent)

That TS 6010 will move to CD ballot. (unanimous consent)

Should WG14 accept SE-001, first part referring to N2510 only?

3 / 16 / 5 (no consensus)

Should WG14 REJECT comments CA-006, US15-048, NEN/NL1-049, US18-050, US17-051, US16-052?

20 / 4 / 3 (rejected)

Should WG14 accept GB-007 with the changes from N3082 applied?

21 / 0 / 2 (accepted)

Should WG14 accept GB-039 with the proposed resolution?

20 / 0 / 2

Should WG14 accept US3-011, without the word "defined"?

18 / 0 / 5 (accepted)

Should WG14 accept GB-012?

3 / 7 / 12 (no consensus)

Should WG14 accept US4-013?

2 / 9 / 10 (no consensus)

Should WG14 accept GB-014?

15 / 0 / 6

Should WG14 accept US5-018?

16 / 0 / 3

Should WG14 accept US7-022?

2 / 7 / 7 (no consensus)

Should WG14 accept GB-024?

16 / 0 / 1

Should WG14 accept GB-190?

3 / 9 / 9 (no consensus)

Should WG14 accept GB-209?

3 / 11 / 4 (no consensus)

Should WG14 accept GB-232?

15 / 1 / 1

Should WG14 accept GB-238, substituting "it is implementation-defined whether these functions avoid data races with other calls to the same function" as the proposed text?

17 / 3 / 1

Should WG14 accept NL-239 using the wording proposed in N3066?

13 / 2 / 7

Should WG14 accept GB-240 using wording option 1?

10 / 0 / 10

Should WG14 accept GB-245 using wording option 1?

12 / 2 / 7

Should WG14 accept GB-028, by adding quotes around "plain" where they are missing before "`char`"?

15 / 3 / 4

Should WG14 accept GB-029, by deleting the first sentence of NOTE 1 in 6.2.6.1?

9 / 5 / 9

Should WG14 accept DE-031?

0 / 17 / 3 (no consensus)

Should WG14 accept GB-032?

19 / 1 / 1

Should WG14 accept US9-034?

0 / 15 / 5 (no consensus)

Does WG14 want to resolve US10-035 and US24-061 using the text in N3077 sections 2.1 and 2.2, and by removing footnote 126 in CD ballot document N5777?

17 / 0 / 6

Should WG14 accept US22-058?

0 / 17 / 4 (no consensus)

Should WG14 accept NEN/NL5-038 by removing the `nullptr` changes in N3042 from C23?

3 / 11 / 8 (no consensus)

Should WG14 accept US23-062?

3 / 11 / 7 (no consensus)

Should WG14 accept the resolution to GB-071 and FR-073 in N3077 section 7.3?

16 / 0 / 2

Should WG14 accept FR-036?

2 / 4 / 12 (no consensus)

Should WG14 accept US11-037?

7 / 10 / 4 (no consensus)

Should WG14 accept GB-043?

13 / 0 / 7

Should WG14 accept the resolution to FR-044 in N3072 section 6.1?

7 / 2 / 9

Should WG14 accept GB-045?

17 / 0 / 3

Should WG14 accept GB-056?

14 / 0 / 5

Should WG14 accept CA3-064?

3 / 11 / 6 (no consensus)

Should WG14 accept CA4-114?

4 / 13 / 3 (no consensus)

Should WG14 accept GB-065, with the alternative wording for the first part?

19 / 0 / 1

Should WG14 accept US31-118?

5 / 5 / 9 (no consensus)

Should WG14 accept US31-118?

7 / 7 / 6 (no consensus)

Should WG14 resolve US26-075 and GB-279 by adopting the wording in N3078 section 6.4, first set of changes, and N3082 sections US26-075 and GB-279?

17 / 0 / 3

Should WG14 accept DE-069?

13 / 0 / 4

Should WG14 resolve GB-080 using the text from N3078 section 2.1 and the first sentence of the GB-080 change?

15 / 0 / 3

Should WG14 resolve GB-079 by adding the suggested example?

17 / 0 / 1

Should WG14 resolve GB-081 by adopting the wording from the N3082 section named "about N3081", including the editorial comment?

15 / 0 / 3

Should WG14 accept GB-083?

17 / 0 / 2

Should WG14 accept CA1-084 with the revised resolution in N3073?

2 / 14 / 2 (no consensus)

Should WG14 accept GB-086, but substituting "stored values of the objects" for "values of the variables" in 7.17.3 p13?

14 / 0 / 5

Should WG14 accept US36-087?

17 / 0 / 2

Should WG14 accept GB-094?

13 / 0 / 5

Should WG14 accept GB-098?

14 / 1 / 3

Should WG14 accept GB-100?

13 / 0 / 4

Should WG14 accept GB-103?

15 / 0 / 2

Should WG14 accept GB-127?

14 / 0 / 4

Should WG14 accept GB-149?

12 / 0 / 4

Should WG14 accept GB-152?

13 / 0 / 4

Should WG14 accept US39-155?

11 / 0 / 6

Should WG14 accept GB-157?

11 / 0 / 5

Should WG14 accept GB-163?

13 / 0 / 3

Should WG14 accept US40-166?

9 / 0 / 8

Should WG14 accept GB-173?

15 / 0 / 2

Should WG14 accept US56-187?

11 / 0 / 6

Should WG14 accept US57-189?

13 / 0 / 4

Should WG14 accept GB-229?

12 / 0 / 5

Should WG14 accept GB-230?

12 / 0 / 4

Should WG14 accept GB-268?

13 / 0 / 4

Should WG14 accept GB-153?

0 / 9 / 9 (no consensus)

Should WG14 resolve GB-164 by applying the change in N3082 section GB-164?

11 / 0 / 7

Should WG14 accept US43-170?

0 / 9 / 9 (no consensus)

Should WG14 accept CA-N3073-006?

1 / 5 / 8 (no consensus)

Should WG14 accept GB-151?

10 / 0 / 5

Should WG14 resolve US71-275 using the change specified in N3082 section US71-275?

9 / 0 / 6

Should WG14 resolve GB-287 using the wording in N3082 section GB-287?

6 / 4 / 8 (not consensus)

Straw poll: (decision, repeated) Should WG14 resolve GB-287 using the wording in N3082 section GB-287?

5 / 3 / 9 (not consensus)

Should WG14 resolve GB-288 using the wording in N3082 section GB-288?

12 / 0 / 5

Should WG14 resolve US60-212, US61-214, IE-211, IE-213, IE-326, and US87-327 by adopting N3059 with the change of "processes" to "other concurrent program executions"?

18 / 0 / 1

Should WG14 resolve US19-066 and US20-067 by adopting the wording in N3090?

16 / 0 / 4

Should WG14 accept GB-105?

18 / 1 / 0

Should WG14 accept GB-106?

19 / 0 / 1

Should WG14 accept US27-110?

1 / 16 / 1 (no consensus)

Should WG14 accept DE-117?

0 / 12 / 6 (no consensus)

Should WG14 accept US32-121?

1 / 11 / 5 (no consensus)

Should WG14 resolve US34-122 and US33-123 by adopting N3079 with editorial changes?

15 / 3 / 1

Should WG14 resolve GB-125 by adopting the wording in N3078 section 4.3?

14 / 1 / 1

Should WG14 accept GB-126 by adopting alternative wording 2?

9 / 0 / 6

Should WG14 accept FR-130 by adopting the wording in N3072 section 4, substituting "predefined" for "mandatory"?

14 / 1 / 2

Should WG14 mass-accept the editorial comments as-listed?

18 / 0 / 1

8.2 List of editorial comments for mass-approval

(all entries moved by unanimous consent)

GB-002
GB-003
GB-005
GB-008
GB-009
GB-010
GB-015
GB-016
GB-017
GB-019
GB-020
US8-021
US6-023
GB-025
GB-026
GB-027
GB-030
GB-033
GB-046
GB-047
GB-053
GB-054
GB-055
GB-057
GB-059
GB-063
GB-070
GB-072
GB-074
GB-077
GB-082
GB-088
GB-090
GB-091
GB-092
GB-093
GB-095
GB-096
GB-097
GB-099
GB-101
GB-102
GB-104

GB-147
GB-156
US63-216
GB-220
GB-267
GB-269
GB-271
GB-276
US67-278
GB-108
US28-109
GB-111
GB-112
US29-113
DE-116
GB-119
GB-120
GB-124
GB-128
GB-129

8.3 Review of Action Items

KEATON to submit TS 6010 to ISO early so the editors can get started reviewing it.

GUSTEDT to create an n-document for this.

KEATON to get permission from the Unicode Consortium to cite the Unicode standard.

(resolves GB-004)

SEACORD to write a paper on the issue highlighted by GB-012 as a second-round comment.

BHAKTA to propose alternative wording to GB-232 in an n-document.

GILDING to review all uses of "negation" in the Standard.

GUSTEDT to make a CD2 comment for `thread_local` changes in N3078. (DONE)

CFP to examine US42-169.

CFP to provide specification for `wcstofN` and `wcstodN` in Annex H.

BALLMAN to provide wording for the US30-115 change.

CFP to provide wording for `strtodN` functions accepting hexadecimal constants.

GUSTEDT to provide wording for GB-078.

9. INCITS/C Business

(removed from N3116)

10. Thanks to host

Thanks to ISO for providing Zoom capabilities.

11. Adjournment

(INCITS/C and WG14 combined motion)

Seacord moves, Tydeman seconds. No objections.

Adjourned.