Draft Minutes for Sept 30 - Oct 4, 2024

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

WG14 / n3372

Dates and Times

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

30 September, 2024	09:00 – 12:00	Lunch 13:30 – 16:30
1 October, 2024	09:00 – 12:00	Lunch 13:30 – 16:30
2 October, 2024	09:00 – 12:00	Lunch 13:30 – 16:30
3 October, 2024	09:00 – 12:00	Lunch 13:30 – 16:30
4 October, 2024	09:00 – 12:00	

Meeting Location

Perforce Software Headquarters
400 First Avenue North #400
Minneapolis, MN 55401

Meeting information

Venue information: n3270
Meeting Scribe: **Svoboda**

Local contact information

Alex Celeste <aceleste@perforce.com>

1. Opening Activities

1.1 Opening Comments (Seacord)

Welcome to 70th meeting.
Seacord likes to keep everything on schedule. Leave time for straw polls - at least 5 minutes.
Wait till section of agenda to hear from national bodies.

1.2 Introduction of Participants/Roll Call

2024 Sept 30-Oct - 4 WG14 meeting

Name	Organization	Country	Notes
Aaron Bachmann	Efkon GmbH	Austria	Austria NB

Name	Organization	Country	Notes
Aaron Ballman	Intel	USA	
Alejandro Colomar	Self	Spain	
Alex Celeste	Perforce Software	USA	MISRA
Anton Zellerhof	Self	Germany	
Ash Chronister	Apple	USA	
Bill Ash	SC 22		SC22 manager
Chris Anley	NCC Group	USA	
Chris Bazley	Arm	UK	
Clive Pygott	LDRA Technology	USA	
Dan Plakosh	SEI/CERT/CMU	USA	
Dave Banham	BlackBerry QNX	UK	MISRA Liaison
David Svoboda	SEI/CERT/CMU	USA	Undef. Behav. SG
Eskil Steenberg	Quel Solaar	Sweden	Sweden NB
Fred Tydeman	Keaton Consulting	USA	INCITS/C Vice Chair
Glenn Coates	Real Time Embedded Ltd	UK	
Henry Kleynhans	Bloomberg	USA	
Jakub Łukasiewicz	Motorola Solutions Systems Polska	Poland	Poland NB
JeanHeyd Meneide	NEN	Netherlands	Neth. NB; C++ Comp. SG
Jens Gustedt	INRIA/ICube	France	France NB
Joseph Myers	Red Hat	UK	UK NB
Joshua Cranmer	Intel	USA	
Lenard Mollenkopf	Self	Germany	
Martin Uecker	Graz University of Technology	Austria	
Michael Wong	Codeplay	Canada, UK	Canada NB; WG21 Liaison
Nevin Liber	Argonne National Laboratory	USA	
Niall Douglas	Ireland ned Productions Ltd	Ireland	Ireland NB
Nick Stoughton	Self	USA	SC22 Austin Group Liaison
Nikita Popov	Red Hat	Germany	Guest
Patrizia Kaye	Self	UK	
Rajan Bhakta	IBM	USA, Canada	USA NB; INCITS/C Chair
Robert Seacord	Woven Planet North America Inc	USA	

Name	Organization	Country	Notes
Roberto Bagnara	BUGSENG	Italy	Italy NB, MISRA Liaison
Shiv Maddanimath	Bureau of Indian Standards	India	India NB
Stephen Huemann	Self	USA	
Ted Johnson	Hewlett Packard Enterprise	USA	

1.3 Procedures for this Meeting (Seacord)

1.4 Required Reading

- 1.4.1 ISO Code of Ethics and Conduct
- 1.4.2 ISO Guidance and Process
- 1.4.3 IEC Code of Conduct
- 1.4.4 JTC 1 Summary of Key Points n2613

1.5 Approval of Previous Minutes (June 2024) n3281

Ballman: Moved
Seacord: Objections to approving the minutes, modulo the fix requested in Reflector Message 25987? *None*
Action Item: **Svoboda**: Re-submit the draft minutes as specified above.

1.6 Review of Action Items and Resolutions

- Agendas
  - **Action Item**: **Douglas**: Write a similar proposal to N3247 using wording from Annex K.

    Done

  - **Action Item**: **Seacord**: Write a paper to clarify implementation-defined vs. unspecified behavior; do they require the standard to document choices?

    No progress

  - **Action Item**: **Ballman**: Add more clear wording about semantics to address **Myers**'s comments to N3259.

    Done

  - **Action Item**: Update committee website for N3273 being on the extensions to obsolete versions of C list.

    Ballman: I did that.

1.7 Approval of Agenda n3213 on Google Docs (motion)

Bhakta: In the future, I prefer fixed agendas. ISO requires preliminary agendas.
Gustedt: The agenda serves two roles: a list of the papers, (pre-meeting mailing), and the schedule of which day we do things, and that can change.
Uecker: Can we move my item on today's agenda to tomorrow?
Seacord: OK

Celeste: Moved
Seacord: Objections? *None*

1.8 Identify National Bodies Sending Experts

Person	Nation
Martin Uecker	Austria
Rajan Bhakta	Canada & US
Jens Gustedt	France
Jyoti Kushwaha	India
Niall Douglas	Ireland
Roberto Bagnara	Italy
JeanHeyd Meneide	Netherlands
Jakob Łukasiewicz	Poland
Eskil Steenberg	Sweden
Joseph Myers	UK
Philipp Krause	Germany

1.9 Note where we are in the current C23 schedule n3156

Seacord: I've got a schedule. C23 is currently under review, and should be completed in October and sent back to WG14 for any changes. Then there will be a 2-3 week review period, so it could be published late November or early December.

2. Reports on Liaison and Collaboration Activities

2.1 ISO, IEC, JTC 1, SC 22

Seacord: I attended SC22 meeting in London last week remotely.

2.2 National bodies in WG 14

None

2.3 WG 21

None

2.4 WG 23

None

2.5 MISRA C (straw poll on communique)

Celeste: There has been no significant changes since the previous meeting. MISRA C 2025 is in development, everyone is busy. I propose that we send a communique that says our official standard titles (C23 not C24, C17 not C18).
Svoboda: Was the first standard C89 or C90?
Seacord: C89 was ANSI, C90 was first ISO.
Banham: We are talking about nicknames, which are never mentioned in the standard. How external organizations create shorthand is up to them, not us. The only standardization we have on nicknames is the identifier in __STDC_VERSION__.
Seacord: For C23, WG14 took a vote to make the "C23" nickname official.
Meneide: In C23, I listed all the attribute identifiers & values. I have no problem adding __STDC_VERSION__ to Annex M's "Nickname" column (in the new working draft).
Gustedt: We can only make recommendations.
Ballman: The committee website has a project page with our own nicknames for old versions.
Bhakta: We don't want it in the standard, the website is good enough.
Straw Poll: Do we vote to send a communique from Reflector Message 26702, with a note for what C89 and C94 are to MISRA? 19-2-2 consensus

2.6 Austin Group

Stoughton: The latest version of POSIX su8 IEEE 1003.1 24 passed DIS with 0 NB comments. There was one comment from the ISO editor saying, "Clean up and start again!". We sent it back again. If they object, we have a backdoor route via IEEE standards. So POSIX is an IEEE standard for now, not an ISO standard.

2.7 Unicode Consortium

None

2.8 Other Liaison or Collaboration Activities

None

3. Study Groups

3.1 C Floating Point Study Group activity report (Bhakta)

Bhakta: We are talking about having an IEEE 754 liaison from WG14. The new IEEE 754 committee is starting up now for a new version with new features. It should be available in about 10 years. For CFP, we are trying to get TR 18661 parts 4 & 5 published. We also have proposals for C2y; some are in this week's meeting.

3.2 C Memory Object Model Study Group activity report (Sewell)

Kleynhans: We got second-round feedback from ISO; they didn't like the format. We want feedback before we try to submit again. We will submit to ISO at the end of next week.

3.3 C and C++ Compatibility Study Group activity report (Meneide, Nina)

Meneide: We have removed of volatile features from C++26. We have planned more C++ features, such as adding an int_least128_t type to their headers. We are also making atomics constexpr. There should be no impact for these on C, and no compatibility issues. There is some interest in bringing int_least128_t to C. They deprecated array comparisons and some features of volatile from C++26.
Seacord: I chatted with **Herb Sutter** about checked integer types for C++26, they have some interest.

Meneide: Yes. But our solution won't work for them and they will want their own implementation. C++ has not talked about getting BitInt at all, they want their own solution.

3.4 Undefined Behavior Study Group activity report (Svoboda)

Svoboda: It is only now that we can craft a document and present it to WG14 within half a year. We spent two years waiting for this. This week, we also have a document in progress to present: n3155 is "Examples of Undefined Behavior for Annex J.2". n3388 is the Educational Undefined Behaviuor TR. We already have lots of feedback on it. Our next UBSG meeting is Friday Nov 1.

- How to manage the Educational Undefined Behaviuor TR

  Ballman: TSs have normative wording. TRs don't, they are just white papers.
  Seacord: For Graz we would need a new work item proposal. The UBSG can submit such a proposal. How many National Body Groups are there would approve the Educational Undefined Behavior TR? *5* There are enough national bodies to publish this.

3.5 New Study Group proposal (if applicable)

None

3.6 Infrastructure Group (Meneide)

Seacord: This is timely as open-std.org seems to be down.
Meneide: The website should have been up by now, but I got sick. I hope to have it up by our next meeting.
Ballman: There is a protected section on the website; that may not belong on a git repo. But we should be able to build a git repo with the old website's contents. Two moving parts are **Plakosh**'s scripts and the email reflector.
Plakosh: I don't run any scripts, I do it by hand.
Ballman: Keld puts email reflector messages onto the website, they should go into Git instead.
Seacord: Someone has to move existing documents, and change **Plakosh**'s procedure to move from ISO website to Git. (He currently uses WinSCP). I will assign setting up a new Git repository to the Infrastructure Group.

4. Future Meetings

4.1 Future Meeting Schedule

February 24 to February 28, 2025, Graz University of Technology, Graz, Austria
Seacord: Graz is hosting WG21 two weeks before this meeting. Does anyone want to move our meeting up so they are back-to-back? *None*
Fall, 2025 Candidate: Suan Sunandha Rajabhat University
Or Pittsburgh
We are looking for hosts for 2026 meetings

4.2 Future Deadlines

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

Pre-Minneapolis 2024 – 30 August, 2024
Post-Minneapolis 2024 – 31 October, 2024

5. Document Review

Monday, 30 September

- **Douglas**, fopen_s "p" and bring fopen_s's mode closer to POSIX 202x n3275 (0.5 hours)

  Seacord: If we can provide precise instructions to the editor on what to change, we could vote on it. Do we have that? We should have Annex K wording follow the POSIX wording.
  Myers: This paper is adding things about the meaning of "x" and "p" in fopen_s(). Why do we need Annex K to explain these things at all if they are already in fopen()?
  Bhakta: We could just vote on the paper as is.
  Straw Poll: Does WG14 want to adopt n3275 for C2y? 15-1-9 consensus

- **наб**, stdarg.h wording n3285 (0.5 hours)

  Meneide: The new version of the paper is n3359
  Bhakta: In section 2.3, the "may" should be a "shall".
  Meneide: Agreed
  Seacord: I have various wordsmithing comments.
  Myers: Declaration of variadic functions should come earlier in the standard, in the Definitions section.
  Seacord: This is homework for later in the week.

Lunch Break

- **Bhakta**, Floating-point exceptions for macro replacements n3286 (0.5 hours)

  Straw Poll: Does WG14 wish to adopt n3286 into C2y? 16-0-6 consensus
  Ballman: Should this document be applied to previous versions of the standard?
  Seacord: Any objections? *None*
  Action Item: **Ballman**: Add n3286, n3322, n3305 to the list of retroactive changes.

- **Bhakta**, Nonsensical Parenthetical n3287 (0.25 hours)

  Straw Poll: Does WG14 wish to adopt n3287 into C2y? 18-0-4 consensus

- **Bhakta**, Give consistent wording for SNAN initialization n3288 (0.5 hours)

  Myers: I commented on this in Reflector Message 26317.
  Bhakta: CFP doesn't have strong opinion on the wording.
  Gustedt: **Bhakta**, do you want to take this as homework?

- **Bazley**, Standardize strnlen and wcsnlen (v3) n3326 (0.5 hours)

  Bhakta: Can we have text saying "insert/delete the following" rather than just coloring (for future documents)? I prefer the first set of changes.
  Svoboda: Why is strnlen() not in ISO C?
  Gustedt: It was proposed in n2351 and rejected.
  Seacord: I see "length" as the number of chars before the bound of an array
  Straw Poll: For n3326, do you prefer wording 1 or wording 2? 6+3-3+3-1+8 preference for wording 1
  Straw Poll: Does WG14 wish to adopt wording 1 from n3326 into C2y? 10+9-0-3 consensus

- **Bazley**, strb_t: A standard string buffer type (v3) n3306 (0.5 hours)

  Ballman: The feedback is mixed. We need a string library. But this interface doesn't feel like a good interface, because it follows streams & file I/O in design. I would prefer something like buffer management.
  Bazley: If I could propose something like std::stream, I would not have bothered. The stream aspect is not

as scary as people might think. People who object to the extra state don't object to the state of storing the end of the string. What harm does the "cursor" do?
Ballman: Users don't like paying for things they do not use, such as the current position within the string.
Meneide: I have the same misgivings. Once I rationalized this was a stream-like interface, the misgivings went away. Consider standardizing this along with a low-level buffer management system.
Gustedt: Alex's idea of sub-clause 8 would solve the problem of this having no implementation experience (except for your implementation).
Bhakta: I strongly oppose the requirements for freestanding to support this.
Celeste: **Bazley** has implemented this in a portable library. So people could use it rather than build their own distinct implementation.
Colomar: Strings in C have problems. I am wary that adding a new interface with little prior art might hide similar problems in other places. I'd like this in a 3rd-party (nonstandard) library first.
Bhakta: For more acceptance, it would be good to not require freestanding implementations to support this.
Ballman: I am uncomfortable with putting this into clause 7 because it does not exist yet. I am more comfortable putting this into a TS. That avoids the risk we had with Annex K where no platforms support it.
Łukasiewicz: I don't remember the last time I used FILE*. I'm not against this interface, but I want to explore alternatives, especially with potential features of C2y, not C23.
Bazley: Some people use FILE*s for doing safe management of string buffers.
Opinion Poll: Would people like to see something along the lines of n3306 as a TS? 3+6-0+1-5+6
Bazley: I'd like a poll about the importance of supporting the C11 character types
Bhakta: Can that wait until after **Meneide**'s paper?
Bazley: OK

- **Celeste**, C Extensions to Support Generalized Function Calls, v3.5 n3315 (0.5 hours)

  Bhakta: Regarding asking if clause 4 should be put in, I prefer not to add it in.
  Celeste: OK
  Bazley: I find the title to be opaque.
  Ballman: About the tail call: We aspire to reject a program if the tail call cannot be honored.
  Bhakta: Functions often have customize-able prologues & epilogues.

- **Celeste**, Obsolete implicitly octal literals and add delimited escape sequences n3353 (0.5 hours)

  Myers: Is this voting with editorial fixes?
  Celeste: Yes.
  Myers: Non-zero should have a dash. And there is a "can" vs. "may" error. I have comments in Reflector Message 26705.
  Gustedt: There is the issue of semantic changes in strtol().
  Straw Poll: Does WG14 wish to adopt n3353 into C2y? 8+7-1+0-1+0 consensus
  Opinion Poll: Does WG14 wish to see a future paper to support prefix octal on the printf family of functions? 6+6-0+0-4+2

- **Celeste**, if declarations, v3 n3356 (0.5 hours)

  Bhakta: I don't like the declaration as part of "if". It is hard to see.
  Celeste: That is what C++ does; we would have to make a conscious decision to diverge.
  Gustedt: We have that rule for if/else.
  Bazley: This might seem redundant, but I think people care about this nicety; they don't want to write more curly brackets & indentation.
  Kaye: I like this; I use it in my day job. It makes code a lot cleaner.
  Bhakta: When writing blocks with curly braces, you can have an identifier with the same name in the if. So the code could be cleaner, but it could also be dirtier.
  Celeste: Tools will normally warn when you shadow variables.
  Straw Poll: Does WG14 wish to adopt n3356 into C2y? 7+8-1+0-3+0 consensus

Gustedt: Are declarations enclosed in loop statements in C++?
Celeste: No
Opinion Poll: Does WG14 wish for us to propose declarations enclosed for for statements into C2y? 2+5-3+2-5+2

- **Celeste**, Named loops, v3 n3355 (0.5 hours)

  Bazley: I support this proposal. A nice feature to fill the gap between branch-wherever-you-want and break/continue which is too limited.
  Łukasiewicz: If we really need something like this, I don't want the same syntax as goto labels.
  Ballman: In OpenMP you have pragma's that precede the loop; depending on the pragma, the name might have to precede or succeed the pragma. We would need some wording clarification.
  Opinion Poll: Does WG14 wish to adopt something along the lines of n3355 into C2y? 5+6-3+2-2+2

- **Celeste**, Case range expressions, v3 n3354 (0.5 hours)

  Ballman: There are lots of use of "Optionally",
  Celeste: Yes, for whether we want something to be a constraint o recommended practice.
  Bhakta: I didn't like the range [number..number].
  Celeste: I would like to put that in a separate paper.
  Bazley: I am not convinced that it is useful to have the recommendation to emit a diagnostic message if the range only contains a single value.
  Gustedt: This will happen a lot in generated code (although not much in hand-written code).
  Myers: I also agree "no" on singleton ranges.
  Ballman: It is macros that cause this problem.
  Meneide: To make an empty range, you do [16…15]. That concerns me, because many people will swap the numbers. I'd like it to be a constraint. That's why I was requesting a half-open range.
  Seacord: Could we make this implementation-defined behavior?
  Ballman: Empty ranges are no different than unreachable code.
  Opinion Poll: Does WG14 prefer a constraint violation for backwards ranges in n3354? 3+2-5+2-1+2
  Opinion Poll: Does WG14 prefer the recommended practice for single-value ranges in n3354? 3+0-4+4-1+4

- **Celeste**, Strong typedefs n3320 (0.5 hours)

  Gustedt In your example for _newtype, there is no _newtype.
  Celeste: Yes, there are several typos.
  Seacord: I would prefer that _newtype is a strong type.
  Bazley: I agree that attributes are good at communicating intent but not good at restrictions. I fear C splitting into two languages, with vs. without attributes.
  Ballman: Has this paper gone in front of the C++ Compatibility Group? Do we want any constraints around the new type for function pointers? C++ wants something along these lines.
  Meneide: Using the "strong" attribute with a safety framework solves the problem but only for compilers that respect the attribute.
  Opinion Poll: Do we want something along the lines of the "strong" attribute from n3220? 6+5-2+1-2+2 strong direction
  Opinion Poll: Do we want something along the lines of the _newtype keyword from n3220? 5+1-1+2-4-5
  Bazley: I am not prepared to judge this proposal. I need more compelling examples of intended usage.
  Ballman: I'd like to see implementation & deployment experience with _newtype.
  Opinion Poll: Do we want the "strong" attribute from n3220 as a keyword? 2+4-3+1-5+4

- **Bhakta**, Allowing stricter alignment for atomic types n3312 (0.5 hours)

  Gustedt: What's going on in the field?
  Bhakta: These types are already lock-free and larger than the fundamental alignment.

Straw Poll: Does WG14 wish to adopt n3312 into C2y? 7+4-1+0-2+5 consensus
Myers: I gave my concerns on the reflector.

- **Celeste**, User-defined qualifiers n3321 (0.5 hours)

  Ballman: I am concerned about type qualifier explosion. We now have generics that allows you to specify a type rather than an expression. Are there any C compiler that do something like this?
  Łukasiewicz: Would these qualifiers create confusion for the users?
  Gustedt: If we change syntax other than indicated in the paper; this could make it clear if this is a user attribute.
  Kaye: I'd like to see example code that makes this clearer
  Bazley: How implementable is this? We could add a restriction on number of bits. I feel like qualifiers have semantics associated with them, and this doesn't touch on these semantics.
  Ballman: I have a concern with the design: We risk closing off design space for ourselves. Can we add a qualifier without stomping over user code?
  Opinion Poll: Would WG14 wish something along the lines of N3321? 2+0-6+6-2+2 direction not to proceed

Tuesday, 1 October

- **Uecker**, Unspecified Sizes in Definitions of Arrays n3295 (0.5 hours)

  Myers: I commented on this in Reflector Message 26614.
  Uecker: Yes, the wording needs work, I am not done yet.
  Gustedt: I like this paper.
  Bhakta: What about 2 declarations that are the same but incompatible when pointing to differently-sized arrays? This makes it harder to reason when looking at the code (even though you can do the same thing with auto). The general idea is good, but this is not the right way to do it.
  Gustedt: That is multiple definitions, so it is not valid.
  Colomar: That is similar to char arrays with an unspecified number of elements, and that is already in the language.
  Uecker: There is an example at the end.
  Meneide: The * here can be a variably-modified type or compile-time constant. If it is the latter, pointers to arrays of different sizes are incompatible types.
  Ballman: If I have an initializer that is a conditional expression, and the arrays have different sizes, is that a constraint violation?
  Gustedt: Yes
  Celeste: This should probably be limited to types that you have a static size for. There should be a constraint violation against the right-hand-side being a variably-modified type.
  Colomar: This seems to be for consistency with string initialization. Incomplete arrays would make more sense. They would also be more consistent with **Meneide**'s paper.
  Meneide: In this room, people thought the paper was making VLAs. So using * might provoke some confusion. We should improve the wording to let these be incomplete arrays.
  Gustedt: I am opposed to that. Incomplete arrays are completely different; these are arrays with known sizes at compile time. We need the * notation to distinguish between them.
  Ballman: It is awkward that you can use * in arrays only in function declarations, not definitions. But in the array you get a fixed-size type. This is hard to explain to people.
  Celeste: Composite types may not be the right mechanism to use here. For auto assignment, the type on the right is authoritative. Composite types is when you have to unify two bits of type information to form a whole.
  Meneide: To do this, we should commit to a mechanism of deducing types on incomplete info.
  Opinion Poll: Would WG14 wish something along the lines of N3295 in C2y? 6+4-2+0-2+5

- Generic block (1 hour)

- 3348 **Uecker**, Matching of Multi-Dimensional Arrays in Generic Selection Expressions (updates n3290 (0.5 hours)

  Bhakta: In your penultimate generic example, is it valid if you put a constant instead of the "m"?
  Uecker: Yes
  Celeste: n3260 doesn't mention incomplete array types.
  Meneide: I am not enthusiastic about undefined behaviuor on matching VLA types, and this encourages that.
  Uecker: This doesn't add undefined behavior, undefined behaviuor is when a size expression is evaluated at runtime, but removal of modified types. That case could use undefined behavior.
  Celeste: Variably-modified types can be compatible with fixed-size types. This is a weakness in the language the paper takes advantage of.
  Ballman: What if I have an association using an array? Inside array bounds has a sizeof with a variably-modified type? This might be nice to clarify with an example.
  Uecker: That's a general problem.
  Gustedt: I disagree that this should be made incompatible. A function could have a variably-modified type as parameter but invoked with a fixed-size type.

- **Meneide**, _Generic Realignment and Improvement, r0 n3331 (0.5 hours)

  Gustedt: We had this discussion 10 years ago. People insisted that Generic is a cheap overloading facility. We've deviated a lot from that.
  Bhakta: I like this paper better than n3290. I can reason with this paper better. One part is not clear:
  Uecker: I agree about all the analysis, but I disagree about the conclusion: Generic is useful for testing properties. It is more confusing to test other properties than type compatibility.
  Meneide: Maybe we should look at incomplete array types and more general fixes. Generic is already like type compatibility. We should take a whole more holistic approach to typing.
  Colomar: I abuse Generic tools, writing macros with stronger type checks than what a function interface does. This paper is what I was looking for. The function compatibility rules are loose and I want something stronger than that.
  Ballman: Perhaps my original paper was mistaken about Generic. There were two purposes: Type-generic macros, being able to call a function like sin() without type-specific functions. The other purpose was to enable users to do type matching. So there is not just one use case. Type compatibility does a disservice by denying transparency on the type system.
  Seacord: It sounds like you want reflection.
  Ballman: Not whole-hog reflection. I just want the ability to match on exact types. I am In favor of **Meneide**'s paper because it sends us into that direction.
  Celeste: Compatibility does not serve this purpose. Variably-modified types and fixed-size types are distinct types. There is a conversion between them, but it is not a compatibility check. The requirement for this check is implicit.
  Myers: Matching array types with Generic is obscure anyway. We need some ways of examining properties of types, it shouldn't be shoehorned into Generic syntax.
  Uecker: We have a not-well-defined notion of identical types, and we have type compatibility, and Generic uses compatibility. If we want to have users explore types, then properties are thrown away at many places. With regard to variably-modified types distinct from fixed types, there is no conversion between these types; they are compatible.
  Ballman: We should give direction so both **Meneide** & **Uecker** know what we want.
  Celeste: I'd like a poll establishing if we want a trait standard.
  Opinion Poll: Would WG14 like something along the lines of n3348 in C2y? 1+2-2+1-7+8
  Opinion Poll: Would WG14 like something along the lines of n3331 in C2y? 5+1-1+1-4+7

- **Uecker**, Slay Some Earthly Demons II n3340 (0.25 hours)

  Celeste: I am unconvinced by undefined behaviors existing just for extensions. Removing them is a good thing.
  Straw Poll: Does WG14 wish to adopt n3340 into C2y? 10+11-0+0-0+0 passes

- **Uecker**, Slay Some Earthly Demons III n3341 (0.25 hours)

  Bhakta: Would "unspecified" be better than "implementation-defined"?
  Uecker: Perhaps. I am unclear what that means.
  Ballman: "Implementation-defined" forces implementations to document what they do.
  Bhakta: People don't read documentation, with regard to "implementation-defined".
  Gustedt: We could add more detail to "implementation-defined".
  Bazley: By using "implementation-defined", the standard should provide two or more possibilities.
  Uecker: Agreed. Nevertheless, we have started to use "implementation-defined" in the broader way, without defining the choices the platform could take.
  Myers: For "implementation-defined" there are still missing semantics.
  Ballman: "Unspecified behavior" means "this document provides two or more possibilities", but we often don't.
  Svoboda: I think those possibilities are often implicit.
  Straw Poll: Does WG14 wish to adopt n3341 into C2y? 9+10-0+0-1+0 consensus
  Banham: Does this make implementation-defined behavior the new undefined behavior, since platforms often don't document their implementation-defined behavior?
  Bazley: Perhaps someone should make explicit the choices in implementation-defined behavior?
  Seacord: Sounds like a good idea. Any volunteers?

- **Uecker**, Slay Some Earthly Demons IV n3342 (0.25 hours)

  Seacord: Implementation-defined is the right choice, because we don't break existing implementations.
  Straw Poll: Does WG14 wish to adopt n3342 into C2y? 10+9-0+0-1+0 consensus

- **Uecker**, Slay Some Earthly Demons V n3343 (0.25 hours)

  Bhakta: A lot of this could be a bulleted list, both here and at the end. This list would be easier to understand. Also in 6.7.7.3, changes there need grammar fixes in the middle of the list.
  Bazley: I like the wording, but am not convinced it was necessary, except the comma.
  Svoboda: Is the editor comfortable with fixing the grammar in the normative wording here?
  Meneide: Yes
  Myers: There are some meaning changes.
  Svoboda: I'll take this as homework.

- **Uecker**, Slay Some Earthly Demons VI n3344 (0.25 hours)

  Ballman: This would become compatible with C++. Clang accepts "register void".
  Bazley: I disagree with **Gustedt**. Semantically yes, it is stupid to say a function has 1 argument of type void. But I'd rather fix this now than wait for a purer fix in 2-5 years.
  Straw Poll: Does WG14 wish to adopt alternative change 1 in n3344 into C2y? 10+7-0-2 consensus

- **Uecker**, Slay Some Earthly Demons VII n3345 (0.25 hours)

  Svoboda: Are there no changes outside Annex J.2?
  Uecker: I didn't find any.
  Seacord: J.2 is editorial, not normative.
  Straw Poll: Does WG14 wish to adopt n3345 into C2y? 10+2-0-6 consensus

- **Uecker**, Slay Some Earthly Demons VIII n3346 (0.25 hours)

  Gustedt: The wording is OK. We change the syntax with an empty initializer. Now this text describes syntax. This text should reformulate the wording.
  Ballman: Paragraph 7 has one wording problem. But for wide string literals, compatible with qualified or qual. These aren't the same with regard to qualification. Oh never mind, this was a pre-existing problem.

Bhakta: Unnamed members are being initialized. Does this make sense with regard to previous "earthly demons" papers?
Straw Poll: Does WG14 wish to adopt n3346 into C2y? 9+4-0+0-1+5 consensus

- **Uecker**, Slay Some Earthly Demons IX n3347 (0.25 hours)

  Gustedt: There is not just double negation, there is also "shall not be incomplete at the end of the TU"; that is weird. I prefer "something that shall be completed before the end of the TU."
  Bhakta: Can we accept as long as there are no objections by next meeting?
  Seacord: Sure
  Straw Poll: Does WG14 wish to adopt n3347, with the change suggested by Reflector Message 26758, into C2y? 9+7-0+0-1+1 consensus

Lunch Break

- **Meneide**, Restartable Functions for Efficient Character Conversion, r12 n3265 (0.5 hours)

  Gustedt: Is there any implementation experience?
  Meneide: I have had an implementation for a long time, I have been shipping it for 5 years. But it is not in Glibc.
  Myers: I have comments based on documentation numbers being suspect.
  Meneide: Those do expand to dates. UTF8, UTF16, UTF32 are stable, they are not going to change.
  Bazley: I am not familiar with macros to know when the date is significant. We shouldn't be following a meaningless convention.
  Meneide: ISO 10646 maintains characters. Our standard can claim to support characters from a particular version of ISO 10646.
  Bhakta: I like it being a date.
  Ballman: We could define it as being a strictly positive value, and ignore what it actually is.
  Gustedt: How do we convince implementers to build this?
  Meneide: I'll be submitting patches.
  Kaye: Could these changes be homework?
  Meneide: Yes.
  Opinion Poll: Does WG14 wish to adopt something along the lines of n3265 into C2y? 9+6-0+0-1+5

- **Gustedt**, Introduce complex literals v. 2 n3298 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3298 into C2y with the change of "real float" to "real floating" and without sub-clause 4.2? 9+7-0-1+4 consensus

- **Celeste**, Essential Effects for C n3317 (0.5 hours)

  Bhakta: I'd like to see this in a TS.
  Bazley: I don't see the compelling reason for adding this. None of my colleagues seem to want this.
  Celeste: These could be in comments or attributes. The purpose is being able to machine-check the effects of statements.
  Gustedt: I would be more sympathetic if these were attributes.
  Svoboda: This is a model. It could use ACSL. WG14 is not the right audience until you have a working implementation & some papers demonstrating its effectiveness.
  Ballman: Why do I want this?
  Celeste: To enforce safety.
  Pygott: Is this the same region as SPARK annotations?
  Celeste: I think so. (I am not that familiar with SPARK.)
  Opinion Poll: Does WG14 wish to adopt something along the lines of n3317? 5+5-1+1-4+4

- **Svoboda**, Examples of Undefined Behavior n3155 (0.5 hours)

Plakosh: Does an un-compile-able undefined behavior matter?
Svoboda: Well, it could be a constraint violation instead.
Myers: For the comment example, C++ moved to checking that comments are valid UTF-8 instead of continuing to allow arbitrary bytes.
Ballman: This is nice to have, but any code that is not accepted is a sample of undefined behaviors that should be removed. Much of Annex J begs the question "how do you even do that?", a question this answers.
Colomar: Obvious cases like Example 4 aren't helpful. Perhaps less obvious, more tricky examples resembling good code would be more useful?
Seacord: I find that two examples, one just barely compliant and one just barely not compliant is very helpful to define where the line falls. What is your goal?
Svoboda: Originally we wanted to make either a Technical Report or an Annex. Now we think there is also value in identifying the undefined behaviors that don't belong; That is, they should not be undefined behaviors.
Seacord: So there are three applications: extracting the slay-able demons; creating examples for use inline in Annex J; and folding them into the educational TR, which is least useful. I find that "pretty boring", so putting the examples into the Annex is more useful.
Gustedt: So are there any compilers that accept Example 4, or can we constrain it?
Bhakta: This isn't undefined in the main body text.
Myers: It is, because it is a "shall" outside of a Constraint.
Svoboda: It serves POSIX, and the undefined behavior is therefore needed.
Ballman: It is rejected by Clang, and admitted in GCC.
Colomar: It is useful when writing code and you are not sure. Having examples can help clarify whether something you write is undefined behavior, and why.
Bazley: I agree that this is best put directly into Annex J, although it seems like time that could also be spent on other things, like removing undefined behaviors outright. If there is any chance an undefined behavior can be removed, it is better to do that.
Ballman: We do give examples for `main`.
Svoboda: So this is not an earthly demon, it's a "good undefined behavior", and there are real uses for it.
Bhakta: This is part of the implementation's ability to implicitly define any undefined behavior.
Svoboda: So yes, we can go through and remove as many as possible. That will happen.
Seacord: Is this useful to the editor?
Meneide: Yes, while we initially were worried about the workload, we will be touching the entire section anyway to introduce stable tags.
Ballman: Note that the sale price of the Standard is determined partially by its page count.
Seacord: However, we are allowed to make electronic-only attachments to the Standard, reducing the page count of the print version. All the Annexes are good candidates for this treatment.
Bhakta: It does matter, I know many people who actually buy it.
Łukasiewicz: This is perfectly fine to remain published as an N-document.
Svoboda: Would including the entirety of Annex J in the text fall under Fair Use?
Seacord: That's irrelevant because ISO also owns the N-documents.
Bhakta: Not everyone is allowed to use N-documents officially, whereas a TR would be free, and informational, and available to those users.
Gustedt: …for now.
Ballman: The intended consumer of the document should be what determines its form, not the price or the IP owner or anything like that.
Svoboda: So we have five ideas: inline into Annex J.2; inline into the main body of the text; publish as either a TR or a TS; publish outside of ISO somehow, which is risky; or keep the document internal.
Seacord: I say "no" to inlining into the main body of the text. You should publish a TR against the C23 list of undefined behaviors, and then we should integrate it into C2Y's list.
Svoboda: We do have an internal table that tracks the changes in undefined behavior numbers.

- **Meneide**, More Modern Bit Utilities, r3 n3330 (0.5 hours)

Seacord: Is there still work to do on this paper, to address **Myers**' comments?
Meneide: I could finish this as homework to address current comments.
Ballman: Didn't we make the exact-width integer types required if a platform has them?
Colomar: I don't see that justified anywhere. Do you have any?.
Meneide: Rotate(-1). The return type for rotate-left & rotate-right are the generic value types. I can rename generic_count_type to be similar to the count_1's in stdc.
Opinion Poll: Does WG14 wish to adopt something along the lines of n3330 in C2y? 9+11-0+0-1+0

- **Colomar**, VLA is a misnomer (rebuttal to n3187) n3327 (0.5 hours)

  Svoboda: When did "size" start to mean "number of bytes" and length start to mean "number of elements"?
  Seacord: size_t came in C89 (not K&R).
  Gustedt: Length comes from strtok()'s "initial length". It is difficult to do for VLA.
  Bazley: In my experience everybody refers to array length. My company survey confirmed that.
  Ballman: There are uses of VLA that are impossible to change: -Wvla in Clang & GCC for instance.
  Colomar: Alternative: don't use "length" in other places.
  Straw Poll: Does WG14 want to adopt n3327 into C2y? 0+1-8+7-2+1 no consensus

- **Colomar**, New nelementsof() operator (v3) n3325 (0.5 hours)

  Bachmann: There is no parenthesis for expressions
  Gustedt: I am in favor of having this in some form. But I disagree that it is not possible for types in C23. With regard to name, I prefer lengthof.
  Svoboda: What does nelementsof(ptr) produce?
  Colomar: That is a constraint violation, you can only provide an array.
  Myers: I have comments are in Reflector Message 26617.
  Colomar: Yes, that was a typo.
  Myers: OK, but there were 5 comments. We do need to sort out various wording things before this can go in.
  Bazley: I'm in favor of this, the kind of global search/replace of the sizeof ratio with nelementsof. Perhaps the footnote should have been deleted or just left alone. With regard to its name; it should be called lengthof. With regard to brackets, it would be nice to support smaller compilers. But I was convinced otherwise, from compiler implementers on the reflector. Also it will annoy and disappoint programmers who know sizeof doesn't require brackets.
  Colomar: Requiring parentheses from the standard need not be implemented, that is not a big problem.
  Bazley: It should not be a quality-of-implementation issue.
  Colomar: I am strongly opposed to lengthof, the only current use in the standard is "length of a string".
  Celeste: I want this feature in. It should be an operator rather than a function or macro. This invites the notion of trait queries. With regard to parentheses, we already have an asymmetry where alignof() requires parentheses, but sizeof does not.
  Ballman: Regarding extent and rank, no one would want to use that form for things. I found 26000 hits for std::extent in C++ and 1000 hits for std::rank in C++ on sourcegraph.com. This is in the same general space as introspection and type traits. The reason why alignof doesn't have the non-parenthetical form is because it only takes a type name.
  Chronister: I was introduced to this macro with the name of count_of(). That is an alternative.
  Meneide: I have email from March 13, 2024: When are we getting count_of()?
  Gustedt: I searched for "length" in the standard; there are lots of instances of "length 1". The biggest usage of "length" is in VLAs. String length doesn't occur at all. So I believe there is strong support that "length" is the term that is used.
  Seacord: Size is bytes, and length is chars before the null termination character for strings.
  Johnson: I vote +1 on rank and extent.
  Łukasiewicz: Many other languages use "length" for number of elements.
  Opinion Poll: Does WG14 want something along the lines of n3225 in C2y? 7+9-2+1-1+0
  Opinion Poll: Does WG14 want to require nelementsof() from N3225 to have a syntax like the sizeof

operator regarding parentheses? 7+6-2+2-1+2
Colomar: I would like another poll question: Does WG14 want to eventually enable the lengthof operator to support array parameters to functions?
Celeste: There will be impact from this, it is impossible to predict.
Colomar: Yes, we will need to discuss this in order to do it right.
Ballman: Because of parameter rewriting rules, there would be a slight chance of breaking code. There could be re-declarations where the dimension changes, for example.
Gustedt: Yes, this needs clarification. For every call, the system must transfer info from call site to caller. We will need to figure out what to do when this is not possible.
Bazley: It would need to be a variably-modified type. WCS is simply a pointer. It is easy to say "yes", but I am wary about how it will be built.
Łukasiewicz: I think this is orthogonal to the current discussion. If we change the definition of how arrays are passed in functions, it could work.
Myers: n3188 is the sort of thing we would need to do this. We had support along the lines for it in Strasbourg. There is another paper up for discussion.
Johnson: I am still not clear if this will return the correct answer for higher-dimensional arrays.
Colomar: With regard to multidimensional arrays, the top-level array is the only one that matters. Others have concerns about the dot-identifiers from n3188. I intend to support the current syntax. Whenever I plan to do this, I will first write an implementation for GCC, and provide a paper. If you are interested, please write me an email. Someone mentioned having arrays keep the array type. **Uecker** tried this, it proved to be complex. It would be simpler to use just a pointer and keep extra info in the compiler internals.
Opinion Poll: Does WG14 want to eventually enable the length-of operator from n3325 to support array parameters to functions? 6+10-3+0-1+1

- **Bazley**, Survey results for naming of new nelementsof() operator n3350 (0.5 hours)

  Colomar: "len" is the length of a string, "dimension" is used in a 3-dimensional space, "range" is used for an interval, "extent" is a 2-argument C++ operator with different semantics, and regarding "nelements" and "count", I like both.
  Seacord: Microsoft has a _count_of macro, which does the same thing; it counts the number of elements of an array.
  Meneide: "nelements" had the least first choice, but the least last choice in **Bazley**'s survey.
  Bazley: "lengthof" was not hated by anyone.
  Opinion Poll: Does WG14 want a name derived from "number-of-elements" or "count" over others in N3225? 2+3-3+5-5+0
  Opinion Poll: Does WG14 want the name to be derived from "length-of" from N3225? 7+6-1+2-2
  Bazley: I feel we haven't explored the "length of" uses.
  Colomar: **Ballman** found cases in Clang where "length of" had an incompatible meaning.
  Bazley: I don't feel confident about WG14's direction.
  Celeste: Adding an ugly name is free, so we should do that unconditionally. There is no pressure to add a header. Adding the header without any kind of feedback seems premature.
  Kaye: We could introduce the header with a view to providing an upgrade path.
  Colomar: Isn't that what we did with bool?
  Ballman: We have precedent both ways. There is no header for _Generic. When we want a pretty name, we do provide a header for it.
  Meneide: Waiting for user feedback is good. When the survey is done, they didn't have to choose everything ugly. I'd like to see a survey about how they'd feel about the name if they had to keep it ugly.
  Seacord: For Memory Model stuff, Sewell surveyed C experts in the community.
  Gustedt: Did you just volunteer to do that?
  Seacord: I could do it on Twitter, or delegate Sewell to do that.
  Svoboda: We are just offering direction.
  Bazley: I feel that ugly vs. non-ugly is a false choice.

Wednesday, 2 October

- **Stoughton**, Timezones and the strftime function n3272 (0.5 hours)

  Seacord: Is POSIX 24 based on a particular standard of C?
  Stoughton: Yes, C17
  Myers: What exactly is a valid struct tm given that there might be additional members? n2648 addresses one particular issue.
  Seacord: Why not have an initialization function?
  Bhakta: Option 1 is the only one that is backwards compatible, right?
  Stoughton: These fields have been around in POSIX for the last 20 years.
  Bhakta: They are not in our implementation of POSIX.
  Seacord: Does ISO C specify which version of POSIX we are compatible with? *No*. Then this is not going to solve your problem.
  Stoughton: Right. We are asking for future direction for C2y to allow these things.
  Gustedt: I would distinguish between localtime() and gmtime(). gmtime() gives you the time without daylight savings. I expect gmtime() to work with whatever fields an implementation might add. But localtime() gives you local time info, including DST.
  Stoughton: They are distinguished in here. gmtime() always uses UTC as the timezone, but localtime() uses your local timezone if available.
  Bazley: There is a helpful paragraph enumerating options. Option 2 cause misbehavior in applications that expect timestamps to succeed. Is this acceptable?
  Stoughton: It's a possibility. Option 2 is what POSIX chose.
  Seacord: Option 2 makes sense for POSIX but option 3 makes sense for C.
  Ballman: My concern is: these both can break ABIs.
  Gustedt: With regard to option 1, you are adding undefined behavior. Why not implementation-defined behavior?
  Stoughton: If you fill in a struct tm by hand and then pass it to strftime(), there are uninitialized pointers in there.
  Gustedt: But you could zero-initialize these pointers and eliminate undefined behavior.
  Stoughton: You're considering option 1 with "undefined behavior" replaced with "implementation-defined behavior". I could live with that.
  Steenberg: This isn't bad functionality, but breaking ABI is bad. I would be against this extending current functionality. We could name it something else and upgrade it. Giving an impossible date should produce some standardize kind of error.
  Myers: If we are adding something new, C99 developers first added a struct and later removed it.
  Opinion Poll: Does WG14 want something along the lines of Option 1 from n3272? 5+3-0+0-4+10
  Bazley: I abstained because I think these fields look useful.
  Svoboda: We could adopt option 1 today and another one later.
  Opinion Poll: Does WG14 want something along the lines of Option 2 from n3272? 0+2-6+4-4+6
  Opinion Poll: Does WG14 want something along the lines of Option 3 from n3272? 0+1-6+1-4+9
  Steenberg: If you want this functionality, I would ask for a poll of getting it under a different name.
  Seacord: Using a different name does not solve the existing problem. Also using a different name suggests a different paper. If we change "undefined behavior" to "implementation-defined behavior", would this solve the problem?
  Bhakta: glibc tried this and failed.
  Ballman: Documenting it as undefined behavior or implementation-defined behavior, there are implementations that don't control their standard library. Clang linking against an older library might get version skew. Is that undefined behavior?
  Gustedt: That is implementation-defined behavior.
  Straw Poll: Does WG14 want to adopt Option 1 from n3272 into C2y? 3+2-2+2-5+8 not consensus
  Straw Poll: Does WG14 want to adopt Option 1 from n3272 into C2y with "undefined behavior" replaced with "implementation-defined behavior"? 4+7-1+2-5+4 consensus

- **Celeste**, Statically Dependent Array Types n3318 (0.5 hours)

Myers: We should only do this for cases like in N3118.
Gustedt: Do you have any implementation experience?
Celeste: No, I am not implementing anything until I know how complicated people want it to be.
Bhakta: This is too "fragile". The best way forward is to put this into a TS to gain implementation experience.
Bazley: This intersects with my optional type qualifier proposal.
Celeste: The rules would need to be made simpler. I did not want this to enable some implementations while disabling others.
Myers: Are there any good reasons the current rules are so complicated?
Gustedt: I prefer to do step-by-step work.
Opinion Poll: Does WG14 want something along the lines of N3318? 6+5-1+0-3+7

- **Ballman**, Allow zero-length operations on null pointers n3322 (0.5 hours)

Bhakta: Did you talk to Krause? He had concerns.
Ballman: He was worried about address spaces, so I think I covered his concerns. I did not realize it was expensive for him to do lookups because of the address spaces.
Colomar: I am concerned about the optional keyword **Bazley** suggests. If we start accepting null for 0-length memcpy() or strlen(), you have to remove the non-null attribute from these functions. This would result in less diagnostics than runtime undefined behavior.
Ballman: It should result in more defined behavior, not undefined behavior.
Colomar: Suppose I pass 74 bytes starting at null to memcpy? Today passing null is detectable, but how would it be detectable if this passes?
Ballman: You need data flow analysis to do that, the compiler can't detect it.
Colomar: The compiler could determine null-ability with the optional keyword.
Bazley: I strongly object for reasons **Colomar** stated. It complicates null pointer correctness. This proposal doesn't give me anything that I want.
Ballman: The purpose of the paper is to ensure that LLVM is conforming. LLVM already requires this (e.g. memcpy(NULL, 0, …). Null pointer analysis typically requires static analysis.
Steenberg: Are there implementations where subtracting 0 from NULL does something?
Gustedt: I am more comfortable with Clause 7, but not convinced by this paper.
Svoboda: Isn't this already well-defined behavior?
Ballman: It's undefined for the Clause 6 rules for addition & subtraction.
Colomar: malloc(0) can return a 0-array pointer, which is a valid pointer. If there are differences, then malloc(0) is broken by design. Also, why is LLVM non-conforming by relying on this? It is OK for LLVM to define this behavior.
Ballman: We allow users to use any CRT they want. We would like our users to be guaranteed that their code is portable.
Kaye: If this is common behavior, we would be standardizing existing practice. So this is a very intuitive change for me.
Meneide: This is extremely helpful, and un-breaks a few problems in the C ecosystem. 5-6 years ago, a compiler was treating memcpy(NULL, 0, …) as undefined behavior and unreachable code.
Bazley: I don't believe that Clang can be made conformant. It is tempting to say, "just going to check for null pointer" on every function. But that is not a direction I'd like to see C go in. This is antithetical to C in some way.
Ballman: Accepting this proposal means people don't have to write code. Most uses of memcpy() don't involve null. Checking for null pessimizes our code. Not simplifying the language does a disservice to our users.
Steenberg: I'm writing a list of dependable undefined behaviors, and this is one. You can take advantage of this because too much code will break. Every compiler will happily eat this code. It is bad that there are secret handshakes that distinguish between what the standard promises and what compilers accept. The smaller this difference is, the better.
Bhakta: I'm strongly for this paper. But I am concerned about pointer subtraction, echoing Krause's concerns.
Celeste: There is some interaction between this and the optional qualifier. I don't think this conflicts with

optional, because an un-optional pointer can be null.
Cranmer: I am in favor of this change. The common pattern is to have a pointer with a size, to represent a size of memory. So (NULL, 0) is a valid slice.
Douglas: Did Wg21 discuss this for C++ recently?
Ballman: I haven't attended WG21 recently.
Meneide: This has been the case for standard C++ for a long time now.
Douglas: So C++ is prior art.
Straw Poll: Does WG14 want to adopt n3322 into C2y? 9+9-1+2-0+1 consensus
Steenberg: Should we consider this for back-porting to earlier standards?
Ballman: I'd be happy to add this to that list, if people agree.
Straw Poll: Does WG14 want to add n3322 onto the list of retroactive papers on our website? 8+8-2+1-0+2 consensus

- **Ballman**, How do you add one to something? n3323 (0.5 hours)

  Bhakta: I am happy with this from the CFP side.
  Straw Poll: Does WG14 want to adopt n3323 into C2y? 10+8-0-1 consensus

- **Thomas**, Proposal for C2Y - Decimal floating-point number—misuse of term n3291 (0.5 hours)

  Seacord: A lot of things say non-empty sequence of decimal digits, then a temporal word. We shoudl probably replace "then" with "followed by".
  Straw Poll: Does WG14 want to adopt n3291, with a note to the editor to look at the word "then", into C2y? 10+8-0-3 consensus

- **Thomas**, Proposal for C2Y - missing HUGE_VAL suffixes n3303 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3303 into C2y? 10+8-0-2 consensus

- **Thomas**, Proposal for C2Y - leftover dependency on WANT macro (updates N3304) n3305 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3305 into C2y? 10+7-0-0+3 consensus
  Straw Poll: Does WG14 want to add n3305 onto the list of retroactive papers on our website? 9+4-0-1+6 consensus

- **Thomas**, C2Y proposal - wording for pole error n3324 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3324 into C2y? 10+4-0-7 consensus

- **Gustedt**, Tail recursion for preprocessor macros n3307 (0.5 hours)

  Douglas: I recommend that you talk to Edward Beamer about his library, which does something similar. This does not make the preprocessor Turing-complete, right?
  Gustedt: It does not add state to the preprocessor.
  Celeste: I love this. I think it is necessary.
  Myers: I didn't find any problems with the wording. If we do this, we want C++ compatibility. I am not convinced with making the processor more complicated.
  Ballman: Assuming we encourage this, would you bring this to the C++ committee?
  Gustedt: I would try to get **Jabot** or the Compatibility SG to present it.
  Opinion Poll: Does WG14 want something along the lines of N3307? 5+4-0-1+4+7
  Pygott: My vote was based on the fact that this adds huge complexity, and does programming in the preprocessor.

Lunch Break

- **Celeste**, The `void`-_which-binds_, v2: type-safe parametric polymorphism n3316 (0.5 hours)

Ballman: Type attributes are painful, they make me uncomfortable.
Celeste: So keywords are better than attributes? OK
Bhakta: I prefer attributes over keywords.
Ballman: Do you want these things to be ignore-able?
Celeste: I want to stick to the rule: if you remove all of these the program remains correct and retains its meaning.
Bhakta: This is a very elegant paper. Casting to generic function types, I didn't like that. Even if its a static function, not externally visible, it still adds a layer between the calls.
Kaye: I am in favor of adding more ability to check things. What do you think of adopting this with attributes, and gauging the committee's feeling about keywords after implementation experience?
Ballman: It is easier to gain implementation experience using attributes.
Colomar: I love the idea. I want to see an actual implementation.
Myers: I commented on this on the reflector, in particular the ignore-ability question for attributes.
Celeste: I would like a poll question like, "Can an attribute impose a constraint?".
Ballman: Macro usage might be a red flag, because people will put these behind macros, for older compilers that don't support the new features.
Opinion Poll: Does WG14 want something along the lines of N3316? 6+5-1+1-2+4
Opinion Poll: Does WG14 want something along the lines of qual_var and qual_with in N3316? 0+2-1+-0-8+7

- **Múgica**, Array subscripting without decay (updates n3352) (0.5 hours)

  Bhakta: In section 3, that's a bad idea; there is no notion of "top level". For part 2, swapping "array" with "int index" is only done for job interview questions. Developers on CISC machines often write assembly code that translates to C this way.
  Myers: You've got both option 1 and option 2, which option should we vote for?
  Gustedt: Option 1 is for if the address is never taken (x.y => address of x is not taken). For option 2, suppose the address is not taken if index is an integer constant expression. I prefer option 1.
  Ballman: Is there implementation experience, such as changes of constant expressions?
  Gustedt: For registers, yes. GCC only complains in strict mode. For constant expressions, I don't know.
  Celeste: In the constraint, the value should be non-negative, but should it be less than the size?
  Gustedt: Yes, I think so too.
  Ballman: I know we can have const expression arrays. Removing decay & pointer arithmetic might mess with compilers that expect decay.
  Meneide: For constant expressions, every compiler does sanitize and check. But GCC & Clang only complain on array[-1] if optimizations are turned on.
  Bhakta: With regard to negative indices, I've seen it done. This doesn't make it a good idea.
  Meneide: Clang generates errors all the time, GCC only when in an optimization mode.
  Steenberg: C is complicated with arrays. The mistake in C is that sizeof(array) gives you array size, when it should always give you the pointer size. People want arrays to be more like a basic type, like pointers. I believe the opposite.
  Bazley: I support this proposal.
  Ballman: What would help me support the paper is if it was implemented in a compiler that was then used to build Linux or similar code.
  Meneide: **Celeste**, when you said this will open up space of the operators, I'm curious as to why?
  Celeste: Hmm, probably not. As of C++23, "break" is another operator function call.
  Myers: There is another option. We could say this feature is obsolescent, rather than removing it altogether.
  Bhakta: We've moved away from always wanting implementation experience, but I'd like to see more experience before we put this into the standard.
  Ballman: There are more than 10,000 hits on arrays with negative indices on sourcegraph.com.
  Meneide: **Jabot** compiled a bunch of code against his multidimensional array work. When you have a risk of breaking code, you have to do it. This is not super-onerous, and clang is a nicer codebase. So this is not as difficult as people make it out to be.
  Celeste: Perhaps this should be split into several papers, which we can tackle one-by-one?

Seacord: Don't go for a "yes" vote, try to get the info you need to move forward.
Opinion Poll: Does WG14 want something along the lines of N3360 into C2y? 8+4-1+1-1+4
Opinion Poll: Does WG14 object to breaking index[array] as in N3360? 5+3-3+2-2+4
Opinion Poll: Does WG14 want to deprecate index[array] as in N3360 in C2y? 8+2-0+1-2+6
Opinion Poll: Does WG14 want to make a constraint violation out of negative integer constant expressions used as subscripts of an array (not a pointer) as in n3360 in C2y? 8+6-1+0-1+4

- **Bhakta**, Give consistent wording for SNAN initialization v3 n3364 (0.5 hours)

  Meneide: Alternative 1 covers what Joseph asks for. Alternative 2 covers thread-storage duration.
  Straw Poll: Does WG14 want to adopt Alternative 1 in n3364 into C2y? 10+3-0-5 adopted

- **Celeste**, Case range expressions v3.1 n3370 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3370 into C2y? 9+8-0-1+2 adopted
  Opinion Poll: Would WG14 like to see further work on ranges for designators? 8+8-0-0+2+2

Thursday, 3 October

- **Mollenkopf**, Abs Without Undefined Behavior n3349 (0.5 hours)

  Bhakta: I liked this paper. It is short.
  Gustedt: I liked it. I am not comfortable with the names. We are claiming more of the non-prefix space.
  Kaye: I like the paper, and the names.
  Łukasiewicz: Given the problems with intmax_t, I'm not sure I'd want more of it.
  Ballman: I searched sourcegraph.com: There are 82 uses of "uabs", so there is some risk of breaking code with these names.
  Gustedt: We usually use suffixes for the types.
  Bazley: Is there any consideration to putting these in a separate header file? *No*
  Kaye: If a program uses these names they should get loud compiler errors, right?
  Ballman: The bigger problem is linking, where you don't get loud errors.
  Myers: Many more processors have the basic abs() function.
  Svoboda: -INT_MIN = INT_MAX+1 < UINT_MAX so there should be no undefined behavior.
  Seacord: I sense strong support. There is some concern about the names. I sense support for a new header file for these functions.
  Celeste: We could put these functions in a separate header and deprecate linking to these names.
  Seacord: Could we deprecate or remove abs() from the standard? It would still be supported.
  Ballman: There would be minimal gain for the rename. I'd like the poll for the names as is. If that fails, we can try renaming.
  Straw Poll: Does WG14 want to adopt n3349 into C2y? 8+10-1+0-1+3 consensus
  Meneide: If we made a new header we could alias these with cleaner names?

- **наб**, stdarg.h wording n3363 fka n3285 (0.5 hours)

  Bhakta: Can we have this paper be provisionally voted in if n3665 gets in?
  Gustedt: I prefer this paper as it is.
  Ballman: I suggest voting for the paper as is.
  Svoboda: I'm happy with this paper. But I'd like to also see "variadic function" defined in our Definitions chapter.
  Celeste: C++ doesn't define "variadic function".

- **Colomar**, New _Lengthof() operator (v4) n3369 (0.5 hours)

  Bazley: Is there any potential ambiguity about signedness?
  Colomar: I would like it to return a size_t, just like sizeof. It should not be explicit.
  Ballman: The value of the result of sizeof is size_t. (according to section 6.5.4.5, paragraph 5)

Seacord: I am voting against this, because I don't like the name change. Right now it rubs me the wrong way. I don't like having _Lengthof return a size_t.
Kaye: We already decided on the _Lengthof name. Why do we not have a header file?
Colomar: 8-5 wouldn't be a strong agreement.
Ballman: With regard to header files, we decided to see if users request it. With regard to naming, we should not spend committee time bikeshedding the name.
Bazley: I don't want to re-discuss the name.
Meneide: If I did a survey on the name of this operator, would that enable re-opening the discussion? *Yes*

Straw Poll: Does WG14 want to adopt n3369 with the changes in reflector message 26815 into C2y? 6+9-3+0-1+4 consensus
Bazley: I am Not sure that "count of" is grammatical English.

- **Meneide**, More Modern Bit Utilities, r4 n3367 (0.5 hours):

  Colomar: Regarding the wording for section 7.18.21, paragraph 4, generate_count argument: When do you need it to be a non-negative value if the type is unsigned? Why not convert -1 to an unsigned value?
  Meneide: If you have the macro and you put 1 in there for stdc_rotate_right(). If you only take unsigned values, you can't pass normal literals to say "a non-negative value but signed or unsigned type". Leaving it as undefined behavior allows us to define it later.
  Colomar: Do you have any plans to do the merge from generic count & return type?
  Meneide: Technically this is editorial, I would do this after this paper was accepted.
  Straw Poll: Does WG14 want to adopt n3367, with the un-sequenced attributes removed, into C2y? 8+11-0-2+2 consensus

- **Meneide**, Restartable Functions for Efficient Character Conversion r13 n3366 (0.5 hours)

  Straw Poll: Does WG14 want to adopt n3366 into C2y? 8+3-0-2+7 consensus
  Bhakta: I abstained because there was not enough time to study it.

- **Celeste**, Named loops, v3 n3355 (0.5 hours)

  Ballman: I retract my objection. _pragma is a unary operator, but it is in the preprocessor. Here is an editorial request: It would be nice if we have an example of a named-loop do-while construct. You could editorially add one.
  Bazley: I'm sympathetic, but dislike having new syntax. I like that you can use the labels for both purposes.
  Colomar: This makes readability worse than goto.
  Łukasiewicz: With code navigation I can go to the brace that I want.
  Gustedt: I am voting against this because we should not add control. This one is bad because if you have nested loops, all named, and you have break/continue statements it becomes spaghetti code.
  Bazley: Goto is still considered harmful. None of my colleagues reported any dissent on this paper. Even colleagues who use goto freely still want this feature.
  Johnson: Labels we add for continue and break apply to goto as well. They enable more spaghetti code.
  Bazley: How can these labels branch inwards?
  Johnson: This will litter code with many more identifiers, and doesn't prevent identifiers being used as goto targets.
  Bazley: It is easy to search for goto in source code. I have never tried to automatically search for labels. I don't see how labels will encourage more goto statements.
  Straw Poll: Does WG14 want to adopt n3355 into C2y? 5+5-2+4-3+3
  Ballman: I abstained because I like the feature but I think it will cause confusion. This conflicts too much with goto labels.
  Chronister: This is syntactically the same feature in other languages, and this makes it a tractable problem.
  Svoboda: I feel we don't have consensus and need to think about it more. Perhaps we can re-discuss this in Graz?

Gustedt: I'm against this. We have other means to improve control flow in C.
Bazley: Not every label is a goto target.
Johnson: I'm fine with goto's. This is a good idea.
Straw Poll: Does WG14 want to adopt n3355 into C2y? 5+6-3+4-2+2 consensus

6. Clarification Requests

The previous queue of clarification requests has been processed.

7. Other Business

Time permitting:

- **Meneide**, Integer Constant Expression-Initialized const Integer Declarations are Implicitly constexpr, r0 n3333 (0.5 hours)
- **Meneide**, _Record types, r0 n3332 (0.5 hours)

Scheduled for Graz:

- **Celeste**, "auto" as a placeholder type specifier n3339 (0.5 hours)
- **Thomas**, Proposal for C2Y - frexp and double-double n3357 (0.5 hours)
- **Thomas**, Proposal for C2Y - lingering references to imaginary types n3358 (0.5 hours)

Old papers that haven't been scheduled yet pending request from the author:

- n2650 2022/01/03 **Bachmann**, Make pointer type casting useful without negatively impacting performance - updates n2484
- n2995 2022/06/22 **Meneide**, __supports_literal needs serious design tweaking and there are likely better, less overt ways to solve the same problem
- n3160 2023/08/20 **Grüninger**, Add min, max for integers to C
- n3121 2023/04/23 **Uecker**, Forward Declaration of Parameters v2 (Updates n2780)

The following paper(s) will not be ready for this meeting:

- **MacDonald**, Defect with wording of restrict specification n3025
- **MacDonald**, Provenance-Style Specification of restrict n3058

8. Recommendations and Decisions reached

8.1 Review of Decisions Reached

Do we vote to send a communique from Reflector Message 26702, with a note for what C89 and C94 are to MISRA? 19-2-2 consensus
Does WG14 want to adopt n3275 for C2y? 15-1-9 consensus
Does WG14 wish to adopt n3286 into C2y? 16-0-6 consensus
Does WG14 wish to adopt n3287 into C2y? 18-0-4 consensus
For n3326, do you prefer wording 1 or wording 2? 6+3-3+3-1+8 preference for wording 1
Does WG14 wish to adopt wording 1 from n3326 into C2y? 10+9-0-3 consensus
Does WG14 wish to adopt n3353 into C2y? 8+7-1+0-1+0 consensus
Does WG14 wish to adopt n3356 into C2y? 7+8-1+0-3+0 consensus
Does WG14 wish to adopt n3312 into C2y? 7+4-1+0-2+5 consensus

Does WG14 wish to adopt n3340 into C2y? 10+11-0+0-0+0 passes
Does WG14 wish to adopt n3341 into C2y? 9+10-0+0-1+0 consensus
Does WG14 wish to adopt n3342 into C2y? 10+9-0+0-1+0 consensus
Does WG14 wish to adopt alternative change 1 in n3344 into C2y? 10+7-0-2 consensus
Does WG14 wish to adopt n3345 into C2y? 10+2-0-6 consensus
Does WG14 wish to adopt n3346 into C2y? 9+4-0+0-1+5 consensus
Does WG14 wish to adopt n3347, with the change suggested by Reflector Message 26758, into C2y? 9+7-0+0-1+1 consensus
Does WG14 want to adopt n3298 into C2y with the change of "real float" to "real floating" and without sub-clause 4.2? 9+7-0-1+4 consensus
Does WG14 want to adopt n3327 into C2y? 0+1-8+7-2+1 no consensus
Does WG14 want to adopt Option 1 from n3272 into C2y? 3+2-2+2-5+8 not consensus
Does WG14 want to adopt Option 1 from n3272 into C2y with "undefined behavior" replaced with "implementation-defined behavior"? 4+7-1+2-5+4 consensus
Does WG14 want to adopt n3322 into C2y? 9+9-1+2-0+1 consensus
Does WG14 want to add n3322 onto the list of retroactive papers on our website? 8+8-2+1-0+2 consensus
Does WG14 want to adopt n3323 into C2y? 10+8-0-1 consensus
Does WG14 want to adopt n3291, with a note to the editor to look at the word "then", into C2y? 10+8-0-3 consensus
Does WG14 want to adopt n3303 into C2y? 10+8-0-2 consensus
Does WG14 want to adopt n3305 into C2y? 10+7-0-0+3 consensus
Does WG14 want to add n3305 onto the list of retroactive papers on our website? 9+4-0-1+6 consensus
Does WG14 want to adopt n3324 into C2y? 10+4-0-7 consensus
Does WG14 want to adopt Alternative 1 in n3364 into C2y? 10+3-0-5 adopted
Does WG14 want to adopt n3370 into C2y? 9+8-0-1+2 adopted
Does WG14 want to adopt n3349 into C2y? 8+10-1+0-1+3 consensus
Does WG14 want to adopt n3369 with the changes in reflector message 26815 into C2y? 6+9-3+0-1+4 consensus
Does WG14 want to adopt n3367, with the un-sequenced attributes removed, into C2y? 8+11-0-2+2 consensus
Does WG14 want to adopt n3366 into C2y? 8+3-0-2+7 consensus
Does WG14 want to adopt n3355 into C2y? 5+6-3+4-2+2 consensus

8.2 Review of Action Items

Svoboda: Re-submit the minutes as specified above.3
Ballman: Add n3286, n3322, n3305 to the list of retroactive changes.

9. Thanks to Host

10. Adjournment

End