Proposal for C2x
WG14 N2410

Title:	The noreturn attribute
Author, affiliation:	Aaron Ballman, GrammaTech
Date:	2019-08-15
Proposal category:	New features
Target audience:	General developers, compiler/tooling developers

Abstract: C11 added support for the `_Noreturn` function type specifier whereas C++11 added support for the `[[noreturn]]` attribute. This proposal attempts to bring the two features into cross-language alignment.

Prior art: C++

The `noreturn` attribute

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2410
Date: 2019-08-15

Summary of Changes

N2410

- Original proposal.

Introduction

Some functions are defined to never return back to the caller. For instance, functions that terminate the program by calling `abort()` or `_Exit()`, or functions that call `longjmp()` to resume execution elsewhere. There are advantages to program readability and performance when the user is able to mark these functions as not returning. To support this case, C++11 adopted an attribute spelled `[[noreturn]]` in 2008 [WG21 N2761], while C11 adopted a function specifier spelled `_Noreturn` to do the same in 2010 [N1478].

Rationale

Now that C2x has support for attributes using the same syntax as C++ [N2269], the `_Noreturn` function specifier may be a point of confusion for users, especially ones who write function declarations in a header file shared included in both a C and C++ translation unit. The `_Noreturn` keyword will not be known to a C++ implementation and the `[[noreturn]]` attribute will not be known to a C implementation, despite the semantics of the concept being the same between languages.

Proposal

This paper proposes adding a new attribute, spelled `[[noreturn]]` and deprecates use of `_Noreturn`. To ease transition to a new language standard, this paper also proposes a migration path forward for users who are using the old `_Noreturn` or `noreturn` spellings.

[[noreturn]]

This paper proposes adding a new attribute spelled `[[noreturn]]`. The `[[noreturn]]` attribute is used to specify that a function does not return execution to its caller. It has the same semantics as the current `_Noreturn` function specifier in that it is undefined behavior to return from a function marked `[[noreturn]]`, with the recommendation that implementations diagnose such a situation.

_Noreturn

The `_Noreturn` function specifier keyword is replaced by a predefined macro of the same spelling. The predefined macro will expand to `[[noreturn]]`, allowing code previously written with `_Noreturn` to continue to work. Despite this predefined macro being a new addition for C2x, the macro is deprecated in order to encourage users to use the more language-agnostic `[[noreturn]]` spelling directly.

noreturn

The `noreturn` macro continues to expand to `_Noreturn`, which in turn will be expanded to `[[noreturn]]`, allowing code previously written with `noreturn` to continue to work. The `<stdnoreturn.h>` header file is deprecated in order to encourage users to use the more language-agnostic `[[noreturn]]` spelling directly. Note that having `noreturn` expand to `[[noreturn]]` was considered, but has a usability issue where including `<stdnoreturn.h>` means that a subsequent use of `[[noreturn]]` will be macro expanded to `[[[[noreturn]]]]` and be a constraint violation.

Standard library interfaces

The standard library interfaces that currently use the `_Noreturn` keyword will instead use the `[[noreturn]]` attribute.

Proposed Wording

The wording proposed is a diff from N2385. Green text is new text, while ~~red~~ text is deleted text.

Modify 6.4.1p1 (the table of keywords) to remove the `_Noreturn` keyword.

Modify 6.7.4p1:

1 *function-specifier:*

```
        inline
```
~~`_Noreturn`~~

Delete 6.7.4p8, p9, and p12:

~~8 A function declared with a `_Noreturn` function specifier shall not return to its caller.~~

~~9 The implementation should produce a diagnostic message for a function declared with a `_Noreturn` function specifier that appears to be capable of returning to its caller.~~

~~12 EXAMPLE 2~~

```
        _Noreturn void f () {
                abort(); // ok
        }
        _Noreturn void g (int i) { // causes undefined behavior if i <= 0
                if (i > 0) abort();
        }
```

Add a new subclause after 6.7.11.2:
Drafting note: I would like the UB in paragraph 2 to be the C equivalent of C++'s "ill-formed, no diagnostic required" so that it's clear that this isn't just your garden variety UB. The example was extended from the previous example to clarify that "appears to be capable" also applies to non-`void` return types.

6.7.11.3 The `noreturn` attribute

Constraints

1 The `noreturn` attribute shall be applied to the identifier in a function declarator. It shall appear at most once in each attribute list and no attribute argument clause shall be present.

Semantics

2 The first declaration of a function shall specify the `noreturn` attribute if any declaration of that function specifies the `noreturn` attribute. If a function is declared with the `noreturn` attribute in one translation unit and the same function is declared without the `noreturn` attribute in another translation unit, the behavior is undefined.

3 If a function `f` is called where `f` was previously declared with the `noreturn` attribute and `f` eventually returns, the behavior is undefined.

Recommended Practice

4 The implementation should produce a diagnostic message for a function declared with a `noreturn` attribute that appears to be capable of returning to its caller.

5 EXAMPLE 1

```
[[noreturn]] void f(void) {
      abort(); // ok
}
[[noreturn]] void g(int i) { // causes undefined behavior if i <= 0
      if (i > 0) abort();
}
[[noreturn]] int h(void);
```

Implementations are encouraged to diagnose the definition of `g()` because it is capable of returning to its caller. Implementations are similarly encouraged to diagnose the declaration of `h()` because it appears capable of returning to its caller due to the non-`void` return type.

Modify 6.10.8.1p1:

1 The following macro names shall be defined by the implementation:

`_Noreturn` Expands to `[[noreturn]]`. The `_Noreturn` macro is an obsolescent feature.

`__DATE__` The date of translation of the preprocessing translation unit: a character string literal of the form "Mmm dd yyyy", where the names of the months are the same as those generated by the `asctime` function, and the first character of `dd` is a space character if the value is less than 10. If the date of translation is not available, an implementation-defined valid date shall be supplied.

`__FILE__` The presumed name of the current source file (a character string literal).179)

`__LINE__` The presumed line number (within the current source file) of the current source line (an integer constant).179)

`__STDC__` The integer constant 1, intended to indicate a conforming implementation.

`__STDC_HOSTED__` The integer constant 1 if the implementation is a hosted implementation or the integer constant 0 if it is not.

`__STDC_VERSION__` The integer constant 201710L.180)

`__TIME__` The time of translation of the preprocessing translation unit: a character string literal of the form "hh:mm:ss" as in the time generated by the `asctime` function. If the time of translation is not available, an implementation-defined valid time shall be supplied.

Modify 7.13.2.1p1:

```
1 #include <setjmp.h>
  ~~_Noreturn~~[[noreturn]] void longjmp(jmp_buf env, int val);
```

Modify 7.22.4.1p1:

```
1 #include <stdlib.h>
  ~~_Noreturn~~[[noreturn]] void abort(void);
```

Modify 7.22.4.4p1:

```
1 #include <stdlib.h>
  ~~_Noreturn~~[[noreturn]] void exit(int status);
```

Modify 7.22.4.5p1:

```
1 #include <stdlib.h>
  ~~_Noreturn~~[[noreturn]] void _Exit(int status);
```

Modify 7.22.4.7p1:

```
1 #include <stdlib.h>
  ~~_Noreturn~~[[noreturn]] void quick_exit(int status);
```

Modify 7.23 (the subclause title):

7.23 ~~_Noreturn~~[[noreturn]] <stdnoreturn.h>

Add a new paragraph 7.23p2:

2 The `noreturn` macro and the `<stdnoreturn.h>` header are obsolescent features.

Modify 7.26.5.5p1:

```
1 #include <threads.h>
  ~~_Noreturn~~[[noreturn]] void thrd_exit(int res);
```

Acknowledgements

I would like to acknowledge the following people for their contributions to this work: myself.

References

[N1478]
Supporting the ‘noreturn’ property in C1x. David Svoboda. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1478.htm

[N2269]
Attributes in C. Aaron Ballman. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2269.pdf

[WG21 N2761]
Towards support for attributes in C++. Jens Maurer, Michael Wong. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2008/n2761.pdf