# Conversion to execution encoding should not lead to loss of meaning

Document #: P1854R0
Date: 2020-05-17
Project: Programming Language C++
Audience: SG-16, EWG
Reply-to: Corentin Jabot <corentin.jabot@gmail.com>

*It's just semantic! - Kevlin Henney*

## Abstract

The current wording does not guarantee that sequences of characters will be preserved by compilation, which reduces the portability and reliability of C++ source code.

## Motivation

Implementation defined behaviors related conversion to execution encoding reduce the portability of C++ programs, and lead to silently incorrect programs as implementations are allowed to modify the characters they cannot represent in the execution encoding. Strings are text which carries intent and meaning; An implementation should not be able to alter that meaning.

## Impact on the standard and implementations

This constitutes a breaking change in the wording, as well as some implementations(MSVC) and matches other existing implementations' behaviour. However, the code that would break would not be code that matches the developer intent.

## Proposed wording

Modify 5.2.5 [lex.phases] as follow

> Each basic source character set member in a character literal or a string literal, as well as each escape sequence and universal-character-name in a character literal or a non-raw string literal, is converted to the corresponding member of the execution character set ([lex.ccon], [lex.string]); if there is no corresponding

member, ~~it is converted to an implementation-defined member other than the null (wide) character~~ the program is ill-formed.

Modify 5.13.3.2 [lex.icon] as follow

A character literal that does not begin with `u8`, `u`, `U`, or `L` is an *ordinary character literal*. An ordinary character literal that contains a single *c-char* representable in the execution character set has type `char`, with value equal to the numerical value of the encoding of the *c-char* in the execution character set. An ordinary character literal that contains more than one *c-char* is a *multicharacter literal*. A multicharacter literal~~, or an ordinary character literal containing a single *c-char* not representable in the execution character set~~, is conditionally-supported, has type `int`, and has an implementation-defined value. An ordinary character literal containing a single *c-char* not representable in the execution character set is ill-formed.

Modify 5.13.3.6 [lex.icon] as follow

A character literal that begins with the letter `L`, such as `L'z'`, is a wide-character literal. A wide-character literal has type `wchar_t`. The value of a wide-character literal containing a single *c-char* has value equal to the numerical value of the encoding of the *c-char* in the execution wide-character set, unless the *c-char* has no representation in the execution wide-character set, in which case the ~~value is implementation-defined~~ program is ill-formed. [*Note:* The type `wchar_t` is able to represent all members of the execution wide-character set (see **??**). — *end note*] The value of a wide-character literal containing multiple *c-char*s is implementation-defined.

Modify 5.13.3.8 [lex.icon] as follow

The escape `\`*ooo* consists of the backslash followed by one, two, or three octal digits that are taken to specify the value of the desired character. The escape `\x`*hhh* consists of the backslash followed by `x` followed by one or more hexadecimal digits that are taken to specify the value of the desired character. There is no limit to the number of digits in a hexadecimal sequence. A sequence of octal or hexadecimal digits is terminated by the first character that is not an octal digit or a hexadecimal digit, respectively.

The value of a character literal is implementation-defined if it falls outside of the implementation-defined range defined for `char` (for character literals with no prefix) or `wchar_t` (for character literals prefixed by `L`).

[*Note:* If the value of a character literal prefixed by `u`, `u8`, or `U` is outside the range defined for its type, the program is ill-formed. — *end note*]

If the value of a character literal is outside the range defined for its type, the program is ill-formed.

## Acknowledgments

Many thanks to JeanHeyd Meneide, Peter Bindels, Zach Laine, Tom Honermann and Steve Downey who reviewed this paper and offered valuable feedback.

## References

[N4830] Richard Smith *Working Draft, Standard for Programming Language C++*
https://wg21.link/n4830