Title: Unicode Length Modifiers v2
Author: Marcus Johnson
Date: April 24th 2022
Document: N2983
Proposal Category: New Feature
Reference: n2912 C2x Working Draft

Revision History:
N2983:
Rebased on N2912 from N2731.
Added char8_t support.
Changed U16/U32 length modifiers to lowercase u as per 7.31.13.
Added the part about setting errno to EILSEQ.
Dropped code point stuff because mbrtoc16 and friends is defined to only operate on valid codepoints; checking in printf and friends is unnecessary.

N2875:
- Added poll about incorporating the definitions.
- Added section about how conversions from Unicode to the execution character set are to be done, and what happens when the execution character set is unable to represent a Unicode character.
- Removed mention of the Precision modifiers.
- Moved the Conversions to Execution Character Set to each function.
- Write the exact conversion procedure for characters, for strings do the same process, as many times as needed.
- Lone Surrogate is undefined behavior -> an encoding error.
- U+XXXXX -> encoding error, set errno, return immediately.

N2761: Original proposal.

Abstract:
Let's add support for char16_t and char32_t characters and strings to the formatted I/O functions.

Motivation:
I can't print char16_t or char32_t characters or strings on MacOS or Windows (see Example Program at the bottom), even when casting to the platform's wchar_t type.

Suggested Changes:
Additions are marked in green, removals in ~~red~~.

Informative change:

3.7.4: Code point
Any value in the Unicode codespace; that is, the range of integers from 0x0 to 0x10FFFF, Inclusive.

3.7.5: Code unit
The minimal bit representation that can represent a unit of encoded text for processing or interchange.

The Unicode Standard mandates at least 8-bit code units in the UTF-8 encoding form, 16-bit code units in the UTF-16 encoding form, and 32-bit code units in the UTF-32 encoding form.

The maximum number of code units per code point is as follows: 4 code units per code point for UTF-8; 2 code units per code point for UTF-16; 1 code unit per code point for UTF-32.

Does WG14 want to incorporate this informative charge into the standard separately from the normative changes in this paper?

Normative Changes:

7.21.3 §14:

An *encoding error* occurs if the character sequence presented to the underlying mbrtowc, **mbrtoc8**, **mbrtoc16**, or **mbrtoc32** functions does not form a valid (generalized) multibyte character, or if the code value passed to the underlying **wcrtomb**, **mbrtoc8, c16rtomb**, or **c32rtomb** functions does not correspond to a valid multibyte character. The wide character input/output functions and the byte input/output functions store the value of the macro EILSEQ in errno if and only if an encoding error occurs.

7.21.6.1 The fprintf function:

§7 The length modifiers and their meanings are:

u8 Specifies that a following c or s conversion specifier applies to a **char8_t** or **char8_t*** argument respectively.

u16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively.

u32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively.

§8 The conversion specifiers and their meanings are:

(c): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a character of **char8_t** type.

Conversion from UTF-8 to the execution character set shall be provided by call(s) to **c8rtomb**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the argument shall be a character of **char16_t** type.

Conversion from UTF-16 to the execution character set shall be provided by call(s) to **c16rtomb**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the argument shall be a character of **char32_t** type.

Conversion from UTF-32 to the execution character set shall be provided by call(s) to **c32rtomb**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

(s): If no ~~l~~length modifiers ~~is~~ are present

If a width is specified, no more than that many bytes are written, if a code point would require more bytes than are available, the codepoint is ignored.

If a u8 length modifier is present, the argument shall be a string of **char8_t*** type.

Conversion from UTF-8 to the execution character set shall be provided by call(s) to **c8rtomb**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the argument shall be a string of **char16_t*** type.

Conversion from UTF-16 to the execution character set shall be provided by call(s) to **c16rtomb**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the argument shall be a string of **char32_t*** type.

Conversion from UTF-32 to the execution character set shall be provided by call(s) to **c32rtomb**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

7.30.2.1 The fwprintf function:

§7 The length modifiers and their meanings are:

u8 Specifies that a following c or s conversion specifier applies to a **char8_t** or **char8_t*** argument respectively.

u16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively.

u32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively.

§8 The conversion specifiers and their meanings are:

(c): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a string of **char8_t** type.

Conversion from UTF-8 to the wide execution character set shall be provided by call(s) to **c8rtomb** followed by **mbrtowc**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the argument shall be a character of **char16_t** type.

Conversion from UTF-16 to the wide execution character set shall be provided by call(s) to **c16rtomb** followed by **mbrtowc**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the argument shall be a character of **char32_t** type.

Conversion from UTF-32 to the wide execution character set shall be provided by call(s) to **c32rtomb** followed by **mbrtowc**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

(s): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a string of **char8_t*** type.

Conversion from UTF-8 to the execution character set shall be provided by call(s) to **mbrtoc8**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the argument shall be a string of **char16_t*** type.

Conversion from UTF-16 to the wide execution character set shall be provided by call(s) to **c16rtomb** followed by **mbrtowc**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the argument shall be a string of **char32_t*** type.

Conversion from UTF-32 to the wide execution character set shall be provided by call(s) to **c32rtomb** followed by **mbrtowc**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

7.21.6.2: The fscanf function:

§11 The length modifiers and their meanings are:

u8 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char8_t.**

u16 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char16_t.**

u32 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char32_t.**

§12 The conversion specifiers and their meanings are:

(c):

P2: If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a string of **char8_t** type.

Conversion from UTF-8 to the execution character set shall be provided by call(s) to **mbrtoc8**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop printing, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the argument shall be a character of **char16_t** type.

Conversion to UTF-16 from the execution character set shall be provided by call(s) to **mbrtoc16**. if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the argument shall be a character of **char32_t** type.

Conversion to UTF-32 from the execution character set shall be provided by call(s) to **mbrtoc32**. if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

(s):

P2: If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a string of **char8_t*** type.

If a u16 length modifier is present, the argument shall be a string of **char16_t*** type.

If a u32 length modifier is present, the argument shall be a string of **char32_t*** type.

([]):
P2: If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the argument shall be a string of **char8_t*** type.

If a u16 length modifier is present, the argument shall be a string of **char16_t*** type.

If a u32 length modifier is present, the argument shall be a string of **char32_t*** type.

7.30.2.2 The fwscanf function

§11 The length modifiers and their meanings are:

u8 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char8_t.**

u16 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char16_t.**

u32 Specifies that a following c or s conversion specifier applies to an argument with type pointer to **char32_t.**

§12 The conversion specifiers and their meanings are:

(c): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the corresponding argument shall be a pointer of char8_t type.

Conversion from the wide execution character set to UTF-8 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc8**. If a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the corresponding argument shall be a pointer of char16_t type.

Conversion from the wide execution character set to UTF-16 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc16**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the corresponding argument shall be a pointer of char32_t type.

Conversion from the wide execution character set to UTF-32 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc32**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

(s): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the corresponding argument shall be a string of **char8_t*** type.

Conversion from the wide execution character set to UTF-8 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc8.** If a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

If a u16 length modifier is present, the corresponding argument shall be a string of **char16_t*** type.

Conversion from the wide execution character set to UTF-16 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc16**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

If a u32 length modifier is present, the corresponding argument shall be a string of **char32_t*** type.

Conversion from the wide execution character set to UTF-32 shall be provided by call(s) to **wcrtomb** followed by **mbrtoc32**; if a code point is not available in the execution character set, it is an encoding error, the function is to stop scanning, set errno to EILSEQ, and return immediately.

([): If no ~~l~~length modifiers ~~is~~ are present

If a u8 length modifier is present, the corresponding argument shall be a string of **char8_t*** type.

If a u16 length modifier is present, the corresponding argument shall be a string of **char16_t*** type.

If a u32 length modifier is present, the corresponding argument shall be a string of **char32_t*** type.

Example Program (Tested with Xcode 13.1 and Visual Studio 2019):

```
#include <stdint.h>
#include <stdio.h>
#include <wchar.h>
#if defined(__has_include) && __has_include(<uchar.h>)
#include <uchar.h>
#else
typedef uint_least16_t char16_t;
```

```
typedef uint_least32_t char32_t;
#endif
int main(int argc, const char *argv[]) {
#if (WCHAR_MAX <= 0xFFFF)
  char16_t *Fire = u"\U0001F525";
#elif (WCHAR_MAX <= 0xFFFFFFFF)
  char32_t *Fire = U"\U0001F525";
#endif
  printf("%ls\n", (wchar_t*) Fire);
  return 0;
}
```