Generalized Attribute Support

A new syntax for annotating code with attributes affords the portable provision of supplementary information for compiler implementations and external tools.

Description

Developers are often aware of information that cannot be easily deduced directly from the source code within a given **translation unit**. Some of this information might be useful to certain compilers, say, to inform diagnostics or optimizations; typical attributes, however, are designed to avoid affecting the semantics of a well-written program. By *semantics*, here we typically mean any observable behavior apart from runtime performance. Generally, ignoring an attribute is a valid and safe choice for a compiler to make. Sometimes, however, an attribute will not affect the behavior of a *correct* program but might affect the behavior of a **well-formed** yet incorrect one (see *Use Cases — Stating explicit assumptions in code to achieve better optimizations* on page 16). Customized annotations targeted at external tools might be beneficial as well.

C++ attribute syntax

C++ supports a standard syntax for attributes, introduced via a matching pair of `[[` and `]]`, the simplest of which is a single attribute represented using a simple identifier, e.g., `attribute_name`:

```
[[attribute_name]]
```

A single annotation can consist of zero or more attributes:

```
[[]]          // permitted in every position where any attribute is allowed
[[foo, bar]]  // equivalent to [[foo]] [[bar]]
```

An attribute might have an **argument** list consisting of an arbitrary sequence of tokens:

```
[[attribute_name()]]           // zero-argument attribute
[[deprecated("bad API")]]      // single-argument attribute
[[theoretical(1, "two", 3.0)]] // multiple-argument attribute
[[complicated({1, 2, 3} + 5)]] // arbitrary tokens[1]
```

Note that having an incorrect number of arguments or an incompatible argument type is a compile-time error for all attributes defined by the Standard; the behavior for all other attributes, however, is **implementation defined** (see *Potential Pitfalls — Unrecognized attributes have implementation-defined behavior* on page 18).

[1] GCC offered no support for certain tokens in the attributes until GCC 9.3 (c. 2020).