Add support for preprocessing directives `#elifdef` and `#elifndef`

Committee: ISO/IEC JTC1 SC22 WG14

Document Number: N2645
Date: 2020-01-25

Authors: Melanie Blower
Reply to: Melanie.Blower@intel.com

Contents

Introduction and Rationale

The `#ifdef` and `#ifndef` preprocessing directives exist today as shorthand for `#if defined(identifier)` and `#if !defined(identifier)`, respectively. However, no analogous shorthand preprocessing directives exist for `#elif defined(identifier)` and `#elif !defined(identifier)`. Some users have expressed surprise that these directives are not available: there are comments on the stack overflow (see https://stackoverflow.com/questions/20729032/can-we-use-elif-in-c, https://stackoverflow.com/questions/9461927/invalid-preprocessing-directive-for-elseifdef-in-xcode, https://stackoverflow.com/questions/65138617/is-there-a-c-preprocessor-which-can-replace-contiguous-else-and-ifdef-directives) and twitter (see https://twitter.com/samykamkar/status/956784258164563968) forums from newbie users discussing the absence of these preprocessing directives.

Add two new preprocessing directives `#elifdef` and `#elifndef` that exactly parallel the functionality supplied in the existing `#ifdef` and `#ifndef` directives. This improves the expressivity and predictability of the language, especially for new users.

Note that these directives do not add any new, problematic combinations of conditional inclusion directives. e.g., this code is fine:

```
#if FOO
#elifdef BAR
#else
#endif
```

by the same reasoning that this code is already fine today:

```
#ifdef BAR
#elif FOO
#else
#endif
```

Prior Art

The major C compilers do not support these directives but since it is a straightforward extension of the existing capability it is certain to be simple to implement. There is some prior art that's adjacent to C: software.hixie.ch provides a C-like preprocessor with #elifdef and #elifndef support, pikt.org provides tools for Linux administration and includes a file preprocessor that supports #elifdef and #elifndef. There are other tools such as lypp, a Lex Yacc preprocessor that supports %elifdef and %elifndef, and gpp, a generic preprocessor.

https://software.hixie.ch/utilities/unix/preprocessor/
http://pikt.org/pikt/ref/ref.3.ifdef_endifdef_define_setdef.html
https://github.com/trixirt/lypp
https://docs.rs/gpp/0.6.0/gpp/

Note that #ifdef has been present since C89 and I couldn't find any papers in the WG14 log that seemed to touch on this topic.

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 Preprocessing Directives Syntax p 1. Extend the definition of #elif-group adding 2 alternatives:

> # **elifdef** *identifier new-line group$_{opt}$*

> # **elifndef** *identifier new-line group$_{opt}$*

Modify 6.10.1 Conditional Inclusion, Constraints p 5 adding elifdef and elifndef:

> The #ifdef, and #ifndef, #elifdef, and #elifndef directives, and the defined conditional inclusion operator, shall treat __has_c_attribute as if it was the name of a defined macro.

Modify 6.10.1 Conditional Inclusion, Semantics p 8
Preprocessing directives of the forms

> # **ifdef** *identifier new-line group$_{opt}$*

> # **ifndef** *identifier new-line group$_{opt}$*

> # **elifdef** *identifier new-line group$_{opt}$*

> # **elifndef** *identifier new-line group$_{opt}$*

check whether the identifier is or is not currently defined as a macro name. Their conditions are equivalent to #if defined identifier, ~~and~~ #if !defined identifier, #elif defined identifier, and #elif !defined identifier respectively.

Modify 6.10.1 adding a 2nd example.

```
#ifdef __STDC__
#define TITLE "ISO C Compilation"
#elifndef __cplusplus
#define TITLE "Non-ISO C Compilation"
#else   /* C++ */
#define TITLE "C++ Compilation"
#endif
```

Modify A.3 Preprocessing directives adding 2 alternatives to the 6.10 elif-group
        # **elifdef** *identifier new-line group*$_{opt}$

        # **elifndef** *identifier new-line group*$_{opt}$

Modify J.6.2 Particular identifiers or keywords, adding 2 identifiers to this list: elifdef and elifndef.