Proposal for C2x
WG14 N2198

Title: Adding the u8 character prefix
Author, affiliation: Aaron Ballman, GrammaTech
Date: 2017-12-31
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 \texttt{u8} character prefix

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2198
Date: 2017-12-31

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

One thing to note is that C and C++ have diverged in their treatment of character literals. The C standard relies on the environment macros defined in 6.10.8.2 to determine the encodings of character values in their corresponding types. The C++ standard, in [lex.ccon]p3-5 require the character literals \texttt{u8}, \texttt{u}, and \texttt{U} to correspond exactly to an ISO 10646 code point. Because of this requirement, C++ is able to specify the mapping between the single multibyte character and the execution character set wide character by deferring to another standard. C uses the multibyte APIs from the C Standard Library to perform this mapping, but I do not see any existing facilities that will suffice for UTF-8. Should I also consider adding an \texttt{mbrtoc8} function as part of this proposal, or should I use a formulation similar to how we treat UTF-8 string literals? Do I need to add an environment macro to 6.10.8.2 for UTF-8 encodings?

Proposed Wording
The wording proposed is a diff from the committee draft of ISO/IEC 9899-2017. \textcolor{green}{Green} text is new text, while \textcolor{red}{red} text is deleted text.

Modify 6.4.4.4p1:

\begin{verbatim}
character-constant:
  \texttt{c-char-sequence}\texttt{\textbackslash}c
  \texttt{u8 c-char-sequence}\texttt{\textbackslash}
  \texttt{u c-char-sequence}\texttt{\textbackslash}
  \texttt{U c-char-sequence}\texttt{\textbackslash}
  encoding-prefix\texttt{opt} \texttt{\textbackslash}c-char-sequence\texttt{\textbackslash}

encoding-prefix: one of
  \texttt{u8 u U L}
\end{verbatim}

Modify 6.4.4.4p2:

An integer character constant is a sequence of one or more multibyte characters enclosed in single-quotes, as in \texttt{\textquoteleft x\textquoteleft}. A wide character constant is the same, except prefixed by the letter \texttt{u8}, \texttt{L}, \texttt{u}, or \texttt{U}. 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 |

Modify 6.4.4.4p11:

A wide character constant prefixed by the letter \texttt{L} has type \texttt{wchar\_t}, an integer type defined in the \texttt{<stddef.h>} header; a wide character constant prefixed by the letter \texttt{u} or \texttt{U} has type \texttt{char16\_t or char32\_t}, respectively, unsigned integer types defined in the \texttt{<uchar.h>} header. A wide character constant prefixed with \texttt{u8} has type \texttt{char}. The value of a wide character constant containing a single multibyte character that maps to a single member of the extended execution character set is the wide character corresponding to that multibyte character, as defined by the \texttt{mbtowc,mbtowl6, or mbtowl632} function as appropriate for its type, with an implementation-defined current locale. The value of a wide character constant containing more than one multibyte character or a single multibyte character that maps to multiple members of the extended execution character set, or containing a multibyte character or escape sequence not represented in the extended execution character set, is implementation-defined.

Modify 6.4.5p1:

\begin{verbatim}
string-literal:
  encoding-prefix_opt ” s-char-sequence_opt ”

encoding-prefix:
  u8
  u
  U
\end{verbatim}

References

[WG21 N4267]