The deprecated attribute

Reply-to: Aaron Ballman (aaron@aaronballman.com; aballman@cert.org)
Document No: N2050
Date: 2016-04-13

Introduction

Sometimes an API no longer meets the developer's needs and should be retired from use. However, there is no standard mechanism by which an API designer can communicate this to the consumer of their API. Lacking such machinery, the producer of the code must rely on unreliable sources such as documentation to convey the information to the consumer of the API. One such example is the C Standard Library function `gets()`, which was deprecated in C99 and eventually removed in C11. However, this need is not limited to just the Standard Library, it is applicable to the producer of any library code, including in-house libraries.

Proposal

This document proposes the `[[deprecated]]` attribute as a way for a programmer to specify that an entity is discouraged from being used. This gives the developer a mechanism by which they can alert the consumer of their code to pending breaking changes, and can optionally provide the consumer with additional information such as alternatives superseding the deprecated functionality.

The `[[deprecated]]` attribute can be applied to the declaration of `struct`, `union`, `enum`, *typedef-name*, variable, non-static data member, function, or enumerator. It can optionally accept a string literal argument that an implementation is encouraged to display when the deprecated entity is used. For instance, if such an attribute were present in C99, the `gets()` function could have been declared:

```
[[deprecated("Consider using fgets() instead")]] char *gets(char *);
// Alternatively:
[[deprecated]] char *gets(char *);
```

Rationale

The `[[deprecated]]` attribute has considerable real-world use, being implemented by Clang and GCC as vendor-specific extensions under the name `__attribute__((deprecated))`, by Microsoft Visual Studio under the name `__declspec(deprecated)`, and was standardized under the name `[[deprecated]]` by WG21.

Unlike other proposed attributes like `[[nodiscard]]` and `[[maybe_unused]]`, what constitutes a "use" can be more concretely determined to be any naming of a deprecated entity other than in the entity's declaration (or redeclaration). For instance, this encourages an implementation to diagnose a deprecated variable that is named in a `sizeof` expression despite the operand being unevaluated. However, because the semantics of the attribute are informative rather than normative, the notion of what constitutes a "use" is still a matter of QoI.

Proposed Wording

This proposed wording currently uses placeholder terms of art and is expected to change as N2049 progresses. It assumes a new subclause, 6.7.11, Attributes that describes the referenced grammar terms. The [Note] in paragraph 1 of the semantics is intended to convey informative guidance rather than normative requirements.

6.7.11.1 Deprecated attribute

Syntax

deprecated-attr:
deprecated deprecated-argument$_{opt}$

deprecated-argument:
(string-literal)

Constraints

1 The *attribute-token* `deprecated` can be used to mark names and entities whose use is still allowed, but is discouraged for some reason. [Footnote: in particular, `deprecated` is appropriate for names and entities that are deemed obsolete or unsafe.] It shall appear at most once in each *attribute-list*.

2 The attribute shall be applied to the declaration of a struct, a union, a *typedef-name*, a variable, a non-static data member, a function, an enumeration, or an enumerator.

Semantics

1 [Note: Implementations may use the `deprecated` attribute to produce a diagnostic message in case the program refers to a name or entity other than to declare it, after a declaration that specifies the attribute. The diagnostic message may include text provided within the *deprecated-argument* of any `deprecated` attribute applied to the name or entity. -- end note]

2 A name or entity declared without the `deprecated` attribute can later be redeclared with the attribute and vice versa. An entity is considered marked after the first declaration that marks it.

3 EXAMPLE

```
[[deprecated]] void f(void);

void g(void) {
  f();
}
```

Implementations are encouraged to warn that `f` is deprecated at the function call expression within `g`.

Acknowledgements

I would like to recognize the following people for their help in this work: David Keaton and David Svoboda. I would also like to thank the US Department of Homeland Security, without whose funding this proposal would not have been made.

References

[N0249]
Attributes in C. Aaron Ballman. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2049.pdf

[N3394]
[[deprecated]] attribute. Alberto Ganesh Barbati. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3394.html