Proposal for C2x
WG14 N2527

Title: Minor attribute wording cleanups
Author, affiliation: Aaron Ballman, GrammaTech
Date: 2020-05-11
Proposal category: Editorial & Bug Fixes
Target audience: People reading the standard

Abstract: Proposes minor editorial changes and small bug fixes to the attribute wording, based on feedback from the community.
Minor attribute wording cleanups

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N2527
Revises Document No: N2482
Date: 2020-05-11

Summary of Changes

N2527
- Clarify that attributes have name spaces.
- Correct the grammar for for loops to allow attributes in the clause-1 position.

N2482
- Original proposal.

Introduction

Now that attributes have been added to the working draft, implementers are starting to update their implementations accordingly. Through this process, some issues have been identified where the wording in the standard could be clarified. This proposal is an omnibus paper to clarify the wording in the standard without modifying the intent.

Proposed Wording

The wording proposed is a diff from WG14 N2478. Green text is new text, while red text is deleted text.

Clarify that attributes have their own name spaces

Modify 6.2.3p1:

If more than one declaration of a particular identifier is visible at any point in a translation unit, the syntactic context disambiguates uses that refer to different entities. Thus, there are separate name spaces for various categories of identifiers, as follows:

- label names (disambiguated by the syntax of the label declaration and use);
- the tags of structures, unions, and enumerations (disambiguated by following any\textsuperscript{24} of the keywords struct, union, or enum);
- the members of structures or unions; each structure or union has a separate name space for its members (disambiguated by the type of the expression used to access the member via the . or -> operator);
- standard attributes and attribute prefixes (disambiguated by the syntax of the attribute specifier and name of the attribute token) (6.7.11);
- the trailing identifier in an attribute prefixed token; each attribute prefix has a separate name space for the implementation-defined attributes that it introduces (disambiguated by the attribute prefix and the trailing identifier token);
- all other identifiers, called ordinary identifiers (declared in ordinary declarators or as enumeration constants).
Clarify what an attribute appertains to in a declaration
Modify 6.7.6p5:
If, in the declaration "T D1", D1 has the form

\[ \text{identifier attribute-specifier-sequence}_{\text{opt}} \]

then the type specified for ident is T and the optional attribute specifier sequence appertains to D1 the entity that is declared.

Allow `[[nodiscard]]` to be applied to a function regardless of the syntactic form of its declaration
Modify 6.7.11.2p1:
The `nodiscard` attribute shall be applied to the identifier in a function declarator declaration or to the definition of a structure, union, or enumeration type. It shall appear at most once in each attribute list and no attribute argument clause shall be present.

Disallow declarations following a `[[fallthrough]]` attribute
Modify 6.7.11.5p1:
The attribute token `fallthrough` shall only appear in an attribute declaration (6.7); such a declaration is a `fallthrough declaration`. The attribute token `fallthrough` shall appear at most once in each attribute list and no attribute argument clause shall be present. A fallthrough declaration may only appear within an enclosing `switch` statement (6.8.4.2). The next statement block item (6.8.2) that would be executed encountered after a fallthrough declaration shall be a labeled statement whose label is a case label or default label for the same `switch` statement.

Allow attributes on an expression in the `clause-1` position of a `for` loop
Modify 6.8.5p1:

\[ \text{while ( expression ) statement} \]
\[ \text{do statement while ( expression ) ;} \]
\[ \text{for ( expression-statement}_{\text{opt}} ; expression_{\text{opt}} ; expression_{\text{opt}} ) \text{ statement} \]
\[ \text{for ( declaration expression}_{\text{opt}} ; expression_{\text{opt}} ) \text{ statement} \]

Modify 6.8.5.3p1:
The statement

\[ \text{for (clause-1; expression-2; expression-3) statement} \]

behaves as follows: The expression `expression-2` is the controlling expression that is evaluated before each execution of the loop body. The expression `expression-3` is evaluated as a void expression after each execution of the loop body. If `clause-1` is a declaration, the scope of any identifiers it declares is the remainder of the declaration and the entire loop, including the other two expressions; it is reached in the order of execution before the first evaluation of the controlling expression. If `clause-1` is an
expression statement, its expression it is evaluated as a void expression before the first evaluation of the controlling expression.\textsuperscript{168)}

Acknowledgements

I would like to acknowledge the following people for their contributions to this work: Joseph Myers and Jens Gustedt.