C and C++ Compatibility Study Group Meeting Minutes (Aug 2021)

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2795
SG Meeting Date: 2021-08-06

Fri Aug 06, 2021 at 1:00pm EST

Attendees

Corentin Jabot	WG21	
Clive Pygott	WG14 (21)	
Aaron Ballman	WG21 WG14	chair
Will Wray	(14) (21)	scribe
Thomas Köppe	WG21	co-chair
Krystian Stasiowski	(21)	
Theodoric Stier	(21)	
JeanHeyd Meneide	WG14 WG21	co-chair
Michael Wong	WG21 (14)	
Miguel Ojeda	WG14 WG21	
Zhihao Yuan	WG21	

Code of Conduct: follows ISO, IEC, and WG21 CoCs (no current WG14-specific CoC)

Agenda

Discussing the following papers:

WG21 P2340R0 (https://wg21.link/p2340r0) Clarifying the status of the “C headers”
WG21 P1997R1 (https://wg21.link/p1997r1) Relaxing Restrictions on Arrays
WG21 P2314R2 (https://wg21.link/p2314r2) Character sets and encodings (Not presented; author was unavailable)

P2340R0 Clarifying the status of the “C headers”

Thomas Köppe, author, presents the paper
This paper was written after a long time mulling.
C++ headers that come from C are somehow part of the standard, but at the same time deprecated; it is not clear what that means.
The C and C++ headers have similar content, and likely share much of the implementation.
Deprecation implies that they are subject to future removal - but they can't be removed. As they can't be deprecated the status quo doesn't reflect intent.
The header definitions should be moved out of the deprecate-annex and into the main document.
The C headers in C++ are for interoperability with ISO C, and all the other systems that use C’s de-facto ABI and linkage conventions.
There should be discussion of what the headers are and are not for.
Guidance on how they are intended to be used.

FIN
Q&A
CP: happy to see this; the current situation makes no sense.
TK: there are legitimate uses, but some use C headers everywhere. So guidance is needed; things are made more difficult by the absence of any discussion in standard.
AB: (to TK) are you seeking polls, or WG14 concerns?
TK: concerns. It's unanimous in WG21. Looking for corrections, heads up, and thoughts.
CJ: "it makes me sad"; even if C header definitions are moved into the C++ standard, WG21 doesn't control what's allowed. Like adding constexpr to C functions.
CJ: My position is to remove the definitions. If C++ depends on C anyway, then that's ok (I understand that this is only my position, and no one else agrees).
TK: It's a constant source of friction. They're "C library" headers only in name; it's really a C++ feature. After all, you can't have a library for a different language; even if the headers are textually identical, what do they mean?
TK: Implementers would protest against any sweeping changes. Interoperability calls for parallel implementations - implementers won't fork the sources. This seems like a problem that cannot be solved? It's just not realistic to remove the definitions.
CJ: [is still sad.]
TK: Freestanding implementations will [almost always] have a C library, and they are free to omit certain headers.
AB: 99% sure that the meaning of headers changes between C and C++. For instance the restrict keyword in C is defined by semantic effect in C++, but then does not modify the library.
AB: When we added the synopsis (https://wg21.link/p0175), used to list names, C++ has overloading on language linkage - i.e. extern "C" / "C++", so there's a notional interface difference (but much the same in practice)
CJ: Implementers push back on constexpr e.g.
AB: It doesn't make things worse to express intent.
TK: There were some dissenting voices when presented to WG21, so thanks CJ for raising some objections.
TS: So, although C++ does change signatures in the library headers, is anyone aware of C projects with semantics changed in C++?
TK: functions that return char [as int in C?] As in string.h, C++ const char* -> char* in C
TS: there's the as-if rule?
AB: Thanks all. This looks likely to move through WG14

P1997R1 Relaxing Restrictions on Arrays

Presented by Theodoric Stier, and Krystian Stasiowski
Not that much to say, so we'll go slowly; please stop and ask questions.
The paper proposes:
- Initialization and assignment of LHS array from another array on RHS.
- Placeholder semantics, deducing element type and/or size.
- Array return type from function.

Arrays are objects just like any other in C / C++ but the standard has special provisions - exceptions for arrays - that actively limit what is allowed.
For instance memcpy is 'enshrined' to do what operator= usually does, then infix '=' is seen as just a convenience. But it's natural, especially for beginners, to reach for '='.
std::array works around these limitations; the cost is template instantiations. Daveed Vandervoorde showed a significant 'abstraction penalty'.
std::array has some backward compatibility with C array, instead of exactly compatibility. It cannot have any constructors for convenience, as it has to remain an aggregate type.

FIN
Q&A
KS: The paper has C interoperability, there are no breaking changes. A future standard is free to add another array type.
JHM: Not clear if it addresses the semantics of unknown bounds. Could be a breaking change. Need to collect thoughts.
AB: What do the authors want? Go straight to EWG, or to WG14? Advice on potential difficulties for WG14? Advice on deprecating formal array parameters?
CP: a practical note: for C23 the October WG14 meeting agenda is full. Knowing what's in the pipeline, and going for vote next year, there's no time for new ideas. This will be postponed - put in the pipeline for the next revision.
AB: On WG14 side, array syntax leaks over into practically all areas. Array parameter syntax is very frequently used. So is VLA passing (Variable Length Array), in which an integer 'n' argument followed by an array T a[n] argument makes a VM type (Variably Modified) which conveys information on the array size to C implementations. We have to be very careful not to break these.
AB: Also, there are static extents in C, which expect n or more elements.
AB: These are hard to deprecate. C++ vendors also implement VLAs, so this impacts C++ even if the standard doesn't include VLAs. Is there implementation experience? WG14 is inclined to reject proposals which don't have any.
TS: We don't have any model implementations. Is this necessary? My suspicion is that, although C arrays are deeply tied to compiler foundations, the implementation needed is Not Heroic. Could be wrong...
TK: Even taken to WG21, there will be voices calling for implementation. This is likely to impact real code in one way or another. Predictions of no-breakage require evidence to back them up.
CP: WG14 is averse to breakage, and to lack of implementation experience.
AB: there are ABI issues to resolve on array as a return type. How is this information going to be passed to the ABI group, to avoid incompatibilities? Say C thinks it's a bounded array, and C++ thinks otherwise?
TS: As far as we are aware, semantic changes in the proposal, as of now, only make invalid code become valid. They don't change the semantics of valid code. The relationship between array type and pointer type is built upon, not changed (but maybe only within C++).
JHM: The paper isn't touching array formal parameters. And, array returns of static extent are banned in both C and C++. So this proposal does seem to match with the 'no impact' prediction.
JHM: The array return extension looks fine, and should work out nicely. The parameter stuff should be left to another paper - ignore for now. Initialization and assignment are enough for now.
AB: [implied by comments] proposing to WG14 makes sense, as the subject matter is very much of interest, even if the timing is off. This can bring a whole lot of value to core. People reach for assignment with '='. Making it work is beneficial.
CP: Even if there's no time for the paper, it is something to flag up.
CJ: For WG21, any decision may need to wait for C++26 target.
AB: (yes always more work...) How does this work for mulitdimensional arrays?
TS: The language is recursive, nested (there are no true multidim arrays) so it just applies, recursively.
JHM: There's an example that copies 2d to 2d; the example checks out.
AB: Poll suggestion: sentiment - do we want to see again in SG22 before EWG? Or, have we given enough guidance, ready to forward on with changes?
JHM: [on implementation experience] Although WG21 doesn't require experience, it will be asked for; even if only a branch that works for you, that will go a long way to not face undue resistance. They'll ask questions about multidimensional array deduction, of multibounds for all dimensions; can mark to leave alone and extend later. There's not much more that this group can do but help to refine any implementation, though nothing special can be done to speed that up. C will be interested a lot in this. The last part is implementation. WG14 group can't really help on that. If we're not speaking about parameters then there's not much difference in arrays between C and C++.

AB: One benefit of coming back to SG22 - it will smoke out problems. (Experience porting real freeBSD code to clang smoked out many issues.) Based on how novel the syntax is for C array, features may look the same but actually have drastic differences.
CJ: I'd question the novel syntax involved in array types - it may not be an issue if it is not touching parameters - I cannot imagine we'll find such a scenario. I still want to see an implementation.
TK: I'd be curious to get early feedback from evolution. It has worked before alright, to ask for an early preview. Once we have an implementation then it is fine to ask. Can keep an eye on it.. not sure.
AB: Poll => Send to ewg?
Authors: agree.
AJB: Does SG22 want to forward P1997 on to EWG for further refinement?
JHM: Stick with "don't touch parameters" (something about templates); it doesn't change formal array parameters legality - see section 4.4. So it's fine from the compatibility aspects. Template stuff is C++ - WG21 will have fun with that.
JHM: We're only dealing with assignment and return types. Need to be careful with conversions. Say if array-of-array-of ... changes something. Nothing is fundamentally changed. Recommend to reach out - listen - but need to spend time; it can be bad to push to be seen early and then get torn apart. So reach out and slip schedule as needed.
CJ: I don't agree with that last point. As TK says - if evolution doesn't like it, then it may not be worth implementing. You can waste a lot of time on a full implementation, so starting an implementation before feedback is best avoided.
AB: Explain the template aspect, please
TS: Template type deduction in variable declarations uses a placeholder syntax, the same as for function template argument deduction. Krystian's approach is surprisingly simple (more than my earlier one) We measured code usage of array parameters in codebases, they're mostly used for passing char arrays.
AB: This proposal seems to make arrays first class objects. So, what about atomics? Could we have atomic assignment of array? At some point. How do you envision array-of-atomic vs atomic-array.
TS: It could be standardised as a partial specialization.
AB: In C++ we have std::atomic so this is much easier to do. On the C side this is more attractive, but may have memory overhead. We have to deal with that for struct, so it can be dealt with for array.
TK: what about auto type deduction with init list? Does the meaning of auto = {} change? It currently deduces std::initializer list (may need to re-analyse?)
KS: I'm certain we dont change that. That would block the route to successful adoption of the paper.

POLL: Does SG22 want to forward P1997R1 on to EWG for further refinement?

Committee	SF	F	N	A	SA	Notes
WG14	1	2	1	0	0	Consensus
WG21	2	4	1	0	0	Consensus, author positions were SF and N

AB: Congratulations; consensus for both groups
TS: Thanks
AB: Next steps in WG14 - Looking for help to shepherd and/or champion the proposal?
authors: Yes
AB: On procedural issues I'm happy to help - email for that. For writing papers, that's more difficult time-wise, but I can review, from WG14 perspective. Volunteers for any other aspect? Closer to WG14 don't worry about wording until implementation experience is gained, so we have plenty of time. Once the nascent ideas are down then we can get early feedback on the reflector and ask for champions. Some members are very interested in array mechanics (Martin Uecker). Otherwise, we'd encourage others to Volunteer.
TS: dumps of links etc, anything is useful - we can filter through it.

Wrapup

AB: Next meeting should be first Fri in Sept
CP: 3rd Sept WG14... unless.. push back Sept 10th same time and channel.
AB: Good call, I'll schedule the next meeting and get minutes for this meeting out shortly, thanks everyone for coming.

End at 2:08 pm EST

Chat logs:

13:05:33 From Will to Everyone : I can scribe... but may be biased...|I can nominate?
13:05:50 From Will to Everyone : hahahah
13:06:01 From Will to Everyone : ok
13:19:11 From Theodoric Stier to Everyone : i do have a question if there is still time
13:19:24 From Aaron Ballman to Everyone : absolutely, you're next in the queue
13:20:38 From Thomas Köppe to Everyone : For more discussion, see C++ LWG email reflector topic "Proposal idea: where to move the "C headers"".
13:29:56 From Zhihao Yuan to Everyone : I think they mean deprecating non-static, array of known bound parameters.
13:30:30 From Aaron Ballman to Everyone : Zhihao: thanks!
13:30:32 From JeanHeyd to Everyone : void f(int x[4]) is just another word for "pointer, but with spicy annotation" in both C and C++, right?
13:30:42 From Aaron Ballman to Everyone : Correct
13:36:58 From Miguel Ojeda to Everyone : +1
13:48:26 From Aaron Ballman to Everyone : Does SG22 want to forward P1997 on to EWG for further refinement?
13:49:59 From Theodoric Stier to Everyone : i can explain the template stuff briefly
13:54:07 From Thomas Köppe to Everyone : Question: does the meaning of "auto x = {1, 2, 3}" change, which currently deduces an std::initializer_list?
13:54:09 From Krystian Stasiowski to Everyone : Precisely
13:54:44 From Krystian Stasiowski to Everyone : (^ in response to Aarons question about making arrays first class types)
13:56:02 From Theodoric Stier to Everyone : +1
13:56:13 From Krystian Stasiowski to Everyone : Thomas: We don't touch that particular copy-list-initialization case -- it remains unchanged
13:56:21 From Thomas Köppe to Everyone : Thanks!
13:56:33 From Thomas Köppe to Everyone : I can' t hear anyone, all audio is gone... but I need to run anyway.
13:56:39 From Thomas Köppe to Everyone : Was great seeing you all, till next time!
13:56:44 From Aaron Ballman to Everyone : Thanks, Thomas!
13:57:07 From Krystian Stasiowski to Everyone : I'm certain that we don't change it smile
13:57:49 From Aaron Ballman to Everyone : Does SG22 want to forward P1997 on to EWG for further refinement?
13:58:00 From Aaron Ballman to Everyone : Does SG22 want to forward P1997R1 on to EWG for further refinement?
14:01:45 From Will to Everyone : can't fine had on this client- I'm For
14:02:33 From Miguel Ojeda to Everyone : Will was worth twice! smile
14:03:43 From Krystian Stasiowski to Everyone : Thank you for you time smile
14:06:04 From Miguel Ojeda to Everyone : indeed