Document Number: WG14 N481/X3J11 95-082

C9X Revision Proposal

Title: Add // comments
Author: Douglas A. Gwyn
Author Affiliation: United States Army Research Laboratory
Postal Address: 801-L Cashew Court, Bel Air, MD 21014, USA
E-mail Address: gwyn@arl.mil
Sponsor: X3J11
Date: 1995-10-16
Proposal Category:

__ Editorial change/non-normative contribution
__ Correction
● New feature
  __ Environment
  __ Language
  ● Preprocessor
  __ Library
    __ Macro/typedef/tag name
    __ Function
__ Addition to obsolescent feature list
__ Addition to Future Directions
__ Other (please specify)

Prior Art: Naval Postgraduate School, Geotronics Corp., and C++. The first were contributed to the Unix user community, based on the Reiser preprocessor and Ritchie C compiler, during the late 1970s. C++ supports // comments, and they are often seen in otherwise "pure" C code when the programmer is using a dual C/C++ implementation, which is a common situation these days.

Target Audience: all C programmers
Related Documents (if any): all C++ specifications
Proposal Attached: ● Yes __ No, I just want to gauge the level of interest

Abstract: Current (C89) practice for the very useful "marginal note" in source code is to use the only real commenting facility in C89 as follows:

```
        status = print( &doc_hdr );      /* could be out of paper */
```

Each use of this paradigm involves, in addition to the actual comment, six characters of overhead. Multiply this by a large number of marginal comments and it becomes evident that even a modest reduction in overhead would benefit all C programmers.

During the early years of Unix, the book "Software Tools" by Kernighan and Plauger was very influential; it introduced a C-like language, Ratfor, which supported comments consisting of # followed by arbitrary comment text to the end of the source line. Later languages often adopted this or a similar syntax for comments. Unfortunately, # already had conflicting meaning in C, but in the mid-1970s the Naval Postgraduate School made a few simple changes to the Unix C compiler and preprocessor (which were separable at that time) to support a similar comment style in addition to the standard /*...*/ style. They used // instead of #, but otherwise it was the same scheme as in Ratfor. While at Geotronics Corporation in the late 1970s, I made similar changes to the Reiser preprocessor and Ritchie C compiler and contributed them to USEnix, the Unix user's group. I also created a small utility based on a finite state machine to convert //-commented source code to standard /*...*/-commented code, to permit exporting our nonstandard (due to use of // comments) source code to other C compilers. Typical usage of // comments is:

```
        status = print( &doc_hdr );   // could be out of paper
```

which reduces overhead by three characters per marginal comment.

Bjarne Stroustrup was apparently aware of some of this prior art, and adopted // comments as part of his C++ language specification. Consequently, all C++ translators support them and they are now widely used. Indeed, some C programmers who use merged C/C++ translators are surprised when they port their code to other C implementations and their // comments are rejected as syntax errors.

Proposal: The intent is to add // comments as described above, so that they have properties virtually identical to /*...*/ comments, apart from the difference in syntax. While compatibility with the draft C++ standard is desirable, differences between the standards requires that the required changes for C9x be determined in its own context.

There are many possible ways to add // comment support, but to ensure that they behave as much like /*...*/ comments as possible, I suggest the following set of detailed edits:

No change to §5.1.1.2 **Translation phases.** (Comment still is replaced with a space, after logical line splicing.)

No change to §6.1 **Lexical elements.** (Comment still is handled specially while scanning for preprocessing tokens.)

Change in §6.1.7 **Header names, Semantics** from:

> If the characters ', \, ", or /* occur in the sequence between the < and > delimiters, the behavior is undefined. Similarly, if the characters ', \, or /* occur in the sequence between the " delimiters, the behavior is undefined.

to:

> If the characters ', \, ", //, or /* occur in the sequence between the < and > delimiters, the behavior is undefined. Similarly, if the characters ', \, //, or /* occur in the sequence between the " delimiters, the behavior is undefined.

(This change is debatable, since **#include "//my.com/a.h"** might be better implementation-defined than undefined.)

Change in §6.1.9 **Comments,** from:

> The contents of a comment are examined only to identify multibyte characters and to find the characters */ that terminate it.
> [*FOOTNOTE:* Thus, comments do not nest.]

to:

> The contents of such a comment are examined only to identify multibyte characters and to find the characters */ that terminate it.
> [*FOOTNOTE:* Thus, /*...*/ comments do not nest.]

and add a new paragraph:

> Except within a character constant, a string literal, or a comment, the characters // introduce a comment that includes all multibyte characters up to, but not including, the next new-line character. The contents of such a comment are examined only to identify multibyte characters and to find the terminating new-line character.

(This adds // comments to the language and prevents the two forms of comment from interleaving.) I think some **Examples** added to this section would be helpful; I can provide these upon request, also **Rationale** wording. The **Examples** would obviate any need to footnotes elsewhere to warn programmers about various subtleties involving comments of either kind.

Also, I suggest a generic editing pass over the C9x draft to change all *genuinely* marginal comments to use // style. (Use of "/* ... */" to represent elided source test should not be changed, and there are some other embedded /*...*/ comments, particularly in the preprocessor section, that should not be changed since they are essential to the functioning of the examples.)