Document Number: P2246R1
Date: 2021-01-15
Reply-to: Aaron Ballman <aaron@aaronballman.com>
Audience: Evolution Working Group

Character encoding of diagnostic text

Summary of Changes

R1

- Changed a “shall” to “should” in the wording for `static_assert`
- Updated rationale
- Added wording to change #error from a “shall” to a “should”

R0

- Initial 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.

WG14 had a similar issue with these same constructs and resolved them in WG14 N2563, which was adopted at the Oct 2020 virtual meeting. This paper is a companion paper intended to keep C and C++ aligned.

[[deprecated]]

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

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. The diagnostic message should include the text provided 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 program is ill-formed, and the resulting diagnostic message ([intro.compliance]) shall include the text of the *string-literal*, if one is supplied, except that characters not in the basic source character set are not required to appear in the diagnostic message.

#error

The `#error` directive specifies, in part:

… causes the implementation to produce a diagnostic message that includes the specified sequence of preprocessing tokens, and renders the program ill-formed.

[[nodiscard]]

The `[[nodiscard]]` attribute specifies in its recommended practice, in part:

… The *string-literal* in a `nodiscard` *attribute-argument-clause* should be used in the message of the warning as the rationale for why the result should not be discarded.

Proposal

To understand the proposal, we have to first understand something about character sets. The source character set is the character set used to encode the input to the implementation (and is often determined by the encoding the user's editor saved the source code file in). The execution character set is the character set used by the target architecture the implementation is translating the source code for. Neither character set describes the environment in which diagnostics are displayed to the user. For instance, the user may encode their source in UTF-16, be compiling their code for a system whose execution character set is UTF-8, but the shell running the compiler may be using Windows Latin 1. There is no relationship between the character set used to display diagnostics from the implementation and any of the character sets defined by the C++ Standard. Further, there are no requirements on what producing diagnostics actually means – even if the user pipes all compiler diagnostic output to /dev/null, the implementation is still strictly conforming. This freedom allows an implementation to provide the correct behavior for their diagnostic output environment (which could be a shell, a file, a serial interface, or any number of other things the standard may or may not be able to talk about).

Because the display of diagnostic messages should be merely a matter of Quality of Implementation, the proposal is to place no character set related requirements on the diagnostic output with the understanding that implementations will do what makes the most sense for their situation when issuing diagnostics in terms of which characters need to be escaped or otherwise handled in a special way.

Proposed Wording

The wording proposed is a diff from the committee draft of WG21 N4868. Green text is new text, while ~~red~~ text is deleted text.

Modify 9.1p6:

In a *static_assert-declaration*, the *constant-expression* shall be a contextually converted constant expression of type `bool` (7.7). If the value of the expression when so converted is `true`, the declaration has no effect. Otherwise, the program is ill-formed, and the resulting diagnostic message (4.1) ~~shall~~ should include the text of the *string-literal*, if one is supplied~~, except that characters not in the basic source character set (5.3) are not required to appear in the diagnostic message~~.

Modify 15.8p1:

… causes the implementation to produce a diagnostic message that ~~includes~~ should include the specified sequence of preprocessing tokens, and renders the program ill-formed.

Acknowledgements

I would like to recognize the following people for their help in this work: Tom Honermann, Hubert Tong, Joseph Myers, Alisdair Meredith, Steve Downey, JF Bastien, Martinho Fernandes, and Erich Keane.

References

[WG14 N2563]
Character encoding of diagnostic text. Aaron Ballman. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2563.pdf