C and C++ Compatibility Study Group Meeting Minutes (Jun 2022)

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N3015
SG Meeting Date: 2022-06-17

Fri Jun 17, 2022 at 11:06am EST

Attendees

Aaron Ballman	WG21/WG14	chair
Hubert Tong	WG21/(14)	
Michael Wong	WG21/(14)	
Jens Gustedt	WG14	
Steve Downey	WG21	
Philipp K. Krause	WG14	
Tom Honermann	WG21	scribe
Martinho Fernandes	WG21	
Corentin Jabot	WG21	
Inbal Levi	WG21	

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

Agenda

P2071R2 (https://wg21.link/P2071R2) Named Universal Character Escapes

P2558R1 (https://wg21.link/P2558R1) Add @, $, and ` to the basic character set

P2071R2: Named Universal Character Escapes

Steve provided an introduction:

This paper was accepted for C++23.

Extends UCNs to also allow character names in addition to UCS scalar values.

Requires exact match of names from the Unicode standard; space and case sensitive.

Corentin has provided an implementation; his implementation requires ~375K of data.

Other than the data size, there are no particular implementation concerns.

Named escapes can be used in identifiers; everywhere that UCNs are currently accepted.

Since UCS scalar value typos are not obvious, use of names can help avoid errors.

Especially important when the author does not know the language for the character they are trying to encode.

The Unicode standard offers guarantees for various name matching algorithms.

C++ adopted spelling in the most strict form because that is what had the greatest consensus.

There was considerable discussion about which names from the Unicode standard to use; there are a few categories.

ISO/IEC 10646 conflates some of those categories.

Aaron: If we wanted to relax the name matching algorithm in the future, we can, yes?

Steve: Yes, relaxing the requirements would not create ambiguity or make any existing code ill-formed.

Aaron: Can these names be produced by macros?

Martinho: It is UB to form a UCN via token pasting.

Hubert: Agreed; a macro can produce a UCN, just not via token pasting.

Aaron: Corentin's Clang implementation has not landed yet; pending questions about where to record licensing.

Aaron: The LLVM board has approved use of the Unicode data.

Aaron: For C, there is concern about compiler size; compilers may be distributed in embedded system for code generation purposes.

Tom: If that becomes a significant concern, this could be an optional facility, but that would make me sad.

Aaron: That would be possible, but then obviates the portability benefits.

Corentin described how his implementation reduces the size of the data needed.

Corentin: I'm not aware of any suggestions for how to further reduce the size of the data.

Steve: The implementation techniques are not that complicated.

Tom: It is basically a trie, right?

Corentin: Yes.

Steve: The data could be a good candidate for #embed.

Tom: In chat:

With regard to ISO/IEC 10646 vs Unicode concerns:

WG2-N5168: https://www.unicode.org/wg2/docs/n5168R1-ISO10646.pdf

WG2-N5174: https://www.unicode.org/wg2/docs/n5174-namesaliases.pdf

Steve presented the grammar changes in C++ and stated they should be similar for C.

Jens: Need to watch out for similar grammar having different semantics.

Aaron: (chair hat off) I like this feature; it helps with the problem of programmers accidentally getting things wrong via, e.g., transposing characters.

Aaron: I also like that it avoids having to add a comment along with the UCS scalar value.

Martinho: There are still some concerns: some of the names in the Unicode standard are incorrect, but won't be fixed due to immutability.

Aaron: WG14 would likely want to see an implementation in a C compiler.

Tom: Corentin, does your implementation enable this in C?

Corentin: We could enable it as a conforming extension.

Tom: Agreed; '\N' is currently implementation defined.

Steve: A key question is whether this would break existing C code.

Aaron: That would only be the case for implementations that have assigned semantics to '\N'; many implementations just substitute 'N' in that case.

Aaron: The same problem would appear for any other chosen character.

Tom: That concern exists only within character and string literals.

Aaron: Real life cases would probably involve generated code.

Martinho: Perhaps, but universal escaping doesn't really work; would still have to avoid doing so for defined escapes.

Aaron: To Philipp, since you maintain the SDCC compiler, does the 375K pose a concern for you?

Philipp: SDCC is only officially supported as a cross compiler. Users sometimes want to shrink it down to fit on such a system, but that isn't supported.

Tom: To summarize: No new concerns identified, no specific WG14 concerns.

Corentin: We don't know if WG14 wants such a paper.

Aaron: Need a paper to answer that. Implementation experience would help. I can see this being adopted.

Jens: I think the concerns will be about the feature being well-motivated.

Jens: Would be good to have data regarding uptake in the field.

Tom: Would you expect use in other languages to offer compelling data?

Aaron: We have been considering such use for other features; like lambda expressions in C.

Aaron: A Tony Table would be useful.

Tom: Like an example that demonstrates DRY violations.

Steve: And a spot the error example.

Jens: May need to address why Unicode is important.

P2558R1 Add @, $, and ` to the basic character set

Steve introduced:

This is being round-tripped; this functionality was adopted in WG14 and is on track for WG21 adoption.

The method of specification is a bit different in WG21 vs WG14.

For WG14, the basic source character set is abstract and not tied to any particular encoding.

For WG21, we are pursuing a requirement to support UTF-8 source files; source characters are translated to the translation character set.

For WG21, the changes make some uses of UCNs that name the new characters outside literals ill-formed where they were not previously.

Hubert: WG14 did not add the new characters to the basic source character set, but did impose the requirement for the new characters to be encoded as a single code unit.

Hubert: Could someone in WG14 confirm that WG14 adopted all three new characters? The original paper only mentions two.

Jens: Confirmed.

Aaron: Confirmed.

Hubert: It may not be the case that C requires the values of the new characters to be encoded as a positive numeric value.

Aaron: I don't recall.

Philipp: I don't think that was discussed in WG14.

Aaron: I'll review notes.

Steve: I'm not aware of any encodings that encode characters with negative values.

Jens: The C2X wording, these characters are not described differently than other characters; that may imply they should be positive.

Aaron: What is the target for C++?

Steve: C++26; this missed the C++23 cutoff.

Aaron: Should this be pursued as a DR?

Tom: All implementations already support these characters and the edge cases aren't worth being concerned about.

Steve: We haven't adopted these characters for anything outside literals at this point.

Hubert: I think we should investigate wording to support that these characters are positively valued.

Tom: Is there any motivation to suggest that WG14 should put these characters in the basic source character set and have WG14 have the same edge case compatibility issues as WG21?

Corentin: C++ doesn't really have the same notion as basic source character set any more.

Tom: The basic character set states what is portably allowed in a character literal.

Hubert: There are several consequences to inclusion in the basic character set; another is for disallowing forming UCNs that name basic characters because those characters are significant to lexing.

Steve: And we don't want to hide them.

Hubert: The motivation for adding them to the basic character set is to enable their use in syntax.

Tom: And that same motivation would presumably exist for C if WG14 wanted to start using these new characters in syntax.

Hubert: Correct.

Wrapup

Aaron: I'll schedule the meeting for July, but it might not be for the first week given US holidays.

End at 12:01pm EST