Document: P2156R0
Date: April 15, 2020
Project: Programming Language C++
SG17: Evolution Incubator
EWG: Evolution Working Group
CWG: Core Working Group
Reply-to: Erich Keane <erich.keane@intel.com>

P2156R0: Allow Duplicate Attributes

1 Motivation

CWG1914 [CWG1914] brings up the issue that our limitations on duplicate attributes are inconsistent:

> The standard attributes noreturn, carries_dependency, and deprecated all specify that they cannot appear more than once in an attribute-list, but there is no such prohibition if they appear in separate attribute-specifiers within a single attribute-specifier-seq. Since intuitively these cases are equivalent, they should be treated the same, accepting duplicates in both or neither.

EWG discussed this issue in the teleconference occurring on April 15, 2020. During this discussion, it was brought up that the duplication across attribute-specifiers are to support cases where macros are used to conditionally add attributes to an attribute-specifier-seq, however it is rare for macros to be used to generate attributes within the same attribute-list. Thus, removing the limitation for that reason is unnecessary.

However, that differentiation is unnecessary and uncompelled. The benefits derived from the limitation are minimal at the expense of bifurcating the definition. The limitation on duplicates does not provide simplified standardization for the attributes, as duplication allowed across attribute-specifiers requires a defined behavior for these cases. While the limitation does force the programmer to limit some redundant information, this does not seem to be sufficient motivation to require the difference in behavior. Additionally there is a standardization overhead to this, as each attribute must separately specify that duplications are disallowed.

This paper proposes to remove the limitation on duplicate attributes in an attribute-list for all attributes that specify it. The author believes this should be done as a Defect Report against the C++ Standard, however believes the benefits are equivilent if only applied to the current standard.

2 Wording

The following difference markers are in reference to N4849[N4849]:

[dcl.attr.depend] p1

> The attribute-token carries_-dependency specifies dependency propagation into and out of functions. ~~It shall appear at most once in each attribute-list and n~~No attribute-argument-clause shall be present. The attribute may be applied to the declarator-id of a parameter-declaration in a function declaration or lambda, in which case it specifies that the initialization of the parameter carries a dependency to each lvalue-to-rvalue conversion of that object. The attribute may also be applied to the declarator-id of a function declaration, in which case it specifies that the return value, if any, carries a dependency to the evaluation of the function call expression.

[dcl.attr.deprecated] p1

> The attribute-token deprecated can be used to mark names and entities whose use is still allowed, but is discouraged for some reason. [Note: In particular, deprecated is appropriate for names and entities that are deemed obsolescent or unsafe. — end note] ~~It shall appear at most once in each attribute-list.~~ An attribute-argument-clause may be present and, if present, it shall have the form:

[dcl.attr.fallthrough] p1

> The attribute-token fallthrough may be applied to a null statement; such a statement is a fallthrough statement. ~~The attribute-token fallthrough shall appear at most once in each attribute-list and n~~No attribute-argument-clause shall be present. A fallthrough statement may only appear within an enclosing switch statement. The next statement that would

be executed after a fallthrough statement shall be a labeled statement whose label is a case label or default label for the same switch statement and, if the fallthrough statement is contained in an iteration statement, the next statement shall be part of the same execution of the substatement of the innermost enclosing iteration statement. The program is ill-formed if there is no such statement.

[dcl.attr.likelihood] p1

The attribute-tokens likely and unlikely may be applied to labels or statements. ~~The attribute-tokens likely and unlikely shall appear at most once in each attribute-list and n~~No attribute-argument-clause shall be present. The attribute-token likely shall not appear in an attribute-specifier-seq that contains the attribute-token unlikely.

[dcl.attr.unused] p1

The attribute-token maybe_-unused indicates that a name or entity is possibly intentionally unused. ~~It shall appear at most once in each attribute-list and n~~No attribute-argument-clause shall be present.

[dcl.attr.nodiscard] p1

The attribute-token nodiscard may be applied to the declarator-id in a function declaration or to the declaration of a class or enumeration. ~~It shall appear at most once in each attribute-list.~~ An attribute-argument-clause may be present and, if present, shall have the form:

[dcl.attr.noreturn] p1

The attribute-token noreturn specifies that a function does not return. ~~It shall appear at most once in each attribute-list and n~~No attribute-argument-clause shall be present. The attribute may be applied to the declarator-id in a function declaration. 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 program is ill-formed, no diagnostic required.

[dcl.attr.nouniqueaddr] p1

The attribute-token no_-unique_-address specifies that a non-static data member is a potentially-overlapping subobject ([intro.object]). ~~It shall appear at most once in each attribute-list and no attribute-argument-clause shall be present.~~ The attribute may appertain to a non-static data member other than a bit-field.

3 References

[CWG1914] Duplicate standard attributes
http://www.open-std.org/jtc1/sc22/wg21/docs/cwg_closed.html#1914

[N4849] Working Draft, standard for Programming Language C++
urlhttp://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/n4849.pdf