Title: The noreturn attribute
Author, affiliation: Aaron Ballman, Intel
Date: 2021-03-29
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: N2700
Date: 2021-03-29

Summary of Changes

N2700
• Proposes deprecating _Noreturn and noreturn when adding [[[noreturn]].

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 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. N2410 attempted to specify _Noreturn to be a predefined macro that expands to [[[noreturn]]] and change the definition of the noreturn macro <stdnoreturn.h> to do the same. However, because _Noreturn is a function specifier, we cannot migrate user code in this way because the potential exists to break code. Consider the following conforming C17 declaration:

```c
void _Noreturn func(void);
```

If _Noreturn (or the noreturn macro from <stdnoreturn.h>) were to expand to [[[noreturn]]] when written in this position, the declaration would become invalid because [[[noreturn]]] applies to function declarations but is written as applying to the void return type instead.

[[[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
_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 being deprecated by this proposal.

_noreturn_ and _stdnoreturn.h_

The _noreturn_ macro and _stdnoreturn.h_ headers are being deprecated by this proposal. The _noreturn_ macro continues to expand to _Noreturn._

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 N2596. *Green* text is new text, while *red* text is deleted text.

Modify 6.7.4p8: *Drafting note: the cross reference is to let the reader know there’s a replacement already available.*

A function declared with a _Noreturn function specifier shall not return to its caller. The _Noreturn function specifier is an obsolescent feature (6.7.11.6).

Delete 6.7.4p12:

**EXAMPLE 2**

```c
_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 6.7.11.5:

*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. Also, this drafting takes N2557 into account and does not add a restriction on appearing multiple times in an attribute list.*

6.7.11.6 The _noreturn attribute_

**Constraints**

1 The _noreturn_ attribute shall be applied to the identifier in a function declaration. 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

```c
[[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 7.13.2.1p1:

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

Modify 7.22.4.1p1:

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

Modify 7.22.4.4p1:

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

Modify 7.22.4.5p1:

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

Modify 7.22.4.7p1:

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

Add a new paragraph 7.23p2:

2 The noreturn macro and the <stdnoreturn.h> header are obsolescent features.
Modify 7.26.5.5p1:

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

References

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

[N2269]

[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