Document: N1563

Submitter: Martin Sebor, Derek M. Jones
Submission Date: 2011-03-06
Source:
Reference Document:
Version:
Date: 2011-03-06
Subject: DR #017, Question 19 missing from C99

The Response to DR #017, Question 19, is:

> If a fully expanded macro replacement list contains a function-like macro name as its last preprocessing token, it is unspecified whether this macro name may be subsequently replaced. If the behavior of the program depends upon this unspecified behavior, then the behavior is undefined.

The first sentence of the Response is reflected in J.1 of the standard below but there appears to be no corresponding wording in 6.10.3:

> When a fully expanded macro replacement list contains a function-like macro name as its last preprocessing token and the next preprocessing token from the source file is a (, and the fully expanded replacement of that macro ends with the name of the first macro and the next preprocessing token from the source file is again a (, whether that is considered a nested replacement (6.10.3).

In addition, the second sentence of the Response is not reflected in either Annex J.2 or anywhere in 6.10.3.

Suggested Technical Corrigendum

Either add the following paragraph to 6.10.3.4, immediately after paragraph 2:

> When a fully expanded macro replacement list contains a function-like macro name as its last preprocessing token and the next preprocessing token from the source file is a (, and the fully expanded replacement of that macro ends with the name of the first macro and the next preprocessing token from the source file is again a (, it is unspecified whether that is considered a nested replacement.

or, alternatively, an example to the end of 6.10.3.4:

> EXAMPLE 1 Given the following macro definitions
>
> ```
> #define f(a) a*g
> #define g(a) f(a)
> ```
>
> The invocation

```
f(2)(9)
```

results in undefined behavior. Among the possible behaviors are the generation of the preprocessing tokens:

```
2*f(9)
```