IEEE Standards Interpretations for IEEE Std 1003.1™-2001 IEEE Standard for Information Technology - Portable Operating System Interface (POSIX®)

Interpretation Request #36

Topic: BRE nested subpatterns **Relevant Sections:** XBD 9.3.6

There seems to be a discrepancy between the description of BREs in XBD6 and the description of regcomp() in XSH6 as regards the treatment of nested subpatterns with a following * repeater. The example that prompted this is whether: echo aba | sed 's/\(a\(b\)*\)*/<\1|\2>/' should output <a|b> or <a|>. According to the description of BREs: "If the subexpression referenced by the back-reference matches more than one string because of an asterisk ('*') or an interval expression (see item (5)), the back-reference shall match the last (rightmost) of these strings." which seems to imply that the correct output is <a|b> since the last string that the second subpattern matches is the b that it got from the first iteration of the first subpattern. (In the second iteration of the first subpattern the second subpattern doesn't match anything.)

Several sed implementations do output <a|b>. However, the description of regcomp() says: "3. If subexpression i is contained within another subexpression j, and i is not contained within any other subexpression that is contained within j, and a match of subexpression j is reported in pmatch[j], then the match or non-match of subexpression i reported in pmatch[i] shall be as described in 1. and 2. above, but within the substring reported in pmatch[j] rather than the whole string. The offsets in pmatch[i] are still relative to the start of string." Since the second subpattern in the example does not match anything within the last string matched by the first subpattern, this implies that regcomp() would report the second subpattern as a non-match. Although there is no requirement for sed to be implemented using regcomp(), presumably this is an unintentional difference between the descriptions of regcomp() and BREs. The text quoted above on BREs was an addition in POSIX.1-2001 intended to clarify what happens for repeated subpatterns. Possibly the added text is too simplistic and should have treated nested subpatterns in the way that regcomp() does. On the other hand, since several

current sed implementations do not behave as the regcomp() description implies, perhaps it is the regcomp() description that needs alteration. (So far I am only aware of one, historical, sed implementation which outputs <a|> for the above example, however I have not tested many.)

Issue an official interpretation of the current requirement. Depending on the outcome of the interpretation, implement a clarification or correction in the next revision.

Interpretation Response
The standard is unclear on this issue, and no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.

Rationale for Interpretation
None.

Notes to the Editor (not part of this interpretation)
Proposed changes for a future revision: 9.3.6 BREs Matching Multiple Characters [* Current description (page 174 lines 6129-6135): *] For example, the expression "\(.*\)\1$" matches a line consisting of two adjacent appearances of the same string, and the expression "\(a\)*\1" fails to match 'a' . When the referenced subexpression matched more than one string, the back-referenced expression shall refer to the last matched string. If the subexpression referenced by the back-reference matches more than one string because of an asterisk ('*') or an interval expression (see item (5)), the back-reference shall match the last (rightmost) of these strings. [* Change to: *] The string matched by a contained subexpression shall be within the string matched by the containing subexpression. If the containing subexpression does not match, or if there is no match for the contained subexpression within the string matched by the containing subexpression then back-reference expressions corresponding to the contained subexpression shall not match. When a subexpression matches more than one string, a back-reference expression corresponding to the subexpression shall refer to the last matched string. For example, the expression "^\(.*\)\1$" matches lines consisting of two adjacent appearances of the same string, the expression "\(a\)*\1" fails to match 'a', the expression "\(a\(b\)*\)*\2" fails to match 'abab', and the expression "^\(ab*\)*\1$" matches 'ababbabb' but fails to match 'ababbab'. ed [* Current description (page 345 lines 13312-13313): *] As a more general feature, the characters '\n' , where n is a digit, shall be replaced by the text matched by the corresponding back-reference expression. [* Add this: *] If the corresponding back-reference expression does not match then the characters '\n' shall be replaced by the empty string. ex [* Current description (page391 lines 15168-15170): *] The sequence '\n' , where n is an integer, shall be replaced by the text matched by the pattern enclosed in the nth set of parentheses '\(' and '\)'. [* Change to: *] The sequence '\n' , where n is an integer, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match then the characters '\n' shall be replaced by the empty string. expr [* Current description (page 431 lines 16738-16739): *] Alternatively, if the pattern contains at least one regular expression subex-

pression "[\(...\)]", the string corresponding to "\1" shall be returned. [* Change to: *] Alternatively, if the pattern contains at least one regular expression subexpression "[\(...\)]", the string matched by the back-reference expression "\1" shall be returned. If the back-reference expression "\1" does not match then the null string shall be returned. pax [* Current description (page 707 line 27455): *] '\n' (where n is a digit) backreferences, ... [* Change to: *] '\n' (where n is a digit) back-references, ... sed [* Current description (page 846 lines 32803-32804): *] The characters "\n", where n is a digit, shall be replaced by the text matched by the corresponding backreference expression. [* Change to: *] The characters "\n", where n is a digit, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match then the characters "\n" shall be replaced by the empty string.