Proposal for C2x
WG14 N2563

Title:	Character encoding of diagnostic text
Author, affiliation:	Aaron Ballman
Date:	2020-09-01
Proposal category:	Change/Clarification Requests

Abstract: The standard is unclear and inconsistent regarding what character encoding should be used when issuing diagnostic text as a result of an operation with user-supplied text (`static_assert`, `#error`, etc).

Character encoding of diagnostic text

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2563
Date: 2020-09-01

Summary of Changes

N2563

- Original proposal

Introduction and Rationale

The standard provides a few mechanisms that suggest an implementation issues a diagnostic based on text written in the source code. However, the standard does not uniformly address what should happen if the execution character set of the compiler cannot represent the text in the source character set.

`[[deprecated]]`

The `[[deprecated]]` attribute specifies in its recommended practice:

> Implementations should use the deprecated attribute to produce a diagnostic message in case the program refers to a name or entity other than to declare it, after a declaration that specifies the attribute, when the reference to the name or entity is not within the context of a related deprecated entity. The diagnostic message may include text provided by the string literal within the attribute argument clause of any deprecated attribute applied to the name or entity.

`static_assert`

The `static_assert` declaration specifies, in part:

> Otherwise, the constraint is violated and the implementation shall produce a diagnostic message that includes the text of the string literal, if present, except that characters not in the basic source character set are not required to appear in the message.

`#error`

The `#error` directive specifies, in part:

> …causes the implementation to produce a diagnostic message that includes the specified sequence of preprocessing tokens.

`[[nodiscard]]`

N2448 added an optional string literal argument to the `[[nodiscard]]` attribute similar to the one for the `[[deprecated]]` attribute in that the text is recommended to appear in a diagnostic message. The proposal specifies:

> The diagnostic message may include text provided by the string literal within the attribute argument clause of any nodiscard attribute applied to the name or entity.

This proposal was adopted with the understanding that the character encoding issue would be resolved in a future paper.

Proposal

The wording for `static_assert` either captures the intent directly or is sufficiently close to use as a model. The source code is written in the source character set and the implementation executes within some execution environment with its own execution character set. According to C17 5.2.1p3, all characters that exist in the basic source character set also exist in the basic execution character set, making it safe to assume those characters can be properly represented in both environments.

In discussion with the WG21 SG16 Unicode study group chair, it was noted that the minimum restriction necessary is to restrict to the basic execution character set rather than the basic source character set. This is the same as the basic source character set except that it also adds the \a (alert), \b (backspace), \r (carriage return), and \n (new line) characters. Using the basic execution character set may ease the implementation burden for handling \r differently than \n, though as a matter of QoI, we recommend that implementations do not pass control characters directly to the output.

The current proposed wording uses the basic source character set consistently for these features because it is the least amount of change from the status quo, but the author would like the committee to explicitly decide whether to use the basic source or execution character set. A suggested straw poll wording is: does the committee wish to use the basic execution character set for user-supplied diagnostic text from source code?

Proposed Wording

The wording proposed is a diff from the committee draft of WG14 N2478 with WG14 N2448 applied. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.7.11.2p4 (added by WG14 N2448):

The diagnostic message ~~may~~should include text provided by the string literal within the attribute argument clause of any `nodiscard` attribute applied to the name or entity, except that characters not in the basic source character set may be omitted from the message.

Modify 6.7.11.4p5:

Implementations should use the `deprecated` attribute to produce a diagnostic message in case the program refers to a name or entity other than to declare it, after a declaration that specifies the attribute, when the reference to the name or entity is not within the context of a related deprecated entity. The diagnostic message ~~may~~should include text provided by the string literal within the attribute argument clause of any `deprecated` attribute applied to the name or entity, except that characters not in the basic source character set may be omitted from the message.

Modify 6.10.5p1:

A preprocessing directive of the form

`# error` *pp-tokens*$_{opt}$ *new-line*

causes the implementation to produce a diagnostic message that includes the specified sequence of preprocessing tokens, except that characters not in the basic source character set are not required to appear in the message.

Acknowledgements

I would like to recognize the following people for their help in this work: Tom Honermann, Hubert Tong, and Joseph Myers.