SC22/WG14/N1330

July 10, 2008

Reply to: Thomas Plum, tplum@plumhall.com

Static Assertions

This paper defines a specification for static assertions which would achieve maximal compatibility with the C++ definition. It is a word-for-word adaptation of SC22/WG21/N1720, by Robert Klarer (klarer@ca.ibm.com), Dr. John Maddock (john@johnmaddock.co.uk), Beman Dawes(bdawes@acm.org), and Howard Hinnant(hinnant@twcny.rr.com). [Incorporates feedback from WG14.11495 Myers]

A static_assert-declaration takes the following form:

static_assert (constant-expression , string-literal) ;

If the constant-expression in the static assertion evaluates as 0, the compiler will issue a diagnostic message containing the literal. Otherwise, the static_assert-declaration has no effect.

The static_assert-declaration does not declare a new type or object, and does not imply any size or time cost at runtime.

Proposed Wording

To clause 6.4.1, Keywords, add the new keyword "_Static_assert"

To clause 6.7, Declarations, add an additional entry to *declaration*:

static_assert-declaration

To clause 6.7, Declarations, add an additional grammar element:

static_assert-declaration:
    `_Static_assert (` *constant-expression* `,` *string-literal* `) ;`

To clause 6.7.2.1, Structure and union specifiers, add an additional entry to *struct-declaration:*:

static_assert-declaration

To clause 6.7, at a location to be determined by the Project Editor, add a new section or sub-section:

> **static_assert declaration**
>
> The *constant-expression* shall be an integer constant expression (6.6). If the value of the expression compares unequal to 0, the declaration has no effect. Otherwise, the program is ill-formed, causing the implementation to produce a diagnostic message (3.10) that includes the text of the *string-literal*, except that characters not in the basic source character set (5.2.1) are not required to appear in the diagnostic message.

[Note to editor: The footnote (99 in N1256) listing uses of integer constant expressions should also be updated to list this new use. That footnote is also missing array designators in initializers.]

> To clause 6.7, Declarations, add an additional entry to *declaration*:
>
> *static_assert-declaration*
>
> **7.x Compatibility <*to-be-determined*.h>**
> The header <*to-be-determined*.h> defines NN macros, which are provided for compatibility with other ISO/IEC languages and systems.
> The macro **`static_assert`** expands to **`_Static_assert`**.
> Notwithstanding the provisions of 7.1.3, a program may undefine and perhaps then redefine the macros **`static_assert`** [etc].

[Note: the name *to-be-determined* could become **std_static_assert** by analogy with **bool.** Alternatively, WG14 might consider a more generic approach, such as **stdcompatibility**, in which future “compatibility” macros might be housed.]

Examples

Static assertions at file scope

```
// At file scope, the static_assertion declaration
// may be used as an alternative to the #error preprocessor
// directive.
//
static_assert(sizeof(long) >= 8,
        "64-bit code generation not enabled/supported.");
```

Static assertions at block scope

```
#include <sys/param.h> // for PAGESIZE
int do_something()
{
        struct VMPage {
                // ...
        };
        static_assert(sizeof(VMPage) == PAGESIZE,
               "Struct VMPage must be the same size "
               "as a system virtual memory page.");

                // ...
}
```

Interactions

This proposal does not affect or alter any existing language feature. Legacy code is affected only by the well-known issue of introducing a new keyword in the implementer’s name space and a new macro in a new header.

Prior Art

Static assertions have been used in the Boost libraries since around the year 2000. This proposal is based upon that experience.

Implementations exist in several released compilers, including gcc 4.3, Comeau 4.3.9, and in EDG front-ends since version 3.9 (March 2007).

Implementation of this feature requires modification of the compiler front end only; no changes to the backend, runtime library, linker, or debugger are necessary.