The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

6.8.6.2 The continue statement

Constraints

continue shall only appear

A **continue** statement shall appear only in or as a loop body. 1792

Commentary

The behavior of the **continue** statement is only defined in this context.

Coding Guidelines

Some coding guideline documents recommend against the use of the **continue** statement. The rationale appears to be based on an association being made with the **goto** statement.

The **continue** statement can be thought about either in implementation terms (i.e., jump to just before the end of the iteration statement that contains it) or conceptual terms (i.e., execute the next iteration of the loop). Neither of these are likely to require significant effort to comprehend. While it may not require a significant amount of effort to comprehend, readers still have to notice its existence in the source code. The **continue** statement has the same reader visibility issues as all jump statements.

jump statement causes jump to

Semantics

A **continue** statement causes a jump to the loop-continuation portion of the smallest enclosing iteration statement; 1793

Commentary

The **continue** statement performs the dual role of **goto** statement and virtual label creator.

Rationale

The C89 Committee rejected proposed enhancements to **continue** and **break** which would allow specification of an iteration statement other than the immediately enclosing one on grounds of insufficient prior art.

that is, to the end of the loop body. 1794

Commentary

The following C statement clarifies what is meant by "end of the loop body".

More precisely, in each of the statements 1795

```
while (/* ... */) {
   /* ... */
   continue;
   /* ... */
contin: ;
}
```

```
do {
   /* ... */
   continue;
   /* ... */
contin: ;
} while (/* ... */);
```

```
for (/* ... */) {
   /* ... */
   continue;
   /* ... */
contin: ;
}
```

unless the **continue** statement shown is in an enclosed iteration statement (in which case it is interpreted within that statement), it is equivalent to **goto contin;**.[135)]

Commentary

These equivalent mappings describe an effect, not an implementation technique. The **for** statement example could be rewritten as:

```
{
clause_1;
while (expression_2)
   {
     {
```

```
6         /* ... */
7         continue;
8         /* ... */
9         contin: ;
10        }
11      expression_3;
12      }
13  }
```

C++

The C++ Standard uses the example (6.6.2p1):

6.6.2p1

```
while (foo) {          do {              for (;;) {
  {                     {                 {
   // ...                 // ...            // ...
  }                     }                 }
contin: ;              contin: ;         contin: ;
}                      } while (foo);    }
```

The additional brace-pair are needed to ensure that any necessary destructors (a construct not supported by C) are invoked.

Coding Guidelines

Many developers have a mental model in which the **continue** statement jumps to the top of the loop. This model makes sense in that there is usually no information at the end of the loop body that developers need to consider (use of the **do** statement is relatively rare).

References