Title: Unicode Length Modifiers v3
Author: Marcus Johnson
Date: June 23rd 2022
Document: N3016
Proposal Category: New Feature
References: n2912 C2x Working Draft; n2966 Character Conversions
Acknowledgements: Aaron Ballman, JeanHeyd Meneide, Tom Honermann.

Paper of Interest for obsolete standards: David Keaton requested I add this to help keep this paper from being forgotten

Revision History:
N3016:
- Removed duplicate definitions of encoding error, they’re already present in 7.21.3 §14
- Character conversion for UTF-8 incorrectly mentioned strings instead of character.
- Scanset changes removed, misinterpretation.
- Changed lowercase U back to uppercase.
- Changed printf/wprintf functions to use the new Character Conversion functions, based on N2999.
- Dropped the definitions because JeanHeyd pointed out his character conversion paper already does it.

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 mbtetc16 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.
7.21.3 §14:
An encoding error occurs if the character sequence presented to the underlying mbtowc, mbtowc8, mbtowc16, or mbtowc32 functions does not form a valid (generalized) multibyte character, or if the code value passed to the underlying wcrtomb, mbtowc8, 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 length modifiers 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 narrow execution character set shall be provided by call(s) to stdc_c8nrtomcn.

If a U16 length modifier is present, the argument shall be a character of char16_t type. Conversion from UTF-16 to the narrow execution character set shall be provided by call(s) to stdc_c16nrtomcn.

If a U32 length modifier is present, the argument shall be a character of char32_t type. Conversion from UTF-32 to the narrow execution character set shall be provided by call(s) to stdc_c32nrtomcn.

(s): If no length modifiers are present
If a width is specified, no more than that many bytes are written, if a USV 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 narrow execution character set shall be provided by call(s) to stdc_c8snrtomcsn.

If a U16 length modifier is present, the argument shall be a string of char16_t* type. Conversion from UTF-16 to the narrow execution character set shall be provided by call(s) to stdc_c16snrtomcsn.

If a U32 length modifier is present, the argument shall be a string of char32_t* type. Conversion from UTF-32 to the narrow execution character set shall be provided by call(s) to stdc_c32snrtomcsn.

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 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 wide execution character set shall be provided by call(s) to `stdc_c8nrtomwcn`.

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 `stdc_c16nrtomwcn`.

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 `stdc_c32nrtomwcn`.

(s): If no l-length modifiers 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 `stdc_c8snrtomwcsn`.

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 `stdc_c16snrtomwcsn`.

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 `stdc_c32snrtomwcsn`.

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 are present
If a U8 length modifier is present, the argument shall be a character of `char8_t` type. Conversion from the narrow execution character set to UTF-8 shall be provided by call(s) to `stdc_mcnrtoc8n`.

If a U16 length modifier is present, the argument shall be a character of `char16_t` type. Conversion from the narrow execution character set to UTF-16 shall be provided by call(s) to `stdc_mcnrtoc16n`.

If a U32 length modifier is present, the argument shall be a character of `char32_t` type. Conversion from the narrow execution character set to UTF-32 shall be provided by call(s) to `stdc_mcnrtoc32n`.
(s):

**P2:** If no `-length modifiers` are present

If a U8 length modifier is present, the argument shall be a string of `char8_t*` type. Conversion from the narrow execution character set to UTF-8 shall be provided by call(s) to `stdc_mcsnrtoc8sn`.

If a U16 length modifier is present, the argument shall be a string of `char16_t*` type. Conversion from the narrow execution character set to UTF-8 shall be provided by call(s) to `stdc_mcsnrtoc16sn`.

If a U32 length modifier is present, the argument shall be a string of `char32_t*` type. Conversion from the narrow execution character set to UTF-8 shall be provided by call(s) to `stdc_mcsnrtoc32sn`.

---

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 `-length modifiers` 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 `stdc_mwcnrtoc8sn`.

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 `stdc_mwcnrtoc16sn`.

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 `stdc_mwcnrtoc32sn`.

**(s):** If no `-length modifiers` 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 `stdc_mwcsnrtoc8sn`.

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 `stdc_mwcsnrtoc16sn`.

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 `stdc_mwcsnrtoc32sn`.

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

```c
#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;
}