N3033=10-0023
2010-2-5
Daveed Vandevoorde (daveed@edg.com)

Core issue 951: Various Attribute Issues

Notes

The write-up of 951 suggests allowing attributes in "a *for-init-statement* that is an *expression-statement*" or preceding various *compound-statements*: I did not follow that suggestion since there is no existing parallel for it. I also did not follow the suggestion to allow attributes preceding a *type-specifier-seq* in a *type-id*, since in other contexts prefix attributes appertain to the *declarator-id* entity, and there is no *declarator-id* in this case.

A trailing optional *attribute-specifier* has been folded into *decl-specifier-seq* and *type-specifier-seq* to simplify the overall wording (and avoiding oversights in the future).

In a few places the location of the optional *attribute-specifier* has been moved to make it consistent with similar uses elsewhere.

The changes are against N3000. In some places the changes overlap with changes for core issues 743/950 and 962 (the latter is in ready status).

Wording Changes

In 5.3.4 [expr.new] paragraph 1 amend the grammar rule for *noptr-new-declarator* as follows (to match its *noptr-abstract-declarator* counterpart):

noptr-new-declarator:
  [*expression*] *attribute-specifier*$_{opt}$
  noptr-new-declarator [*constant-expression*] *attribute-specifier*$_{opt}$

In 5.3.4 [expr.new] paragraph 5, append the following sentence:

... The *attribute-specifier* in *noptr-new-declarator* appertains to the associated array type.

In 6.4 [stmt.select] paragraph 1 amend the grammar rule for *condition* as follows:

condition:
  expression
  attribute-specifier$_{opt}$ *type-specifier-seq* ~~*attribute-specifier*$_{opt}$~~
    declarator = *initializer-clause*
  attribute-specifier$_{opt}$ *type-specifier-seq* ~~*attribute-specifier*$_{opt}$~~
    declarator braced-init-list

In 6.5 [stmt.iter] paragraph 5 amend the grammar and text as follows:

...
for-range-declaration:

expression
attribute-specifier$_{opt}$ *type-specifier-seq* ~~*attribute-specifier*$_{opt}$~~ *declarator*

See 8.3 [dcl.meaning] for the optional *attribute-specifier* in a *for-range-declaration*. ...

In 7 [dcl] paragraph 1 amend the grammar rule for *simple-declaration* as follows:

simple-declaration:
attribute-specifier$_{opt}$ *decl-specifier-seq*$_{opt}$ ~~*attribute-specifier*$_{opt}$~~
init-declarator-list$_{opt}$;

and amend the text that follows as indicated:

...
The *simple-declaration*
attribute-specifier$_{opt}$ *decl-specifier-seq*$_{opt}$ ~~*attribute-specifier*$_{opt}$~~
init-declarator-list$_{opt}$;

is divided into ~~four~~three parts. Attributes are described in 7.6 [dcl.attr]. *decl-specifiers*, the principal components of a *decl-specifier-seq*, are described in 7.1. ~~The two optional *attribute-specifiers* and~~ *declarator*s, the components of an *init-declarator-list*, are described in Clause 8. The optional *attribute-specifier* in a *simple-declaration* appertains to each of the entities declared by the *declarator*s; it shall not appear if the optional *init-declarator-list* is omitted. [*Note:* In the declaration for an entity, attributes appertaining to that entity may appear both at the start of the declaration and after the *declarator-id* for that declaration. —*end note*][*Example:*

```
[[noreturn, nothrow]] void f [[noreturn]] (); // okay
```

—*end example*]

In 7 [dcl] paragraph 9, delete the second sentence:

... ~~If it is omitted, an *attribute-specifier* shall not appear.~~

In 7.1 [dcl.spec] paragraph 1 amend the grammar and text as follows:

...
decl-specifier-seq:
~~*decl-specifier-seq*$_{opt}$ *decl-specifier*~~
decl-specifier attribute-specifier$_{opt}$
decl-specifier decl-specifier-seq

The optional *attribute-specifier* in a *decl-specifier-seq* appertains to the type determined by the *decl-specifier-seq* (8.3 [dcl.meaning]). The *attribute-specifier* affects the type only for the declaration it appears in, not other declarations involving the same type.

In 7.1.6 [dcl.type] paragraph 1 amend the grammar and text as follows:

...
type-specifier-seq:

~~*type-specifier-seq*$_{opt}$ *type-specifier*~~
type-specifier attribute-specifier$_{opt}$
type-specifier type-specifier-seq

trailing-type-specifier-seq:
~~*trailing-type-specifier-seq*$_{opt}$ *trailing-type-specifier*~~
trailing-type-specifier attribute-specifier$_{opt}$
trailing-type-specifier trailing-type-specifier-seq

The optional *attribute-specifier* in a *type-specifier-seq* or *trailing-type-specifier-seq* appertains to the type denoted by the preceding *type-specifier*s (8.3 [dcl.meaning]). The *attribute-specifier* affects the type only for the construct it appears in, not other constructs involving the same type.

In 7.2 [dcl.enum] paragraph 1 amend the grammar and text as follows:

...
enum-head:
enum-key *attribute-specifier*$_{opt}$ *identifier*$_{opt}$
~~*attribute-specifier*$_{opt}$~~ *enum-base*$_{opt}$ ~~*attribute-specifier*$_{opt}$~~
enum-key *attribute-specifier*$_{opt}$ *nested-name-specifier identifier*
~~*attribute-specifier*$_{opt}$~~ *enum-base*$_{opt}$ ~~*attribute-specifier*$_{opt}$~~
opaque-enum-declaration:
enum-key *attribute-specifier*$_{opt}$ *identifier*
~~*attribute-specifier*$_{opt}$~~ *enum-base*$_{opt}$ ~~*attribute-specifier*$_{opt}$~~ ;
...

The ~~first~~ optional *attribute-specifier* in the *enum-head* and the *opaque-enum-declaration* appertains to the enumeration; the attributes in that *attribute-specifier* are thereafter considered attributes of the enumeration whenever it is named. ~~The second optional *attribute-specifier* in the *enum-head* and the *opaque-enum-declaration* shall appear only if the *enum-base* is present; it appertains to the *enum-base*.~~

In 8 [dcl.decl] paragraph 2 amend the first sentence as follows (note also the added comma):

2 The ~~two~~three components of a *simple-declaration* are the attributes (7.6 [dcl.attr]), the specifiers (*decl-specifier-seq*; 7.1), and the declarators (*init-declarator-list*). ...

In 8 [dcl.decl] paragraph 4 amend the grammar rule for *trailing-return-type* and *ptr-operator* as follows:

trailing-return-type:
-> ~~*attribute-specifier*$_{opt}$~~ *trailing-type-specifier-seq*
~~**->** *attribute-specifier*$_{opt}$~~ *abstract-declarator*$_{opt}$
ptr-operator:
***** *attribute-specifier*$_{opt}$ *cv-qualifier-seq*$_{opt}$
& *attribute-specifier*$_{opt}$

&& *attribute-specifier*$_{opt}$
::$_{opt}$ *nested-name-specifier* ***** *attribute-specifier*$_{opt}$ *cv-qualifier-seq*$_{opt}$

In 8.1 [dcl.name] paragraph 1 amend the grammar rule for *type-id* as follows:

type-id:
type-specifier-seq ~~*attribute-specifier*$_{opt}$~~ *abstract-declarator*$_{opt}$

In 8.3 [dcl.meaning] paragraph 3 amend the following phrase as indicated:

... of the form *attribute-specifier*$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier*$_{opt}$~~ and ...

In 8.3 [dcl.meaning] amend paragraph 5 as follows:

5 In a declaration *attribute-specifier*$_{opt}$ **T** ~~*attribute-specifier*$_{opt}$~~ **D** where **D** is an unadorned identifier the type of this identifier is "**T**". ~~The first optional *attribute-specifier* appertains to the entity being declared. The second optional *attribute-specifier* appertains to the type **T**, but not to the class or enumeration declared in the *decl-specifier-seq*, if any.~~

In 8.3.2 [dcl.ref] amend paragraph 1 as follows:

1 In a declaration **T D** where **D** has either of the forms

& *attribute-specifier*$_{opt}$ **D1**
&& *attribute-specifier*$_{opt}$ **D1**

and the type of the identifier in the declaration **T D1** is "*derived-declarator-type-list* **T**," then the type of the identifier of **D** is "*derived-declarator-type-list reference* to **T**." The optional *attribute-specifier* appertains to the reference type. Cv-qualified ...

In 8.3.5 [dcl.fct] paragraph 1 amend the grammatical form as follows:

D1 (*parameter-declaration-clause* **)** ~~*attribute-specifier*$_{opt}$~~ *cv-qualifier-seq*$_{opt}$
ref-qualifier$_{opt}$ *exception-specification*$_{opt}$ *attribute-specifier*$_{opt}$

In 8.3.5 [dcl.fct] paragraph 1 amend the grammatical form as follows:

D1 (*parameter-declaration-clause* **)** ~~*attribute-specifier*$_{opt}$~~ *cv-qualifier-seq*$_{opt}$
ref-qualifier$_{opt}$ *exception-specification*$_{opt}$ *attribute-specifier*$_{opt}$
trailing-return-type

In 8.3.5 [dcl.fct] paragraph 3 amend the grammar follows:

...
parameter-declaration-clause:
parameter-declaration_list$_{opt}$ ~~**...**$_{opt}$~~ *ellipsis-param*$_{opt}$
parameter-declaration_list$_{opt}$ **,** ~~**...**$_{opt}$~~ *ellipsis-param*
...
parameter-declaration:

attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~ *declarator*
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~ *declarator*
= *assignment-expr*
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
abstract-declarator$_{opt}$
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
abstract-declarator$_{opt}$ = *assignment-expr*

ellipsis-param:
. . . *attribute-specifier$_{opt}$*

and append the following text:

> The optional *attribute-specifier* in a *parameter-declaration* appertains to the parameter. The optional *attribute-specifier* in an *ellipsis-param* appertains to the variadic parameter it denotes.

In 8.4 [dcl.fct.def] paragraph 1, amend the grammar rule for *function-definition* as follows:

function-definition:
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
declarator *function-body*
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
declarator `= default ;`
attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
declarator `= delete ;`

and append the following sentence at the end of the paragraph:

> ... The optional *attribute-specifier* in a *function-definition* appertains to the function.

In 8.4 [dcl.fct.def] paragraph 2, amend the grammatical form as follows:

`D1 (` *parameter-declaration-clause* `)` *cv-qualifier-seq$_{opt}$* *ref-qualifier$_{opt}$*
exception-specification$_{opt}$ *attribute-specifier$_{opt}$* *trailing-return-type$_{opt}$*

In 8.4 [dcl.fct.def] paragraph 9, amend the grammatical form as follows:

attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
declarator `= default ;`

In 8.4 [dcl.fct.def] paragraph 10, amend the grammatical form as follows:

attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier$_{opt}$*~~
declarator `= delete ;`

In the introduction of 9.2 [class.mem], amend the first production in the grammar rule for *member-declaration* as follows:

member-declaration:

attribute-specifier$_{opt}$ *decl-specifier-seq* ~~*attribute-specifier*$_{opt}$~~
member-declarator-list **;**

...

In 10 [class.derived] paragraph 1 amend the grammar rule for *base-specifier* as follows:

base-specifier:
attribute-specifier$_{opt}$ **::**$_{opt}$
nested-name-specifier$_{opt}$ *class-name* ~~*attribute-specifier*$_{opt}$~~
attribute-specifier$_{opt}$ **virtual** *access-specifier*$_{opt}$
::$_{opt}$ *nested-name-specifier*$_{opt}$ *class-name* ~~*attribute-specifier*$_{opt}$~~
attribute-specifier$_{opt}$ *access-specifier* **virtual**$_{opt}$
::$_{opt}$ *nested-name-specifier*$_{opt}$ *class-name* ~~*attribute-specifier*$_{opt}$~~

In 12.3.2 [class.conv.fct] paragraph 1 amend the grammar rule for *conversion-type-id* as follows:

conversion-type-id:
type-specifier-seq ~~*attribute-specifier*$_{opt}$~~ *conversion-declarator*$_{opt}$

In 15 [except] paragraph 1 amend the grammar and text as follows:

...
exception-declaration:
attribute-specifier$_{opt}$ *type-specifier-seq declarator*
attribute-specifier$_{opt}$ *type-specifier-seq abstract-declarator*$_{opt}$
ellipsis-param
~~*type-specifier-seq*~~
~~**...**~~
...
The optional *attribute-specifier* in an *exception-declaration* appertains to the formal parameter of the catch clause (15.3 [except.handle]).