Proposal for C2x
WG14 N2673

Title:	`__has_include` for C
Author, affiliation:	Aaron Ballman, Intel
	Javier Múgica
Date:	2021-03-01
Proposal category:	New features
Target audience:	C library authors, application programmers

Abstract: Highly portable code that needs to adapt to different translation environments may need to query whether specific headers exist or not.

`__has_include` for C

Reply-to: Aaron Ballman (aaron@aaronballman.com), Javier Múgica (javier@aerotri.es)

Document No: N2673
Date: 2021-03-01

Summary of Changes

N2673

- Initial proposal

Introduction and Rationale

This paper describes the `__has_include` feature, which allows the programmer to determine at preprocessing time whether the specified header file exists. This feature was standardized in C++17 and is a commonly implemented extension in C compilers (minimally, it is supported in Clang, GCC, MSVC, EDG, and TCC (Tiny C Compiler)). The feature is intended for highly portable code such as libraries to adapt to different translation environments.

Adopting this proposal also serves to keep the preprocessor synchronized between C and C++.

This proposal was originally seen as WG14 N2101 at the Markham 2017 meeting where it was added to SD-3 for inclusion in C2x.

Proposed Wording

The wording proposed is a diff from WG14 N2596. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.10.1 to add a new Syntax section before Constraints:

> **Syntax**
>
> *defined-macro-expression:*
>   `defined` *identifier*
>   `defined (` *identifier* `)`
>
> *h-preprocessing-token:*
>   any *preprocessing-token* other than >
>
> *h-pp-tokens:*
>   *h-preprocessing-token*
>   *h-pp-tokens h-preprocessing-token*
>
> *header-name-tokens:*
>   *string-literal*
>   < *h-pp-tokens* >
>
> *has-include-expression:*
>   `__has_include (` *header-name* `)`
>   `__has_include (` *header-name-tokens* `)`

has-attribute-expression:

    `__has_c_attribute` `(` *pp-tokens* `)`

Modify 6.10.1p1:

> The expression that controls conditional inclusion shall be an integer constant expression except that: identifiers (including those lexically identical to keywords) are interpreted as described below[180]~~.~~ and it may contain zero or more defined macro expressions and/or has include expressions and/or has attribute expressions as unary operator expressions.

Modify 6.10.1p2:

> ~~It may contain unary operator expressions of the form~~
> ~~`defined` *identifier*~~
> ~~or~~
> ~~`defined (` *identifier* `)`~~
> ~~which evaluate~~ A defined macro expression evaluates to 1 if the identifier is currently defined as a macro name (that is, if it is predefined or if it has been the subject of a `#define` preprocessing directive without an intervening `#undef` directive with the same subject identifier), 0 if it is not.

Insert new paragraphs after 6.10.1p2:

> 3 The second form of the has include expression is considered only if the first form does not match, in which case the preprocessing tokens are processed just as in normal text.
>
> 4 The header or source file identified by the parenthesized preprocessing token sequence in each contained has include expression is searched for as if that preprocessing token were the pp-tokens in a `#include` directive, except that no further macro expansion is performed. Such a directive shall satisfy the syntactic requirements of a `#include` directive. The has include expression evaluates to 1 if the search for the source file succeeds, and to 0 if the search fails.

Modify the existing 6.10.1p3:

> ~~The conditional inclusion expression may contain unary operator expressions of the form~~
> ~~`__has_c_attribute (` *pp-tokens* `)`~~
> ~~which are~~ Each has attribute expression is replaced by a nonzero pp-number matching the form of an integer constant if the implementation supports an attribute with the name specified by interpreting the pp-tokens as an attribute token, and by 0 otherwise. The pp-tokens shall match the form of an attribute token.

Modify the existing 6.10.1p5:

> The `#ifdef` and `#ifndef` directives, and the `defined` conditional inclusion operator, shall treat __has_include and `__has_c_attribute` as if ~~it was~~ they were the name of ~~a~~ defined macros. The identifiers __has_include and `__has_c_attribute` shall not appear in any context not mentioned in this subclause.

Modify the existing 6.10.1p7:

> … After all replacements due to macro expansion and evaluations of ~~the `defined` and `__has_c_attribute` unary operators~~ defined macro expressions, has include expressions, and has attribute expressions have been performed, all remaining identifiers (including those lexically

identical to keywords) are replaced with the pp-number 0, and then each preprocessing token is converted into a token. …

Add an example to 6.10.1p10 (before the `__has_c_attribute` example):

EXAMPLE This demonstrates a way to include a header file only if it is available.

```
#if __has_include(<feature.h>)
#include <feature.h>
#else
#error "support for <feature.h> is unavailable"
#endif
```

Acknowledgements

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

References

[N2101]
__has_include for C. Nelson. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2101.htm