

Relax requirements on `wchar_t` to match existing practices

| | |
|--------------------------|---|
| Document #: | P2460R0 |
| Date: | 2021-10-09 |
| Programming Language C++ | |
| Audience: | SG-22, EWG, LEWG |
| Reply-to: | Corentin Jabot < corentin.jabot@gmail.com > |

Target

C++23

Abstract

We propose to remove the constraints put on the encoding associated with `wchar_t` in the core wording.

Motivation

The standard claims that `wchar_t` should encode all characters of all wide encoding as a single code unit. This does not match existing practices, as `wchar_t` denotes UTF-16 on Windows. The [Windows Documentation](#) states:

Windows represents Unicode characters using UTF-16 encoding, in which each character is encoded as a 16-bit value. UTF-16 characters are called wide characters, to distinguish them from 8-bit ANSI characters. The Visual C++ compiler supports the built-in data type `wchar_t` for wide characters.

This is not merely an issue of MSVC being non conforming. It makes C++ unsuitable for development on a widely deployed operating system.

ISO 10646 also mentions:

NOTE – Former editions of this document included references to a two-octet BMP form called UCS-2 which would be a subset of the UTF-16 encoding form restricted to the BMP UCS scalar values. The UCS-2 form is deprecated.

Moreover, the requirement that “the values of type `wchar_t` can represent distinct codes for all members of the largest extended character set specified among the supported locales” also precludes any 2 bytes encodings (including UCS2), if (one of) the execution character set is UTF-8, as not all Unicode codepoints (21 bits) are representable in a single 2 bytes `wchar_t`.

Instead of stating Windows, and environments where UTF-8 is used are non-conforming, which is not useful to users, we propose to remove the constraint from the core wording.

However, we cannot change the wide functions, both for API/ABI reasons, because they are controlled by C, and at best, this requires complex surgery.

Instead, we move the constraints from the type of `wchar_t` to the constraints of the execution encoding, as defined by [P2314R3](#) [1].

Previous discussions can be found in this [SG-16 issue](#).

Behavior changes

This paper makes UTF-16 in wide literals well-formed. This does not affect implementations that were already accepting them [\[Compiler Explorer\]](#). This paper is therefore standardizing standard practices.

What about the library?

Still the status quo. Further work is needed there.

C compat

C has the same wording.

wide character
value representable by an object of type `wchar_t`, capable of representing any character in the current locale

C should consider adopting a similar resolution, however, the proposed change has no impact on C compatibility. (we are removing a constraint).

Previous polls

SG16 POLL: Add expanded motivation to D2460R0 and forward the paper so revised to EWG with a recommended ship vehicle of C++23.

| SF | F | N | A | SA |
|----|---|---|---|----|
| 5 | 3 | 1 | 0 | 0 |

Wording

[Editor's note: Modify [\[basic.fundamental\]](#) p8 as follow:]

Type `wchar_t` is a distinct type that has an implementation-defined signed or unsigned integer type as its underlying type. ~~The values of type `wchar_t` can represent distinct codes for all members of the largest extended character set specified among the supported locales.~~

[Editor's note: Change 16.3.3.3.5.1 [character.seq.geneal] paragraph 1. The wording below is relative to P2314R4]

The C standard library makes widespread use of characters and character sequences that follow a few uniform conventions:

- Properties specified as *locale-specific* may change during program execution by a call to `setlocale(int, const char*)` (28.5.1 [clocale.syn]), or by a change to a `locale` object, as described in 28.3 [locales] and Clause 29 [input.output].
- The *execution character set* and the *execution wide-character set* are supersets of the basic literal character set (5.3 [lex.charset]). The encodings of the execution character sets and the sets of additional elements (if any) are locale-specific. All elements of the execution wide-character set are encoded as a single code unit representable by a value of type `wchar_t`. [Note: The encoding of the execution character sets can be unrelated to any literal encoding. — end note]
- A letter is any of the 26 lowercase or 26 uppercase letters in the basic execution character set. The decimal-point character is the locale-specific (single-byte) character used by functions that convert between a (single-byte) character sequence and a value of one of the floating-point types. It is used in the character sequence to denote the beginning of a fractional part. It is represented in Clause 17 through Clause 32 and Annex D by a period, '.', which is also its value in the "C" locale.

Acknowledgment

Thanks to SG-16 for their feedback on this paper, notably Hubert Tong for mentioning that even UCS-2 does not always satisfy the core wording requirements.

References

- [1] Jens Maurer. P2314R3: Character sets and encodings. <https://wg21.link/p2314r3>, 9 2021.
- [N4885] Thomas Köppe *Working Draft, Standard for Programming Language C++* <https://wg21.link/N4885>