Proposal for C2x
WG14 N2418

Title:	Adding the u8 character prefix
Author, affiliation:	Aaron Ballman, GrammaTech
Date:	2019-09-02
Proposal category:	New features
Target audience:	C programmers using the UTF-8 character set

Abstract: C++17 adopted the u8 character literal prefix as complement to u8 string literal prefixes.
Prior art: C++.

Adding the `u8` character prefix

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2418
Date: 2019-09-02

Summary of Changes

N2418

- Added rationale as to why `unsigned char` is the correct underlying type
- Clarified that an environment macro is not necessary
- Moved specification of u8 away from the specification of wide character constants
- Added constraints to the proposed wording

N2198

- Original proposal

Introduction and Rationale

In C17, there are four encoding prefix spellings for string literals: `u8`, `u`, `U`, and `L`, but only three encoding prefixes for character literals: `u`, `U`, and `L`. C++17 adopted a feature adding the `u8` prefix for character literals to represent a UTF-8 encoding [WG21 N4267]. This is a useful feature that allows a programmer working in a narrow character set other than ASCII to obtain ASCII characters by using the `u8` prefix because the single code unit UTF-8 encodings are identical to ASCII. It is also makes character literal prefixes more consistent with string literal prefixes, and aligns the literal syntaxes of C and C++ more closely.

C++ uses `char8_t` as the underlying type for a `u8` character literal. This proposal proposes using `unsigned char` as the underlying type for a `u8` character literal because the underlying type for `char8_t` in C++ is `unsigned char`. Should C adopt a `char8_t` datatype [N2231], it would possibly pick either `unsigned char` or `uint_least8_t` as the underlying type. Using `unsigned char` for `u8` character literals now would then be forward compatible with future harmonization efforts.

All of the other prefixes (`L`, `u`, and `U`) map to an underlying type that has an environment macro specified in 6.10.8.2. UTF-8 character constants do not require an environment macro because they are defined in terms of a single UTF-8 code unit.

Proposed Wording

The wording proposed is a diff from the committee draft of ISO/IEC 9899-2017. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.4.4.4p1:

character-constant:

~~*' c-char-sequence '*~~
~~*L' c-char-sequence '*~~
~~*u' c-char-sequence '*~~

~~U' c-char-sequence '~~
encoding-prefix$_{opt}$ ' *c-char-sequence* '

encoding-prefix: one of
u8 u U L

Modify 6.4.4.4p2:

An integer character constant is a sequence of one or more multibyte characters enclosed in single-quotes, as in 'x'. A wide character constant is the same, except prefixed by the letter L, u, or U. A UTF-8 character constant is the same, except prefixed by u8. With a few exceptions detailed later, the elements of the sequence are any members of the source character set; they are mapped in an implementation-defined manner to members of the execution character set.

Add the following row to the table in 6.4.4.4p9:

u8 | unsigned char

Add 6.4.4.4p10 to the Constraints section:

10 A UTF-8 character constant shall not contain more than one character (e.g., u8'ab'). The value shall be representable with a single UTF-8 code unit.

Add 6.4.4.4p12 (after existing p10 in Semantics):

12 A UTF-8 character constant has type unsigned char. The value of a UTF-8 character constant is equal to its ISO/IEC 10646 code point value, provided that the code point value can be encoded as a single UTF-8 code unit.

Modify 6.4.5p1:

string-literal:
encoding-prefix$_{opt}$ " *s-char-sequence*$_{opt}$ "

~~*encoding-prefix:*~~
~~u8~~
~~u~~
~~U~~
~~L~~

Acknowledgements

I would like to recognize the following people for their help with this work: Tom Honermann and Robert Seacord.

References

[WG21 N4267]
Adding u8 character literals. Richard Smith. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n4267.html

[N2231]
char8_t: A type for UTF-8 characters and strings. Tom Honermann. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2231.htm