Title: Unicode Length Modifiers
Author: Marcus Johnson
Date: November 30th 2021
Document: n2875, updates N2761
Proposal Category: New Feature
Reference: n2731 C2x Working Draft

Revision History:

- 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.

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.

Feedback:

Feedback about using a leading U prefix was convincing; changing the modifier from l16/l32 to U16/U32 has one major benefit: It separates out the locale unaware conversions of plain c/s and lc/ls from the new Unicode aware length modifiers with Unicode specific semantics. Making it easier to understand for programmers.

Char8_t support was brought up during the meeting, I did mention it in a draft of this revision, but noticed that char8_t hasn't officially been made part of the standard as of n2731, so I dropped it.

Security Considerations:

Width modifiers operate exclusively on code points, not code units for the applicable arguments.

Conversion to Execution Character Set:

Conversion from UTF-16 to the execution character set shall be provided by a call to **c16rtomb**; if a code point is not available in the execution character set, it shall be replaced with "U+" followed by the (uppercase) hex digits of the code point, without any leading zeros; e.g. "🔥" is replaced with "U+1F525".

Conversion from UTF-32 to the execution character set shall be provided by a call to **c32rtomb**; if a code point is not available in the execution character set, it shall be replaced with "U+" followed by the (uppercase) hex digits of the code point, without any leading zeros; e.g. "🔥" is replaced with "U+1F525".

Suggested Changes:

Additions are marked in green, removals in ~~red~~.

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.

(WG14 should vote on incorporating the definitions into the standard separately from the other changes in this proposal)

7.21.6.1 The fprintf function:

§7 The length modifiers and their meanings are:

U16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively; width modifiers operate on code points.

U32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively; width modifiers operate on code points.

§8 The conversion specifiers and their meanings are:

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

If a U16 length modifier is present, the argument shall be a character of **char16_t** type. (The allowable range for UTF-16 characters is 0x0-0xFFFF, except the Surrogate Range of 0xD800-0xDFFF, inclusive). Printing a lone Surrogate is undefined behavior.

If an U32 length modifier is present the int argument is converted to a UTF-32 encoded code point of type **char32_t**.

For conversion to the execution character set, see Conversion To Execution Character Set section of this proposal.

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

If a width is specified, no more than that many bytes are written, or code points for the **char16_t*** and **char32_t*** types.

If an U16 length modifier is present, the argument shall be a pointer to storage of **char16_t*** type.

If an U32 length modifier is present, the argument shall be a pointer to storage of **char32_t*** type.

7.21.6.2: The fscanf function:

§11 The length modifiers and their meanings are:

U16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively; width modifiers operate on code points.

U32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively; width modifiers operate on code points.

§12 The conversion specifiers and their meanings are:

(c): Matches a sequence of characters or code points if a U16 or U32 length modifier is present of exactly the number specified by the field width (1 if no field width is present in the directive).

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

If a U16 length modifier is present, the argument shall be a character of **char16_t** type. (The allowable range for UTF-16 characters is 0x0-0xFFFF, except the Surrogate Range of 0xD800-0xDFFF, inclusive) Printing a lone Surrogate is undefined behavior.

If a U32 length modifier is present, the argument shall be a character of **char32_t** type.

For conversion to the execution character set, see Conversion To Execution Character Set section of this proposal.

(s):

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

If a U16 length modifier is present, the argument shall be a character 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 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.

7.29.2.1 The fwprintf function:

§4p3: or the maximum number of wide characters or code points when a U16 or U32 length modifier is present to be written for s conversions.

§7 The length modifiers and their meanings are:

U16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively; width modifiers operate on code points.

U32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively; width modifiers operate on code points.

For conversion to the execution character set, see Conversion To Execution Character Set section of this proposal.

§8 The conversion specifiers and their meanings are:

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

If a U16 length modifier is present, the argument shall be a character of **char16_t** type. (The allowable range for UTF-16 characters is 0x0-0xFFFF, except the Surrogate Range of 0xD800-0xDFFF, inclusive) Printing a lone Surrogate is undefined behavior.

If a U32 length modifier is present, the argument shall be a character of **char32_t** type.

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

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.

7.29.2.2 The fwscanf function

§11 The length modifiers and their meanings are:

U16 Specifies that a following c or s conversion specifier applies to a **char16_t** or **char16_t*** argument respectively; width modifiers operate on code points.

U32 Specifies that a following c or s conversion specifier applies to a **char32_t** or **char32_t*** argument respectively; width modifiers operate on code points.

For conversion to the execution character set, see Conversion To Execution Character Set section of this proposal.

§12 The conversion specifiers and their meanings are:

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

If a U16 length modifier is present, the argument shall be a character of **char16_t** type. (The allowable range for UTF-16 characters is 0x0-0xFFFF, except the Surrogate Range of 0xD800-0xDFFF, inclusive) Printing a lone Surrogate is undefined behavior.

If an U32 length modifier is present, the argument is of type **char32_t**.

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

If a U16 length modifier is present, the argument shall be a pointer to storage of **char16_t*** type.

If a U32 length modifier is present, the argument shall be a pointer to storage of **char32_t*** type.

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

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.

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>)
```

```c
#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;
}
```