Macro Replacement In Pragmas

N810/J11 98-009
Feb 5, 1998
Randy Meyers

Introduction

The C9x forbids macro expansion in standard pragmas. C9x neither requires or forbids that macros be expanded in non-standard pragmas, since such pragmas are implementation defined.

Erin Harris has pointed out that the Draft is not clear what happens in situations like:

```
#define MYPRAGMA  STDC FP_CONTRACT ON
#pragma MYPRAGMA
```

Since by the time an implementation that expands macros in pragmas has expanded **`MYPRAGMA`** in order to discover **`STDC`** follows **`#pragma`**, the implementation has also expanded the rest of the replacement list of **`MYPRAGMA`**.

Since the Draft neither forbids or requires implementations to expand macros in pragmas in their own pragmas (but forbids macro expansion in standard pragmas), the only portable specification of the standard pragmas must be that standard pragmas can not portably result from macro expansion of a token after **`#pragma`**. If a standard pragma could result from macro expansion after **`#pragma`** then all implementations would have to support macro expansion in pragmas. This is a bad idea as I showed in a previous paper: If macros are expanded in pragmas, and the "keywords" of the pragma look like identifiers in the user's namespace, then such pragmas can not be used in any standard header since the user would not be able to define macros of the same name as the pragma keywords before including the header.

So, what the Draft needs to say is that if the preprocessor token following **`#pragma`** before any macro expansion is **`STDC`**, then no macro expansion takes place in the pragma directive and the pragma is a standard pragma.

In other words, in the above example, then pragma is *not* a standard pragma, and so no portable meaning. An implementation is free to macro expand **`MYPRAGMA`** and get the tokens of what looks like a standard pragma (and perform further macro expansions or not, as it pleases). The implementation is also then free to interpret the final results of expansion as if it has the same meaning as the standard **`FP_CONTRACT`** pragma, but such behavior is an implementation extension, and in no way portable: another implementation might not expand **`MYPRAGMA`** at all, or if it does expand it, might treat it in an implementation defined way. (I'd personally recommend that an implementation that expands macros in pragmas treat the results of such expansion as a standard pragma if it looks like a standard pragma.)

Implementation Details

The above semantics does not place an undue burden on implementations.

Implementations that do no expand macros in pragmas have nothing to do at all.

Implementations that do expand macros in pragmas, merely have to look at the preprocessor token that follows **`#pragma`** before doing any macro expansion in order to make the right decisions. If the token is **`STDC`**, no macro expansion is done, otherwise the implementation can perform full macro processing.

Note that the language already has a similar situation: macros are expanded in a **`#if`** directive unless they follow a **`defined`** token.

Changes to Subclause 6.8.6 Pragma directive

Change the following in Paragraph 1 from:

A preprocessing directive of the form
`# pragma` *pp-tokens opt new-line*
where the preprocessing token **`STDC`** does not immediately follow the **`pragma`** on the directive causes the implementation to behave in a manner which it shall document.

To:

A preprocessing directive of the form
`# pragma` *pp-tokens opt new-line*
where the preprocessing token **`STDC`** does not immediately follow the **`pragma`** 1n the directive prior to any macro replacements[new footnote] causes the implementation to behave in a manner which it shall document.

Add the new footnote referenced above:

The Standard never requires an implementation to perform macro replacement in pragmas, but it does forbid macro replacement in standard pragmas (pragmas where **`STDC`** immediately follows **`#pragma`**). If the result of macro replacement in a non-standard pragma has the same form as a standard pragma, then an implementation is still free to behave in an implementation-defined manner, which might be to treat the result as if it were a standard pragma.

Change the following in Paragraph 2 from:

If the preprocessing token **`STDC`** does immediately follow the **`pragma`** on the directive, then no macro replacements are performed on the directive, and the directive shall have one of the following forms whose meaning is described elsewhere:

to:

If the preprocessing token **`STDC`** does immediately follow the **`pragma`** in the directive prior to any macro replacements, then no macro replacements are performed on the directive, and the directive shall have one of the following forms whose meaning is described elsewhere: