

## IEEE Standards Interpretations for IEEE Std 1003.2™-1992 IEEE Standard for Information Technology--Portable Operating System Interfaces (POSIX®)-- Part 2: Shell and Utilities

Copyright © 1996 by the Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue New York, New York 10016-5997 USA All Rights Reserved.

Interpretations are issued to explain and clarify the intent of a standard and **do not** constitute an alteration to the original standard. In addition, interpretations are not intended to supply consulting information. Permission is hereby granted to download and print one copy of this document. Individuals seeking permission to reproduce and/or distribute this document in its entirety or portions of this document must contact the IEEE Standards Department for the appropriate license. Use of the information contained in this document is at your own risk.

IEEE Standards Department, Copyrights and Permissions, 445 Hoes Lane, Piscataway, New Jersey 08855-1331, USA

### Interpretation Request #69

**Topic:** patch -D **Relevant Clauses:** 5.22

POSIX.2 Subclause 5.22 specifies the semantics of the “patch” utility. In subclause 5.22.3 the behavior of the -D option is specified as follows: -D <define> Mark changes with the C preprocessor construct: #ifdef <define> ... #endif

The option-argument <define> shall be used as the differentiating symbol. Can a conforming implementation of the patch utility use “#ifndef” to mark changes that constitute deletions from the original file? Can an implementation freely choose to use #ifdef or #ifndef when either is correct (i.e. gives a file that, after preprocessing, has the correct contents)? For example, the files aa #ifdef uppercase BB #else bb #endif cc and aa #ifndef uppercase bb #else BB #endif cc are equivalent in this sense. Are both valid output files from a call to a conforming “patch -D uppercase ...”?

Note that the use of #ifndef is historical practice. Note also that if it is not permitted, implementations can still conform, but only through the use of such clumsy constructs as #ifdef <define> #else ... #endif In a similar vein: can a conforming implementation of “patch” use the construct #if defined(<define>) rather than #ifdef <define> In general, how broadly can the phrase “the C preprocessor construct:...” be interpreted? Thank you for your attention to this matter.

### Interpretation Response

The standard states the behaviour for #ifdef and #endif and conforming implementations must conform to this. The standard makes no restrictions for the use of further C preprocessor directives between #ifdef and #endif. The standard does not allow using #ifndef and #if defined in the manner specified in the interpretation request. Concerns

about this are being referred to the sponsor.

**Rationale for Interpretation**

None.