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

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.