WG21 2016-06 Oulu Minutes

ISO/IEC JTC1 SC22 WG21 N4597 — 2016-07-11

Jonathan Wakely, cxx@kayari.org

June 20 - 26, 2016 - Oulu, Finland

Chair: Clark Nelson

1. Opening activities (Monday 9:00)

1.1 Opening comments, welcome from host

Voutilainen welcomed everyone to the meeting and to Oulu. Spicer explained the attendance sheet rules and document signup.

1.2 Meeting guidelines

Every participant is responsible for understanding and abiding by the INCITS Antitrust Guidelines and Patent Policy and the ISO Code of Conduct.

1.3 Membership, voting rights, and procedures for the meeting

The chair explained the membership and voting rights for INCITS members, and voting for ISO global directory members.

1.4 Introductions

Representatives from the following countries: Bulgaria, Canada, Finland, France, Germany, Russia, Spain, Switzerland, UK, US.

1.5 Agenda review and approval

Agenda is in a revision of N4590, on the wiki.

Wakely moved to adopt the agenda, Clow seconded. Approved by unanimous consent.

1.6 Editor's reports, approval of working drafts

Document	Editor's report	Prospective working draft
C++ Standard	N4593	N4594
Library Fundamentals V2 TS	N4585	N4584

Document	Editor's report	Prospective working draft
Networking TS	N4589	N4588
Modules TS	-	N4592

Approval of the minutes of the previous meetings (PL22.16 motion)

Meeting	Minutes
WG21 Jacksonville	N4586
PL22.16 Jacksonville	N4587
WG21 pre-Oulu administrative telecon	N4595

Sutter reported that the ballot on the new work item for Modules is active, the results aren't in yet, but we already have a working draft.

Josuttis asked how we will make time to incorporate any new features into the library. Meredith gave an example of default comparison operators, which affects existing types in the library. Sutter requested other groups to give a heads up to LWG about relevant changes.

Boehm asked why Parallelism TS documents (N4578 and N4579) are not in the list of working drafts. Nelson said there was no motion in Jacksonville to change the working draft.

Working papers adopted by unanimous consent.

Hedquist requested deferral of approving the PL22.16 Jacksonville minutes until Friday, pending an amendment.

Voutilainen noted that N4595 has an incomplete sentence, so will require an updated document.

WG21 minutes approved by unanimous consent.

2. Liaison reports, and WG21 study group reports (see pre-meeting WG21 telecon minutes)

3. WG progress reports and work plans for the week (Core, Evolution, Library, Library Evolution)

Reports are in the telecon minutes.

It was suggested to hold joint LWG+LEWG sessions for material affecting C++17.

Yasskin checked support for adding variant to C++17.

4. New business requiring action by the committee

None.

5. Organize working groups and study groups, establish working procedures

(Clarify rooms available for evening sessions)

6. WG and SG sessions

The WG and SG chairs must arrange for any proposals to be written up in the form of a motion, and made available by 2:30 Friday.

7. Review of the meeting (Friday 2:30)

Revised agenda on the wiki with updated PL22.16 minutes, requested to read before the deferred motion to approve them which will happen on Saturday.

WG and SG status and progress reports. Presentation and discussion of proposals to be considered for consensus adoption by full WG21.

SG5: Transactional memory (Wong)

No SG5 meeting, but did have one paper. Continue to meet by telecon, and watch for usage experience. GCC 6.1 now shipped with an implementation.

SG6: Numerics (Crowl)

Brown reported that SG6 did not meet.

SG7: Reflection (Carruth)

SG7 had a long evening session. Two actions of note. Reviewed the entirety or a promising introspection paper, with the author present. Received general support within the study group. Discussed the largest design issue for over an hour and gave the author direction, so a revised paper will be coming. Significant progress but still in the design phase. The design issue was how to handle introspection for things such as typedefs that are not part of the C++ type system.

Voutilainen asked about progress of introspection of private data members. Carruth said there was no progress since the last discussion which agreed it should be possible, but not silent, i.e. separate from introspection of public members.

SG10: Feature test (Nelson)

SG10 did not meet. Newlson drew attention to the fact that SD-6 already tracks all the new features in the working paper, so there is no need to repeat the work of going through motions to create lists of new features. Every document that gets voted on is listed in SD-6, noting how the feature can be detected (or saying so if there is nothing worth detecting).

SG12: Undefined and unspecified behavior (Dos Reis)

SG12 did not meet.

SG14: Games & low latency (Wong)

SG14 had an evening session, shepherding various papers through. Paper on exception handling. Some features are getting a lot of traction. Hoping to get people to submit some papers. Four major driving features are heterogeneous computing, affinity for CPU cache, lightweight exception handling. Continuing to investigate into various topics with crossover with SG1.

Now planning to look into issues affecting the embedded community, after reaching out to game and then finance communities.

Upcoming meeting in Amsterdam with high-frequency traders. Another meeting C++ in Berlin. Sutter requests that agendas and meeting notices be published.

SG1: Concurrency (Boehm)

SG1 met all week. Talked about `joining_thread` but didn't get consensus for that to go forward. Discussed when atomics can be optimized, no consensus on that either. Discussed a few things that aren't making it into C++17 but NB comments are expected. Issue over where and when destructors run. Cleanup of signal handling wording. Almost ready work on synchronized ostreams.

Going forward for C++17 are better names for parallel execution policies; an advisory statement on `memory_order_consume`; a change to parallel algorithms; and wording for forward progress guarantees.

Discussed Concurrency TS v2 and Parallelism TS v2 features. Close to finalizing atomic floating point types, and atomic views, and emplace for promise and future. For Parallelism, datapars is moving along.

Evolution (Voutilainen)

Finished design of important C++17 facilities, for real this time.

Structured bindings and initializers in if statements were totally new features finished and forwarded to Core. Discussed coroutines unification. Reviewed future material such as pattern matching and enouraged future work.

Approved proposals:

- defaulted comparisons
- expression evaluation order
- generic lambda capture with `constexpr_if`
- inline variables

- if-statement with initializer
- structured bindings
- comma elision and comma deletion (not in C++17)
- modernizing using-declarations (not in C++17)
- designated initialization (not in C++17)

A handful of rejected proposals inlcuding operator-dot. Other proposals reviewed will come back with revisions.

Dos Reis requested clarification of the operator-dot status, Voutilainen clarified that it is not being forwarded in its current form, but is expected to return.

Accomplished what was expected. Review continues on Saturday. Thanked Andrew Pardoe for taking notes. Noted that EWG did good work and that co-operation with CWG is smooth as silk.

Library Evolution (Jeffrey Yasskin)

Neil Macintosh will be standing in for Yasskin as chair in Issaquah.

Looked at consequences of default comparisons, will revisit. Decided nothing needs to be done for structured bindings.

Getting more comfortable sending new features straight to the IS rather than to a TS.

Spent a full day reviewing the proposed 2D graphics TS.

Sent 14 new papers to LWG. Made progress on about 10 papers, rejected about 4 and answered questions from LWG about C++17 work.

Continuing to process papers during the remaining time of the meeting.

Thanked the scribes.

Voutilainen asked if the `make_array` proposal was looked at, Yasskin said it wasn't.

Core (Miller)

Reiterated Voutilainen's praise for EWG and CWG collaboration. Thanked Maurer for all his jobs, especially for detailed Core notes.

Did not do any issue processing during the meeting.

Josuttis asked whether issue resolutions could make it into C++17. Miller confirmed that many of them would be ballot resolution comments and so would likely make it into the IS. Sutter explained that the purpose of issuing a CD now was to get more eyes on the draft, to get feedback and discover issues, so we would have a year of tweaks ahead of us.

Sommerlad asked about whether there was any way to vote on proposals for inclusion in a

"C++Next" working draft before the C++17 IS was published. Sutter was not aware of any such mechanism.

Voutilainen asked how many comment ballots there would be. Sutter responded that the CD ballot is the comment ballot. The DIS ballot can receive comments. If an FDIS ballot is required no comments are allowed on it.

Miller reported that CWG processed all papers sent to them for C++17. 14 papers previously approved by EWG will be moved. One paper was referred back to EWG for more work (operator-dot). Two new "not necessarily for C++17" papers were included for C++17 (structured bindings and if-statement initializers).

Dos Reis said the EWG poll on structured bindings was for a preference for C++17.

Snyder asked about the status of the modernizing using declarations paper which was forwarded from EWG. Miller apologized that it wasn't processed.

Stroustrup noted that Core had found genuine inconsistencies in the operator-dot proposal leading to its withdrawal, and thanked core for finding them.

Two papers were approved which needed library review, `has_unique_object_representation` and removing C dependencies from signal handling. The latter wasn't quite able to get through SG1 in time.

CWG Motions

Motion 1 Move to accept as Defect Reports the issues in P0384R0 (Core Language "tentatively ready" issues) and apply their proposed resolutions to the C++ working paper.

Approved by unanimous consent.

Motion 2 Move to accept as a Defect Report issue 26 ("Function concepts not allowed to be declared in more than one TU") in P0396 and apply its proposed resolution to the Concepts TS working paper.

Approved by unanimous consent.

Motion 3 Move to apply to the C++ working paper the proposed wording in P0028R4 ("Using attribute namespaces without repetition").

Approved by unanimous consent.

Motion 4 Move to apply to the C++ working paper the proposed wording in P0035R4 ("Dynamic memory allocation for over-aligned data").

Approved by unanimous consent.

Motion 5 Move to apply to the C++ working paper the proposed wording in P0091R3 ("Template argument deduction for class templates (Rev. 6)").

Josuttis asked for a summary of the final design that was chosen. Spertus said that the ability to give a partial template argument list was removed. Carruth clarified that nothing was added.

In favor: 52 Opposed: 1 Abstain: 13

Approved by consensus.

Vandevoorde explained the objection as being due to preferring that deduction being entirely based on deduction guides, not the mixed form in the proposal.

Motion 6 Move to apply to the C++ working paper the proposed wording in P0127R2 ("Declaring non-type template parameters with auto").

Brown asked whether there had been any discussion of the applicability of this feature for the library. Yasskin said there's a paper about using it in a trait, which hasn't been discussed.

Approved by unanimous consent.

Motion 7 Move to apply to the C++ working paper the proposed wording in P0135R1 ("Wording for guaranteed copy elision through simplified value categories").

Approved by unanimous consent.

Motion 8 Move to apply to the C++ working paper the proposed wording in P0137R1 ("Core Issue 1776: Replacement of class objects containing reference members").

Stroustrup requested a summary of the problem being solved. Smith explained that firstly it improves the clarity and precision of the core language in terms of which parts of the object model have undefined behaviour when using placement-new over an existing object. The second part is a library facility to put a compiler barrier in code to allow storage of objects to be reused, such as in `optional` and `variant`. Voutilainen noted it allows `optional` to actually work.

Approved by unanimous consent.

Motion 9 Move to apply to the C++ working paper the proposed wording in P0145R3 ("Refining Expression Evaluation Order for Idiomatic C++").

Merrill objected to requiring function arguments to be strictly sequenced from left to right, expressing a preference for indeterminate sequencing, to give compilers aditional flexibility. The change has a significant impact on GCC performance. Dos Reis asked whether Merrill's experiments only involved a front end change not back end, which Merrill confirmed. Dos Reis reported that Microsoft's similar experiments showed variance +/- 4%, which was smaller than the GCC experience. He said that the language has the ability to define a strict ordering. Carruth echoed the objection, but also noted that Google changed from one argument evaluation order to another throughout the codebase. They found two things: buggy binary operators, which the paper fixes, and very bad code that made very unsafe assumptions about the structure of arguments and what takes place in them. The code was subtle and hard to maintain. By fixing the code to avoid order dependencies it was simpler and more maintainable and pleased users, so they implemented a compiler mode to prevent users from relying on any particular evaluation order, which the proposal would forbid.

Winters said that optimizers can do better when they have more flexibility, so removing flexibility can only make things worse, or in the best case make no difference. Dos Reis said that if the flexibility helps optimizers then they have not shown evidence of it in 40 years.

Roy noted that there are a number of omnibus papers where people don't agree on every part of a proposal, but the other parts have consensus. Sutter explained that the proposal could be easily modified to remove the one problematic part of the proposal, while retaining the parts everybody is happy with.

Meredith asked for clarifications regarding the difference between using infix operator notation and explicit operator calls. Merrill confirmed that evaluation of `a + b` is not affected by the proposal, but `operator+(a, b)` is.

Stroustrup expressed support for the proposal and noted a longstanding desire to be able to control the order of evaluation, one way or another.

Snyder asked whether there could be a poll for the proposal with and without the controversial part and take the version with stronger consensus, and Sutter confirmed that is an option.

Sankel expressed support from a user perspective as it makes things simpler, and said long term the committee would not regret making this change.

Josuttis asked about the confusion regarding the disagreement on what the feature does.

Spicer explained that the function argument order was originally not in the proposal when brought to Core, then added, and if it's to be removed now it should go back to EWG for discussion.

Winters said all code that depends on a specified order of evaluation for function arguments is code that would not pass code review anyway.

Meredith asked whether it was a surprise to EWG that CWG gave a different answer to his question, or if EWG agreed they were forwarding the same semantics as are being moved.

Voutilainen said the proposal was never intended to specify the evaluation order of everything, as was stated in the proposals.

Finkel expressed sadness when freedom is removed from the compiler, but that in this case operand order is a good thing from a usability standpoint. It can't always be diagnosed anyway.

Carruth apologised for introducing confusion about EWG's understanding of the feature.

Lakos said that being able to rely on evaluation order sometimes but not other times may be worse than never being able to rely on them. Sutter refuted the point, but said it was possible to accept the paper with or without the controversial part.

Giroux expressed concern that for in-order processors the performance impact is likely to be larger. Typical modern processors work very hard to improve on whatever the compiler optimizers couldn't make fast. That means the results might not be relevant for other architectures. Sutter said a CD allows such a feature to be seen and receive feedback. Giroux said the community don't look at such things and expect to be able to trust the committee to not break such contracts between the user and implementation.

Halpern asked what the benefits are of the paper if function argument order is excised. Carruth stated that those others changes are important and fix fundamental problems in the language affecting things like `future.then`

Tong clarified that the weaker form of the proposal would change from unsequenced to indeterminately sequenced with no interleaving.

Sutter took a poll on the proposal with the alternative in section 8.

Favor	Opposed	Abstain
59	5	5

Sutter determined consensus for that alternative.

Sutter took poll on the full proposal.

Favor	Opposed	Abstain
24	24	18

The proposal as modified by Alternative 8 is approved by consensus.

Wakely suggested changing the motion title to reflect the consensus. It was amended to "Move to apply to the C++ working paper the proposed wording in P0145R3 ("Refining Expression Evaluation Order for Idiomatic C++"), using the proposed alternate evaluation order for function calls (section 8)."

Motion 10 Move to apply to the C++ working paper the proposed wording in P0221R2 ("P0221R2: Proposed wording for default comparisons, revision 4").

Voutilainen stated he finds the motivation unconvincing, but doesn't intend to kill it. Van Eerd drew attention to his paper explaining his preference that he dislikes default less-than but likes default-equality.

Brown wished that the library had been able to analyse the library impact. Meredith expressed a similar concern about integration. Josuttis was not concerned with the impact on the library.

Dennett said the proposal was for opt-out, whereas EWG showed strong support for opt-in, but there was no proposal for that.

Spicer expressed a desire to be able to explicitly request the default versions of the functions, as stated on the reflector.

Matthias reported that when he did a quick check in a large codebase he determined that this proposal would require writing more code than today, because the defaults would not work and extra code would have to be written to prevent them causing ppoblems.

Stroustrup responded that there was an opt-in proposal, and a promise to extend the syntax to do some form of that. Van Eerd requested clarification and Stroustrup said to prevent the defaults you must delete or define your own. He imagined opt-in working by letting you request only less-than or only equality, which would prevent both defaults being generated. Van Eerd resplied that that isn't what most people mean by opt-in.

Snyder said this isn't even an opt-out proposal, because there's no way to request the C++14 behaviour. Deleting the operator is not the same as there being none, w.r.t bases that are comparable. Feature has been redesigned two meetings in a row. Had stronger consensus for the simpler Rapperswil opt-in proposal with `=default`, which the author decided not to proceed with.

Favor	Opposed	Abstain
16	31	20

Motion withdrawn.

Motion 11 Move to apply to the C++ working paper the proposed wording in P0283R2 ("Standard and non-standard attributes").

The change allows implementations to ignore unrecognized attributes rather than issue diagnostics.

Approved by unanimous consent.

Motion 12 Move to apply to the C++ working paper the proposed wording in P0292R2 ("P0292R2: constexpr if: A slightly different syntax").

Voutilainen states it is the best thing ever. Garcia disagrees.

Garcia says it is a local partial solution that prevents solving the problems properly later using Concepts. Tong reported that he intends to propose a solution allowing concepts to interact nicely with this feature.

Stroustrup assured people this would cause issues and be used as an argument against concepts.

Vandevoorde said that ability to misuse thigns should not be an argument against them. This fits the language and helps people who don't want to metaprogramme all the time. Van Eerd agreed people would misuse it, but optimistically assumes those people will fix their code to use concepts when they can do so. Garcia expressed a serious concern about the language evolving in the direction of a bundle of unrelated features. Joly believed the feature would be good if it came at the same time as concepts, but without concepts is dangerous. Van Eerd suggested it could be put in the Concepts TS.

Spicer stated the semantics in some contexts are surprising.

Dennett agreed it could be misused, as with regular runtime `if` statements. He reported seeing examples where standard library implementations were broken that could be fixed by this. Yasskin stated that this does some things in a simpler way than is possible with Concepts and so having both is good.

Favor	Opposed	Abstain
48	2	17

The Spanish NB is opposed. The French NB does not have a formal position yet, but could be opposed.

Motion 13 Move to apply to the C++ working paper the proposed wording in P0296R2 ("P0296R2: Forward progress guarantees: Base definitions").

Approved by unanimous consent.

Motion 14 Move to apply to the C++ working paper the proposed wording in P0299R1 ("P0299R1: Forward progress guarantees for the Parallelism TS features").

Approved by unanimous consent.

Motion 15 Move to apply to the C++ working paper the proposed wording in P0386R2 ("Inline Variables").

Dos Reis says the feature has been discussed in various forms, at the previous meeting, earlier this week, and again very recently, and asks if there was another late change of direction, and Voutilainen confirms there was.

Stroustrup says he dislikes it, that it makes it easier to use global state, that it's a short-term patch for something Modules fixes better.

Spicer says it provides important and useful functionality, giving a way to avoid the implicit ODR violations people have been writing for a long time. Uses the mechanism that's already there via variable templates, and if we don't add it people will just use variable templates to do the same thing.

Voutilainen clarified whether `constexpr` variables at namespace scope are implicitly inline. They are no longer inline, but static constexpr data members still are implicitly inline. Dennett notes that reverts to an earlier state of the proposal, it's not a brand new invention.

Dos Reis says that fixing the ODR wording would give a better fix for the problem this proposal introduces a hack to solve.

Finkel says the proposal firstly gives a way to easily declare constant strings in header files without duplicating data in every translation unit, thus reducing code bloat or use of non-standard extensions. Secondly it gives an easier way to make more libraries header-only, helping libraries that are mostly header-only but have a small handful of source files just to define some variables. Boost has a few such libraries. This removes the need for linking to such libraries in build systems, so they can be header-only, which increases the adoption of libraries.

Garcia asks whether depending on the context a variable is inline or not? Smith says the only place you get an inline variable when you don't say `inline` is for static data members, so they no longer need a separate definition to avoid linker errors. Smith and Dos Reis disagree on whether Modules would obviate the need for this feature for static member variables. Miller confirms that this does reduce common misunderstandings regarding separate definitions of static data members. It may not be the best way to solve the problem or the best part of he proposal, but it does simplify teaching the use of static data members with in-class initializers.

Dos Reis says the static data member case is not controversial, that can be fixed just by fixing the language. The controversial part is that at namespace-scope things do get more complicated. This adds a quick-fix hack to the standard instead of taking the time to fix the language properly.

Botet-Escriba says we don't have wording for how to fix things properly, but we can have this now for C++17 and fix it more cleanly next time.

Dennett is confident he can teach the proposed rules, but many of us have long experience that pople cannot be taught the current rules.

Yasskin requested more information on Dos Reis's suggestion of tweaking ODR, and whether that means making all static members into weak symbols.

Spicer says this can't be fixed in the ODR without making everything an inline variable, which doesn't solve the problem, you end up with variables with different addresse. Modules don't fix this, as you can have the same template instantiated in different modules.

Carruth said the ODR change was discussed in EWG and didn't get consensus.

Stroustrup said the static member case can be handled multiple ways. This isn't just about constants, more concern about global variables. Finally, the name, Dennis Ritchie lived to regret reusing `static` and that we'll get a misnamed feature just like that with this feature.

Favor	**Opposed**	**Abstain**
45	6	17

Objections do not currently represent NB positions.

Approved by consensus.

Motion 16 Move to apply to the C++ working paper the proposed wording in P0391R0 ("P0391R0: Introducing the term 'templated entity'").

Introduces a proper name for "temploid" which was viewed as overly informal, which will allow more accurate and less error-prone specification in future.

Approved by unanimous consent.

Motion 17 Move to apply to the C++ working paper the proposed wording in P0217R3 ("P0217R3:

Proposed wording for structured bindings").

Meredith expressed concern that this was not on the schedule at Jacksonville. Vandevoorde says the design is the one accepted in Jacksonville, the only difference now is we have complete wording.

Van Eerd expressed concern at the bit-field part, so there was less "make stuff up"-ish, leaving bit-fields for later. Merrill said the naming of a bit-field uses a mechanism that is analogous to anonymous unions, where you have these names that mean a member access expression.

Favor	Opposed	Abstain
47	2	20

Approved by consensus.

Motion 18 Move to apply to the C++ working paper the proposed wording in P0305R1 ("Selection statements with initializer").

Meredith raised the same objection about scheduling.

Favor	Opposed	Abstain
58	2	8

Approved by consensus.

Motion 19 Move to apply to the C++ working paper the proposed wording in P0398R0 ("P0398R0: Core issue 1518: Explicit default constructors and copy-list-initialization").

Van Eerd asks whether there had been time to check with LWG that this was what they wanted. Wakely, Voutilainen and Miller confirmed this was the same fix as had been in drafting and LWG wanted it.

Library (Clow)

Only one paper that didn't get processed. Some issue processing done. Worked into the night all week. Thanked the working group attendees and scribes.

Clow polled interest in an extra meeting in August, counting at least 17 people.

LWG Motions

Motion 1 Move we apply the resolutions of the following issues in "Ready" status from p0165r2 to the C++ Working Paper:

2549, 2393, 2542, 2436, 2550, 2310, 2181, 2328, 2667, 2669, 2671, 2673, 2670, 2441, 2426, 2309

Approved by unanimous consent.

Motion 2 Move we apply the resolutions of the following issues in "Tentatively Ready" status from p0165r2 to the C++ Working Paper:

2710, 2685, 2698, 2596, 2684, 2689, 2688, 2707, 2674, 2706, 2683

Approved by unanimous consent.

Motion 3 Move we apply the resolutions of the following issues in "Ready" status from p0165r2 to the Library Fundamentals 2 Working Paper:

2555, 2451, 2573, 2551, 2516

Voutilainen noted the issue is important and should be applied to the C++ WP.

Approved by unanimous consent.

Motion 4 Move we apply the resolutions of the following issues in "Tentatively Ready" status from p0165r2 to the Library Fundamentals 2 Working Paper:

2509

Approved by unanimous consent.

Motion 5 Move we apply the resolutions of the following Priority 1 issues in "Immediate" status from p0397r0 to the C++ Working Paper:

2687, 2704, 2711, 2725

Approved by unanimous consent.

Motion 6 Move we apply the resolutions of the following issues in "Immediate" status from p0304r1 to the C++ Working Paper:

2312, 2422, 2709, 2716, 2718, 2718, 2719, 2720, 2721, 2723, 2724, 2726, 2727, 2728

Approved by unanimous consent.

Motion 7 Move we apply to the C++ Working Paper the Proposed Wording from p0063r3, C++17 should refer to C11 instead of C99

Nelson told Vandevoorde it was largely editorial. Stroustrup stated a dislike of Annex K, and Nelson said that there is no requirement to add them to C++ implementations. The other optional parts of the C library such as atomics and thread facilities are also excluded.

Favor	Opposed	Abstain
55	0	12

Approved by consensus.

Motion 8 Move we apply to the C++ Working Paper the Proposed Wording from p0175r1, Synopses for the C library

Approved by unanimous consent.

Motion 9 Move we apply to the C++ Working Paper the Proposed Wording from p0088r3, Variant: a type-safe union for C++17

Approved by unanimous consent, to acclamation.

Motion 10 Move we apply to the C++ Working Paper the Proposed Wording from p0307r2, Making Optional Greater Equal Again

Favor	Opposed	Abstain
48	0	17

Approved by consensus.

Motion 11 Move we apply to the C++ Working Paper the Proposed Wording from p0393r3, Making Variant Greater Equal

Approved by unanimous consent.

Motion 12 Move we apply to the C++ Working Paper the Proposed Wording from p0032r3, Homogeneous interface for variant, any and optional

Naumann stated a problem with a single part. The types are vocabulary types which are meant to be simple to use. In order to use the same tag type for all three of `optional`, `any` and `variant`, it uses references to functions instead of tag structs, which will confuse novices. What we have today, in terms of three different tag types, is simpler for novices to learn. Winters says novices won't use emplace and won't use tag types, but this improves things for the rest of us.

Dos Reis asks if it's possible to separate the controversial part. Botet-Escriba confirmed it would be simple to split it out, but it's not a hack, it's a technique to enable the ideal syntax.

Sankel agreed with Naumann that it's a hack and the only part of the proposal that isn't awesome.

Polukhin says that this technique means you don't need to remember different tag type names for different purposes.

Sommerlad said that IDEs would show a strange type. Koeppe said diagnostics would be confusing too.

Van Eerd says the syntax is easy to learn by example.

Sutter took a poll on the paper as proposed:

Favor	Opposed	Abstain

Favor	Opposed	Abstain
21	6	39

Second poll for the proposal without the changes to the `in_place`, `in_place_type` and `in_place_index` tag types.

Favor	Opposed	Abstain
27	9	28

No consensus.

Suggested to take a three-way poll:

Proposal as written	Amended proposal	Do nothing
34	23	0

P0032R3 as presented has consensus.

Motion 13 Move we apply to the C++ Working Paper the Proposed Wording from p0067r3, Elementary string conversions

Approved by unanimous consent.

Motion 14 Move we apply to the C++ Working Paper the Proposed Wording from p0254r2, Integrating std::string_view and std::string

Clow explains that this reverses the structural dependency between `string` and `string_view`

Approved by unanimous consent.

Motion 15 Move we apply to the C++ Working Paper the Proposed Wording from N4524, If vector::reserve(n) reallocates, capacity()==n

Carruth says the flexibility is important. Wakely responded that he intends to add a feature ASAP which will allow vectors and allocators to coordinate when they want to do something different, and that can be an opt-in which would then remove this guarantee for allocators providing the new feature. Carruth wants this to wait until the new feature is present, so that `std::allocator` can use the new feature.

Meredith says you need to rely on this for allocators that will fail if requested to give extra memory.

Finkel says claiming this helps embedded is not true, because if you're in a constrained environment you need to ensure the OS allocator, `std::allocator` and `std::vector` coordinate anyway, this guarantee wouldn't be good enough.

Van Eerd said having two functions, one to do this and one to possibly overallocate, would be nice.

Tong says that he has an implementation which does try to allocate additional memory, and if that fails it catches the `bad_alloc` and retries the exact quantity.

Smith says that even with this guarantee the `vector` might have to allocate extra memory for its own data member, so there's no guarantee this change ensures exactly the requested size anyway.

Wakely says it works today with all implementations, so no rush to fix it.

Favor	Opposed	Abstain
9	35	22

Motion withdrawn.

Motion 16 Move we apply to the C++ Working Paper the Proposed Wording from p0258R2, has_unique_object_representations

Approved by unanimous consent.

Motion 17 Move we apply to the C++ Working Paper the Proposed Wording from p0040r3, Extending memory management tools

Favor	Opposed	Abstain
46	1	22

Approved by consensus.

Motion 18 Move we apply to the C++ Working Paper the Proposed Wording from p0084r2, Emplace Return Type

Approved by unanimous consent.

Motion 19 Move we apply to the C++ Working Paper the Proposed Wording from p0302r1, Removing Allocator Support in std::function

Wakely notes the original proposal was to deprecate, but after discussion and talking to NB reps it was decided to remove without deprecation.

Approved by unanimous consent.

Motion 20 Move we apply to the C++ Working Paper the Proposed Wording from p0083r3, Splicing Maps and Sets

Joly asks if you can splice a range. Wakely says no, there is a merge function that operates on entire containers. Koeppe asks if you can move from a set of pairs to a map, Wakely says no.

Approved by unanimous consent.

Motion 21 Move we apply to the C++ Working Paper the Proposed Wording from p0181r1, Ordered by Default

Approved by unanimous consent.

Motion 22 Move we apply to the C++ Working Paper the Proposed Wording from p0163r0, shared_ptr::weak_type

Approved by unanimous consent.

Motion 23 Move we apply to the C++ Working Paper the Proposed Wording from p0209r2, make_from_tuple: apply for construction

Approved by unanimous consent.

Motion 24 Move we apply to the C++ Working Paper the Proposed Wording from p0295r0, Adopt Selected Library Fundamentals V2 Components for C++17

Approved by unanimous consent.

Motion 25 Move we apply to the C++ Working Paper the Proposed Wording from p0174r2, Deprecating Vestigial Library Parts in C++17

Roy says that what is vestigial to one person is not vestigial to another, and requests that in future deprecating separate components should be proposed separately.

Voutilainen asks what the replacement for `std::iterator` is, to which Van Eerd responded "typing".

Favor	Opposed	Abstain
41	1	23

Approved by consensus.

Motion 26 Move we apply to the C++ Working Paper the Proposed Wording from p0337r0, Delete operator= for polymorphic_allocator

Approved by unanimous consent.

Motion 27 Move we apply to the C++ Working Paper the Proposed Wording from p0358r1, Fixes for not_fn

Approved by unanimous consent.

Motion 28 Move we apply to the C++ Working Paper the Proposed Wording from p0219r1, Relative Paths for Filesystem

Approved by unanimous consent.

Motion 29 Move we apply to the C++ Working Paper the Proposed Wording from p0392r0, Adapting string_view by filesystem paths

Approved by unanimous consent.

Motion 30 Move we apply to the C++ Working Paper the Proposed Wording from P0394r4, Hotel Parallelifornia: terminate() for Parallel Algorithms Exception Handling

Adelstein-Lelbach explains that you can throw any time you like, but you can never leave. It removes `exception_list` and says that parallel algorithms throw.

Sankel expresses discomfort with the last-minute change with no time to analyse the repercussions. Adelstein-Lelbach says we have had no time to use or analyse `exception_list` properly either. Sankel says he has users who do want to use exceptions in parallel code.

Boehm says that if we're going to do anything with `exception_list` we need to do it now, for C++17. Halpern says it's the most conservative solution for C++17. Fixing `exception_list` would be risky, and hard to change later without ABI breaks. You can handle exceptions in parallel code, you just have to do it yourself, not rely on the algorithms to collect them for you. There are some users of exceptions in parallel code, but far more who go out of their way to avoid them. The plan is to add execution policies later which specify doing something more like `exception_list`.

Bastien says that Adelstein-Lelbach was asked to fix `exception_list` at the last meeting, but came back this meeting with an attempt which didn't work, and the new suggestion was to remove it. You can still handle exceptions, they just can't be thrown out of your code. The proposal is consistent with exceptions escaping `main` or a `std::thread`. With this fix you don't pay for exceptions if you don't use them.

Stroustrup said two days does seem far too soon to understand a paper. Meredith said that there are longstanding objections to `exception_list` and known problems with it, it's only the specific solution that is new.

Spertus asked if the change could be made in response to an NB comment. Yasskin said nobody was objecting to the nature of the change, just the timeline. There was disagreement. Boehm suggested it would be easier to take this preferred direction and address this via NB comments, rather than receive comments on something we know to be broken.

Favor	Opposed	Abstain
54	2	11

Approved by consensus.

Motion 31 Move we apply to the C++ Working Paper the Proposed Wording from p0336r1, Better Names for Parallel Execution Policies in C++17

Approved by unanimous consent.

Motion 32 Move we apply to the C++ Working Paper the Proposed Wording from p0371r1, Temporarily discourage memory_order_consume

Dos Reis asked when we will stop discouraging it, Boehm said it won't be fixed in time for C++17, but maybe by 2020.

Approved by unanimous consent.

Motion 33 Move we apply to the C++ Working Paper the Proposed Wording from p0346r1, A `<random>` Nomenclature Tweak

Approved by unanimous consent.

Motion 34 Move we apply to the C++ Working Paper the Proposed Wording from p0180r2, Reserve a New Library Namespace Future Standardization

Meredith clarified that it doesn't require vendors to diagnose uses of such namespace names, but it puts people on notice such names might be used in future. National bodies can suggest better names that we should reserve.

Joly said it's a bad name and we would be forced to use that name later. Winters and Carruth say it's the best name that was come up with, but we're not bound to use it.

Dos Reis asks whether this should be in the CD if it doesn't require anything from implementations. Responses that if we're ever going to try to squat on a namespace name, the best time to do so is now.

Favor	Opposed	Abstain
61	1	7

Approved by consensus.

WG21 Motions

Move to appoint an editing committee composed of Marshall Clow, Mike Miller, Ville Voutilainen and Jeffrey Yasskin to approve the correctness of the C++ working paper as modified by the motions approved at this meeting, and to direct the Convener to transmit the approved updated working paper for CD ballot.

Meredith thinks there are a lot of changes going into the draft, with no time to analyze how they interact. Don't feel the document is in a suitable condition to send to ballot. Carruth believes the working groups, especially CWG and LWG, have spent significant time analyzing how the features integrate. Sutter said this document is in much better condition than when entering the last CD ballot, there were thanks for those who made that happen. Yasskin said there has been integration work, e.g. on `string_view` and this is ready. Wakely said saying there hasn't been time to analyze does a disservice to the groups who do nothing but that, and have been doing that for the last few years with every proposal.

Josuttis would like a discussion on what we can learn from this process.

Meredith thinks we haven't had any time for synthesis of new core features and the library. Sommerlad agrees, but doesn't think delaying a CD will help.

Favor	Opposed	Abstain

Favor	Opposed	Abstain
64	2	2

Approved by consensus, and to acclamation.

8. WG and SG sessions continue (Saturday 8:30)

9. Closing activities (Saturday 1:30)

9.1 Confirm WG21 consensus to adopt proposals ("consent agenda", approved without discussion if no new information)

Regarding Core Motion 9 (order of evaluation with alternative), Miller reported that Core had now looked at the alternative and provided improved wording as guidance to the editor. The new wording was attached to the wiki as P0400R0.

There was no other new information and the proposals were adopted based on the consent agenda.

9.2 PL22.16 motions, if any

Approval of the minutes of the previous meetings, N4596.

Hedquist moved to accept the minutes, Clow seconded. Accepted by unanimous consent.

9.3 Issues delayed until today

There was no motion to adopt a new Parallelism TS at the start of the meeting, but in fact there had been a new working draft since the previous meeting. Approving that new working draft will be done at the next meeting.

Boehm asked about the policy for rebasing an in-flight TS on C++17. Yasskin explained that the ISO rules say a TS can depend on a DIS, but not any draft before that. He recommended that anything going to PDTS now stays against C++14.

10. Plans for the future

10.1 Next and following meetings

Clow counted availability for the Aug 1, 2016 or Aug 15, 2016 meeting (5 days) in Chicago

Aug 1: 9 people Aug 15: 10 people

- 2016-11-07/12 Issaquah, WA, US (N4571)
- 2017-02-27/03-04 Kona, HI, US (N4573)

The Issaquah meeting is the week of the US elections.

Hoping to be done with CD ballot by Issaquah.

Toronto meeting in July. Details are on isocpp.org

Albuquerque, New Mexico, probably in November. More details forthcoming.

Millington asked if there were more plans for meetings in Europe. Sutter explained that the international meeting next year would be Toronto (and the US meeting is also not in continental US), but the 2018 one should be Rapperswil.

10.2 Mailings

Post-meeting mailing deadline is July 11th, pre-meeting deadline is Oct 17th. Deadline is 14:00 UTC.

11. Adjournment

Brown moved to thank the host, unanimous consent.

Brown moved to thank the convener, the officers, the project editor and TS editors, working group chairs, study group chairs, the scribes, the scribes, the scribes, the scribes, the authors and everyone who helped make C++ better.

Hedquist moved to adjourn, Clow seconded. Accepted by unanimous consent.

12. Attendance

The column "WG21" designates official PL22.16 or WG21 status ("P", "A", "E", "M")

The column "PL22.16" indicates organizations eligible to vote by "V".

PL22.16 members

Company / Organization	NB	Representative	WG21	PL22.16
Apple		Duncan Exon Smith	A	V
Argonne National Lab		Hal Finkel	P	V
Bloomberg		John Lakos	P	V
Bloomberg	UK	Alisdair Meredith	A	
Bloomberg	UK	Dietmar Kühl	A	
Bloomberg		Nathan Myers		
Brown		Walter E. Brown	E	
Cisco Systems		Lars Gullik Bjønnes	P	V
Dinkumware		PJ Plauger	P	V

Company / Organization	NB	Representative	WG21	PL22.16
Dinkumware		Tana Plauger	A	
Edison Design Group		John H. Spicer	P	V
Edison Design Group		Daveed Vandevoorde	A	
Edison Design Group		Jens Maurer	A	
Edison Design Group		Mike Herrick	A	
Edison Design Group		William M. Miller	A	
Embarcadero Technologies		David Millington	A	V
Facebook		Lee Howes	A	V
Facebook		Maged Michael	A	
Google		Chandler Carruth	A	V
Google		Geoffrey Romer	A	
Google		Hans Boehm	A	
Google		James Dennett	A	
Google		Jeffrey Yasskin	A	
Google	CA	JF Bastien	A	
Google	UK	Richard Smith	A	
Google		Thomas Koeppe		
Google		Titus Winters	A	
Intel		Clark Nelson	P	V
Intel		Pablo Halpern	A	
Intel		Adam Stanksi		
Lawrence Berkeley		Bryce Adelstein-Lelback	P	V
Los Alamos National Laboratory		Li-Ta Lo	P	V
Microsoft		Jonathan Caves	P	V
Microsoft		Gabriel Dos Reis	A	
Microsoft		Herb Sutter	A	
Microsoft		Gor Nishanov	A	
Microsoft		Andrew Pardoe	A	
Microsoft		Neil Macintosh	A	
Microsoft		Casey Carter	A	
Morgan Stanley		Bjarne Stroustrup	P	V
NVidia		Olivier Giroux	A	V
Oracle		Fedor Sergeev	A	V
Oracle		Maxim Kartashev	A	
Oracle		Danil Tarakanov	A	
Perennial		Barry Hedquist	P	V
Perennial		Beman G. Dawes	A	
Plum Hall	FI	Ville Voutilainen	A	V
Programming Research Group		Christof Meerwald	A	V

Company / Organization	NB	Representative	WG21	PL22.16
Qualcomm		Marshall Clow	P	V
Red Hat		Jason Merrill	P	V
Red Hat	UK	Jonathan Wakely	A	
Red Hat		Torvald Riegel	A	
Sandia National Labs		Carter Edwards	P	V
Sony Computer Entertainment		Michael Spencer		V
Stellar Science		David Sankel	P	V
Symantec		Mike Spertus	P	V

Other WG21 members

Company / Organization	NB	Representative	WG21
	BG	Vasil Vasilev	M
Codeplay	CA	Michael Wong	M
IBM	CA	Hubert Tong	M
Mozilla	CA	Botond Ballo	M
Christie Digital	CA	Tony Van Eerd	M
Universitè de Sherbrooke	CA	Patrice Roy	M
CERN	CH	Axel Naumann	M
Vollmann Engineering	CH	Detlef Vollmann	M
HSR	CH	Peter Sommerlad	M
	CH	Mauro Bianco	M
	DE	Fabio Fracassi	M
University Carlos III	ES	J. Daniel Garcia	M
CryptoTec	FI	Mikael Kilpeläinen	M
	FR	Loïc Joly	M
	RU	Aleksandr Fokin	M
	RU	Anton Polukhin	M
	UK	Dinka Ranns	M
	UK	Jeff Snyder	M
	UK	Jonathan Coe	M
	UK	Neil Horlock	M
	UK	Roger Orr	M

Participating non-members

Company / Organization	NB	Representative
LTK Engineering		Alan Talbot
University of Akron		Andrew Sutton
Jump		Barry Revzin

Company / Organization	NB	Representative
		Brent Friedman
Bruker Daltonics		Daniel Krügler
Markit		David Stone
		Faisal Vali
Blizzard		James Touton
University of Nice		Jean-Paul Rigault
		Marius Huse Jacobsen
Frankfurt Inst. for Adv. Studies		Matthias Kretz
Nokia		Michał Dominiak
Bob Taco Industries		Michael McLaughlin
Mail Ru Group		Mikhail Maltsev
		Nicolai Josuttis
Xilinx		Ronan Keryell
Sabre		Tomasz Kamiński
Nokia		Vicente J. Botet Escriba
Schonfeld		Wesley Maness
ARM		Will Deacon