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 #78
Topic: ex/vi **Relevant Clauses:** 5.10

The following are issues that I've found in the current POSIX 1003.2-1992 specification of the ex/vi utilities. For each of the areas discussed below I believe that the current POSIX 1003.2 specification does not match historic practice.

#1 Standard Input; Subclause 5.10.5.1, page 516, line 1053 Historic practice was that if the standard input was not a terminal, it was the same as specifying the - s, i. e. batch, option.
#2 Extended Description; Subclause 5.10.7, page 519, line 1170 Historic practice was that the real user ID of the process was checked, not the effective user ID. Weirdnix, the original file could be modified once the edit buffer was written to any file. Change it to be when the original file is written. The phrase "buffer" is used sometimes when ''edit buffer' ' is meant. Additionally, the phrase "editing buffer" should be replaced with "edit buffer", and the definition of "edit buffer" needs to come first, before almost anything else.
#3 Addressing; Subclause 5.10.7.1, page 520, line 1212 Historic practice is that the '+' sign may be omitted. This is documented for the ed command in 1003.2-1991, page 262, line 3635. Historic practice is that any missing address sets the address to, e. g. ", + 2" is the same as ". , + 2", and "+ 4, , , , " is the same as ". ". Historically, this was true for addresses AFTER semicolons, too, but NOT for addresses BEFORE semicolons. For consistency, it should be specified for any missing address. Historic practice is that addresses could have random #60; blank#62; characters thrown in. Historic practice is that #60; blank#62; separated absolute addresses were additive, e. g. "35p" displayed line 8. The % sign wasn' t really an address -- it couldn' t be followed by further addresses. The wording for the ex addressing is unnecessarily different from that for the

ed command, and should be made consistent.
#4 Synopsis; Subclause 5.10. 1, page 515, line 982, 984 The - r flag and the - t flag don’ t make any sense together, and historically didn’ t do anything sensible.
#5 Options; Subclause 5.10. 3, page 515, line 997 The order of execution is as described in the Extended Description, but should include mention of the - r flag as well.
#6 Environment Variables; Subclause 5.10.5.3, page 517, line 1104 Historically, the SHELL environment variable was only used to load the shell edit option, after that it was ignored.
#7 Asynchronous Events; Subclause 5.10.5.4, page 517, line 1125 Historically, SIGTERM had the same effect as SIGHUP. The buffer should be saved unless it was a complete write of the file. This change needs to be applied in lots of places. Historically, SIGINT in text input mode resolved the input into the file. The current specification pretty much covers historic practice, but it needs to be concentrated into the SIGINT language, not spread all over the standard.
#8 Extended Description; Subclause 5.10.7, page 519, line 1149 The: next command takes files as arguments, but does not set the alternate pathname other than as part of switching to a new file. The description of how/when the alternate pathname is set bears no relationship to historic practice. While we’ re there, change all the “current files to “current pathname”. If the screen is too small for any single line, the editor has to be able to punt. (The only solution is logical lines ÒI am NOT going to make an editor handle physical lines in a screen where a single line doesn’ t fit).
#9 write; Subclause 5.10. 7. 2. 37, page 532, line 1670 Historically, if the file doesn’ t have a current pathname, the specified file becomes the current pathname, as is specified for the read command.
#10 Extended Description; Subclause 5.10. 7, page 519, line 1149 There’ s no description of the value of the current and alternate pathnames when ex/vi starts up.
#11 Command Description; Subclause 5.10. 7. 2, page 522, line 1267 Historically, text was stored in buffers even when a buffer wasn’ t specified. Historically, buffers were either character or line mode oriented. This impacts almost all operations that deal with them. Historically, several different commands put text into the buffers, not just yank and delete. Historically, ex did not modify the numeric buffers in any way, nor could it access them. Make the buffer description work for both ex and vi, I only want to explain this once.
#12 Extended Description; Subclause 5.10. 7, page 519, line 1173 Many (most?) ex commands aren’ t legal in the initial . exrc files or EXINIT variable, because no file has yet been read in.
#13 insert; Subclause 5.10. 7. 2. 11, page 526, line 1431 Historically, insert was affected by the number option. Historically, interrupting or suspending insert was the same as entering ‘ . ‘ , but any partial line was discarded.
#14 Extended Description; Subclause 5.10. 7, page 519, line 1173 Historically, you could escape #60; newline#62; characters in the . exrc files, although vi generally messed it up after that.
#15 vi Command Descriptions; Subclause 5. 35. 7. 1, page 631, line 4989 Historically, motions were a lot more complex than this. The general rule was that it was from the starting or stopping cursor position that came first in the file to the starting or stopping

position that came last in the file, but there were lots of special cases.
#16 Command Descriptions; Subclause 5.10. 7. 2, page 522, line 1300 The pattern was subjected to shell expansion only if the meta characters ~, {, [, * , ?, $, `, ` , " or \ were present in the pattern.
#17 abbrev; Subclause 5.10. 7. 2. 1, page 523, line 1328 Abbreviation replacement was a lot trickier than this. Generally, vi took the type of the character before the word/non-word character that triggered the check, and moved back in the inserted text until the type changed or a #60; blank#62; was found.
#18 append; Subclause 5.10.7.2.2, page 523, line 1333 Historically, append was affected by the number option. Historically, interrupting or suspending append was the same as entering ` . ` , but any partial line was discarded.
#19 change; Subclause 5.10. 7. 2. 4, page 524, line 1346 Historically, change was affected by the number option. Historically, interrupting or suspending change was the same as entering ` . ` , but any partial line was discarded. Historically, change copied replaced lines into the unnamed buffer.
#20 chdir; Subclause 5.10. 7. 2. 5, page 524, line 1354 Historically, directory was only shell expanded if one of the meta characters ~, {, [, * , ?, $, `, ` , " or \ were present in the argument. Historically, cd only warned if the current pathname didn' t start with a leading slash. Historically, the buffer had to be completely written, not just any write.
#21 copy; Subclause 5.10. 7. 2. 6, page 524, line 1366 Historically, copy put the cursor on the last line copied.
#22 edit; Subclause 5.10. 7. 2. 8, page 525, line 1388 Historically, the command given to the edit command could be any ex command. Historically, if edit was executed from vi mode and it was the previous file, the previous location was restored. This should be extended for vi to permit any previous location to be restored. As ex is more script/automatic command oriented, it should continue to always move to the end of the file. Historically, the ex command was not affected by the autowrite option or the writeany option. Historically, any replacement of the current edit buffer contents didn' t fail because the file was unreadable, the file was just empty and the name was set. This semantic is difficult to changeÒany change would result in the: next command failing and it being impossible to skip past a file in the argument list.
#23 global; Subclause 5.10. 7. 2. 10, page 525, line 1414 Historically, any non- alphabetic character could be used as the global pattern delimiter. You can' t use an alphabetic because of the substitute command use of flags immediately after the ` s' of the command. Historically, the default command is the print command, i. e. all of the print flags etc. apply. Some ex commands don' t make sense for global commands. Once the underlying file has changed, the command should quit. Historically, if lines that are marked are deleted, the global command ignores them.
#24 join; Subclause 5.10. 7. 2. 12, page 526, line 1439 The historic rules for joining where a bit simpler, actually, but different from the current POSIX. 2 specification: Discard leading spaces from the second line. If the second line is now empty, ignore it. If the first line ended in a #60; blank#62; character, or the next character of the second line is a) character, join the lines without further modification. If the last character of the first line is a ` . ` , join the lines with two #60; space#62; characters between them. Join the lines with a single #60; space#62; character between them.

#25 map; Subclause 5.10. 7. 2. 14, page 527, line 1458 Historical function key behavior should be permitted.
#26 move; Subclause 5.10. 7. 2. 15, page 527, line 1477 Historically it' s an error to attempt to move within the range.
#27 list; Subclause 5.10. 7. 2. 13, page 526, line 1455 print; Subclause 5.10.7.2.21, page 528, line 1526 Historically, the number option was the same as having the #flag set.
#28 open; Subclause 5.10. 7. 2. 19, page 528, line 1511 Historically, open didn' t succeed if pattern wasn' t found. Historically, open set the current line to the address.
#29 put; Subclause 5.10.7.2.22, page 528, line 1533 The language for the put option is needlessly verbose, and the word restored bothers me a lot. Historically, put took an argument of 0.
#30 read; Subclause 5.10.7.2.24, page 529, line 1545 Historically, the shell option value was used for the read ! command, the SHELL environment variable only initialized the option. Historically, the program named by the shell option was calledwith a - c flag, and the rest of the arguments were a single string. Historically, the arguments to the read ! command were bothfile and last- bang expanded, and redisplayed the arguments if they changed as a result of this expansion. In addition, the read ! form of the command set the last- bang value. Historically, the read ! command was affected by the autoprintoption, and the regular read command was not. Historically, read did not permit reads from non- regular files. Historic practice is to write the number of lines/characters read. This should be specified, but characters should be changed to bytes, and mentioned in the Rationale.
#31 recover; Subclause 5.10.7.2.25, page 529, line 1557 Historically, recover did cursor positioning more like the edit command, not the read command. Historically, recover behaved like edit if there was no recovery file.
#32 rewind; Subclause 5.10.7.2.26, page 529, line 1562 Historically, rewind did cursor positioning more like the edit command, not the read command.
#33 set; Subclause 5.10.7.2.27, page 529 line 1573 Historically, the term variable was always displayed (as was "redraw", but I think it' s safe to drop that one) . Historically, the escaping backslash was discarded.
#34 shell; Subclause 5.10.7.2.28, page 530 line 1587 Historically, it was the program named by the shell option.
#35 substitute; Subclause 5.10. 7. 2. 30, page 530 line 1596 Historically, substituting a ^ M into a line split the line. Historically, the command was ' s' , not "substitute".
#36 tags; Subclause 5.10. 7. 2. 32, page 531 line 1628 Historically, the cursor was set as for: edit, if the tagsearch path or line number was not found.
#37 suspend; Subclause 5.10. 7. 2. 31, page 530, lines 1626 There' s no description of how the ! character affects autowrite.
#38 visual; Subclause 5.10. 7. 2. 36, page 531, lines 1661 Historically, the count value was reapplied each time the window was redrawn in visual mode. This count value was separate from the window option, however, and remembered separately.
#39 write; Subclause 5.10. 7. 2. 37, page 532, line 1670 Writing to standard output isn' t going to work well for visualmode. Historically, appends were not otherwise validated, for example by the readonly flag, it was historically always legal to append to a file. His-

torically, the current pathname can be set by the read and write commands as well, and needs to set the "name changed" flag for writing. This check only happened once, however, if you forced the write after a name change, subsequent writes succeeded without forcing. Historic practice is to write the number of characters, not the number of bytes. The standard correctly says bytes, butthe change should be mentioned in the Rationale. Historic practice is to pass the line to the command named by the shell edit option, not the SHELL environment variable. Historically, the program named by the shell option was called with a - c flag, and the rest of the arguments were a single string. Historically, the arguments to the write ! command were both file name and last- bang expanded. If the command line changes as a result of this expansion, it was redisplayed. In addition, the write ! form of the command set the last- bang value. Write shouldn' t fail if there' s no current pathname unless no file was specified.

#40 xit; Subclause 5.10. 7. 2. 38, page 532, line 1692 The editor should not exit, regardless of the reason for the failure.

#41 Adjust Window; Subclause 5.10. 7. 2. 40, page 533, line 1705 Rename the command to ' z' , not ' Adjust Window' for consistency. Historically, a count to the z command set the value of the window option. Historically, a ! after the z command displayed a single screen, (not the window option value, but the full display) instead of the scroll option value x 2. The ! had no effect if a count was specified. The count did not change the remembered z command size. Historically, ' z' incremented the current line by one if no line or type was specified. The calculations in 1003. 2- 1992 are wrong: my guesses at the necessary corrections are: . , =: line = line - (count - 1) / 2; count = (((count - 1) / 2) * 2) + 1; ^: line = line - (((#of ^ ' s + 1) * count) - 1) ; +: line = line + (((#of + ' s) * count) + 1) ; -: line = line - (((#of - ' s) * count) - 1) ; Historically, it was an error if the starting line wasn' t legal, but the ending line could be past EOF.

#42 Escape; Subclause 5.10. 7. 2. 41, page 533, line 1724 Historically, the shell option value was used for the escape command, the SHELL environment variable only initialized the option. Historically, vi prompted the user before continuing, e. g. refreshing the screen. Historically, if a range was specified, the cursor could be moved. Historically, the program could interact with the user, so the standard input etc. need to be specified, e. g. if the user does "! mail", the standard file descriptors will need to be specified. Historically, lines deleted by a filter were saved to the unnamed buffer.

#43 Shift Left; Subclause 5.10.7.2.42, page 533, line 1742 Shifted lines were shifted by column positions, not character positions. Shifted lines were copied into the unnamed buffer. Historically, #60; tabs#62; were turned into spaces, or visa- versa, as necessary.

#44 Shift Right; Subclause 5.10. 7. 2. 43, page 533, line 1748 Shifted lines were shifted by column positions, not character positions. Shifted lines were copied into the unnamed buffer. Historically, #60; tab#62; s were turned into spaces, or visa- versa, as necessary.

#45 Scroll; Subclause 5.10. 7. 2. 45, page 534, line 176.

#60; control- D#62; worked even if the terminal was using a different character for the eof character.

#46 Execute; Subclause 5.10.7.2.47, page 535, line 1778 Historically, buffer was optional, the default was the last buffer executed. The text got this right, but the Synopsis got it wrong. Buffer execution was a bit more complex than is described; it' s pushed on

the input queue, and, if in line mode, or if in character mode, and not the last line in the buffer, a #60; newline#62; character is appended, Buffers were subject to further abbreviations, mapping, and buffer execution. (The implementation was that the characters were pushed on the tty queue.) The remap option did not effect this.
#47 delete; Subclause 5.10. 7. 2. 7, page 524, line 1373 The lines were deleted into a buffer, not from it.
#48 Regular Expressions; Subclause 5.10. 7. 3, page 535, line 1799 Document that the backslashes are discarded.
#49 autoindent; Subclause 5.10. 7. 5. 1, page 536, line 1837 Historically, #60; erase #62; and other normal erase characters could not erase autoindent characters .
#50 list; Subclause5.10.7.5.8, page 537 , line 1884 This is the standard . Rephrase "should be unambiguous" to "shall be unambiguous" , or take the wording from the ed command .
#51 paragraph; Subclause 5 . 10 . 7 . 5 . 12 , page 538 , line 1905 paragraphmovements were much more complex than this . Move this language into the vi specification .
#52 sections; Subclause 5.10.7.5.18, page 539 , line 1945 Section movements were much more complex than this . Move this language into the vi specification .
#53 readonly; Subclause 5 . 10 . 7 . 5 . 14 , page 538 , line 1923 Implementations should not be restricted from providing the best possible information to the user - - the file may have write permission but be on a read - only file system . Historically , "appropriate privileges" were not sufficient reason to turn readonly off , e . g . if you were root on a UNIX system the readonly option would still be set based on the file permissions modes .
#54 Command Descriptions; Subclause 5 . 10 . 7 . 2 , page 522 , line 1300 Historically , the process of ''shell word expansion was based on the shell edit option , not a specific shell .
#55 shell; Subclause 5 . 10 . 7 . 5 . 19 , page 539 , line 1955 Simplification is always safer , and I don ' t want to bother figuring out if this is right .
#56 shiftwidth; Subclause 5 . 10 . 7 . 5 . 20 , page 539 , line 1961 Width should be specified in columns .
#57 tabstop; Subclause 5 . 10 . 7 . 5 . 23 , page 540 , line 1975 The phrase software tab stop is too ambiguous .
#58 tags; Subclause 5 . 10 . 7 . 5 . 24 , page 540 , line 1975 Historically , tags could be #60; tab #62; separated . More random simplification .
#59 warn; Subclause 5 . 10 . 7 . 5 . 27 , page 540 , line 1996 The phrase '' ! command escape ' ' doesn ' t make sense .
#60 window; Subclause 5 . 10 . 7 . 5 . 28 , page 541 , line 2000 Historically , the ' z ' command wasn ' t affected by the value of window . Historically , the value was LINES - 1 , not LINES .
#61 wrapscan; Subclause 5 . 10 . 7 . 5 . 29, page 541 , line 2012 N and n also wrapped , not just / and ? .
#62 wrapmargin; Subclause 5 . 10 . 7. 5. 30, page 541 , line 2017 Should use ''columns ' ' , not ''spaces ' ' . Historically , there was a notion of which character caused the wrap to happen , if it was a #60; blank #62; , a subsequent #60; space #62; was discarded . The entire paragraphneeds rephrasing to support that feature . Historically , wrapmargin

did not apply to maps which included counts and inserted characters , e . g . : : map K 5aabc def ghi #60; escape #62; was not affected by wrapmargin . People wrote macros that depend on this feature , it needs to be left unspecified .
#63 writeany; Subclause 5 . 10 . 7 . 5 . 31 , page 541 , line 2028 Historically , writeany only turned off the overwriting and partial buffer writing checks .
#64 taglength; Subclause 5 . 10 . 7 . 5 . XX , page 540 , line 1978 The historic taglength option was omitted from the standard .
#65 Extended Description; Subclause 5 . 35 . 7 , page 629 , line 4894 It ` s possible to have a single ~ on a line by itself because that ` s the text in the edit buffer.
#66 Extended Description; Subclause 5 . 35 . 7 , page 629 , line 4890 Interrupt historically retained the lines just like escape , but alerted the terminal . People actually use this instead of escape because it ` s easier to type . Interrupt cancelled any partially entered command when entered in command mode.
#67 Extended Description; Subclause 5 . 35 . 7 , page 629 , line 4898 Don ` t require that lines that don ` t fit in their entirety are not displayed . This makes the physical/logical line problem worse .
#68 vi Command Descriptions; Subclause 5 . 35 . 7 . 1 , page 629 , line 4903 General reworking , trying to get the discussion of the current line and column to work . The major change is that the current specification doesn ` t handle characters that take up more than a single screen column . The intent of this fix is to require that the current column be a screen column where some part of the correct character is displayed , and then rely on the adjustment rules to get it on the exact correct column . Historically , each time the window is redrawn , only the window option number of lines are displayed . Historically , the window option could not be set to any value larger than LINES - 1 . This change should include the deletion of the paragraphon page 632 , lines 5051 - 5058 , and adding appropriate wording to the paragraphs on page 629 , around lines 4903 . The fact that the current line has changed , and the screen is required to display the current line should cause the update to happen . This change should include the deletion of the paragraphon page 632 , lines 5059 - 5078 , and adding appropriate wording to the paragraphs on page 629 , around lines 4903 . More specifically , as noted in Request for Interpretation #64 (43) , this has to be specified on a command - by - command basis , there simply aren ` t any general rules . This change also includes a large number of changes , which are NOT individually identified , throughout the standard , for the Current line/column lines . This change also includes a large number of changes , which are NOT individually identified , throughout the standard , where the description of the command implies implementation . The control - D command does not “scroll forward N lines” , it displays a new screen that includes a new current line . How we get there is the implementation ` s problem . This change also includes a large number of changes , which are NOT individually identified , throughout the standard , where the description of the command now has to specify the possible error conditions , for example , when do the scrolling commands fail . There ` s no discussion of folding lines , and there needs to be , since you probably don ` t want leftright scrolling as a default .
#69 vi Command Descriptions; Subclause 5.35.7.1, page 630 , line 4931 The new cursor location may be different , but it should be strictly specified .
#70 vi Command Descriptions; Subclause 5.35.7.1, page 631 , line 5019 Historically ,

sections were paragraphs were sentences . It was also possible to specify single matching characters by putting spaces in the sections/paragraphs options .
#71 Extended Description; Subclause 5.10.7.5.17, page 539 , line 1939 The historic 4BSD default for the scroll edit option was (LINES - 1) / 2 . System V may differ .
#72 vi Command Descriptions; Subclause 5.35.7.1.2 , page 633 , line 5086 The historic 4BSD default for the initial scroll value was (window + 1) / 2 . System V may differ . Don ' t reference 5.10.7.5.17, it had no connection with the scroll option .
#73 Command Descriptions; Subclause 5 . 10 . 7 . 2 , page 525 , line 1408 Don ' t use the phrase ''current position ' ' , match the vi wording (which refers to this).
#74 vi Command Descriptions Subclause 5 . 35 . 7 . 1 . 6 , page 634 , line 5117 There ' s no way that you can guarantee that moving the cursor won ' t erase a character from the screen . Consider what happens if the cursor is on the second logical line of a folded physical line , and the ' h ' command is moving back to the first logical line , and the first logical line isn ' t currently displayed . Historically , vi control - H worked in command mode , too , as did the stty erase character .
#75 vi Command Descriptions; Subclause 5 . 35 . 7 . 1 . 19 , page 637 , line 5238 This subclause has the same problems that the ex version of this command had . Point the vi text at the ex text .
#76 vi Command Descriptions; Subclause 5 . 35 . 7 . 1 . 21 , page 638 , line 5263 Historically , unless the cursor was on a special character , only the current line forwards was checked for a special character . Historic practice was to search for ( ) , {} and [] , not just ( ) and {} .
#77 vi Command Descriptions; Subclause 5 . 35 . 7 . 1 . 22 , page 638 , line 5283 Repeat substitution did not historically change the column .
#78 Command Descriptions; Subclause 5 . 10 . 7 . 2 . 15 , page 527 , line 1479 Historically , users could specify the marks ' and ' , and then ' ' would move to that mark . Permit historic practice . Marks were not historically cursor positions , they were locations in the file defined by line number and count character from the beginning of the line .
#79 Command Descriptions; Subclause 5 . 10 . 7 . 2 , page 522 , line 1309 Previous context applied to ex as well as vi; move the discussion into the ex man page .
#80 Command Descriptions; Subclause 5.10.7.2.14, page 526 , line 1459 No mapped command can ever be a visual mode dot command . Mapped commands could be undone with a single u command .
#81 Command Descriptions; Subclause 5.10.7.1 Historically , if the only thing on the line is an address , vi simply moves the cursor , e . g . ": 3 #60; CR #62; " does nothing more than move the cursor . If there are any non #60; blank #62; s after the address , e . g . in ": 3| #60; CR #62; " , the default ex command is executed as well .
#82 global; Subclause 5.10.7.2.10, page 525 , line 1414 Historically , "global ! " was the same as "v" . You can do multiple commands to a global command by using vertical bar separators , too . Historically , an empty pattern to a global command used the last BRE .
#83 abbrev; Subclause 5.10.7.2.1, page 523 , line 1328 The real name of the abbrev command is abbreviate , and of unabbrev is unabbreviate .
#84 open; Subclause 5 . 10 . 7 . 2 . 19 , page 528 , line 1511 The pattern argument to the open command is optional .

#85 Options; Subclause 5.10.3 , page 515 , line 1009 Historically , if there was no recovery information , it was as if the - r was not specified , so users could enter "vi - r * . c" and recover any files that needed to be recovered . (This didn ` t really work in some implementations , but the intent was clear . .
#86 Output Files; Subclause 5.10.6.3 , page 519 , line 1136 The trailing #60; newline #62; should be written every time the line is written , whether to a file , a utility or the screen. Otherwise , it gets messy , since piping the file to another utility may have a different result than writing the file out and then pointing the utility at the file . The easy way to do this is to specify that the input file has a newline at the end .
#87 Ex; Clause 5.10 , page 515 , line 980 There ` s no discussion in ex of command line or input line editing characters , e.g. control - W, control - H, etc. This is probably a separate section of text.

Interpretation Response
#1: The standard states the behavior for input in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#2 paragraph 1: The standard states the behavior for selection of a UID in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#2 paragraph 2: The standard states the behavior for operations after clearing the edit buffer in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#2 paragraph 3: 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.
#3: The standard states the behavior for the - r and - t flag in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#4: The standard states the behavior for the order of execution in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#5: The standard does not speak to this issue, and as such no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.
#6: 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.
#7: The standard states the behavior for SIGHUP and SIGINT in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#8: The standard states the behavior for alternate pathname in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#9: The standard does not speak to this issue, and as such no conformance distinction can be made between alternative implementations based on this. This is being referred

to the sponsor.
#10: The standard does not speak to this issue, and as such no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.
#11: The standard states the behavior for buffer in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#12: The standard states the behavior for input modes in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#13: The standard states the behavior for insert in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#14: The standard states the behavior for input modes in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#15: The standard states the behavior for motion in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#16: The standard states the behavior for shell word expansions in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#17: The standard states the behavior for abbrev in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#18: The standard states the behavior for append in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#19: The standard states the behavior for change in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#20: The standard states the behavior for chdir in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#21: The standard states the behavior for copy in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#22: The standard states the behavior for edit in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#23: The standard states the behavior for global in ex, and conforming implementations must conform to this . However , concerns have been raised about this which are being referred to the sponsor.
#24: The standard states the behavior for join in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being

referred to the sponsor.
#25: The standard states the behavior for map in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#26: The standard states the behavior for move in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#27: The standard states the behavior for list in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#28: The standard states the behavior for open in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#29: The standard states the behavior for put in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#30: The standard states the behavior for read in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#31: The standard states the behavior for recover in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#32: The standard states the behavior for rewind in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#33: The standard states the behavior for set in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#34: The standard states the behavior for shell in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#35: The standard states the behavior for substitute in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#36: The standard states the behavior for tags in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#37: The standard states the behavior for suspend in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#38: The standard states the behavior for visual in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#39: The standard states the behavior for write in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being

referred to the sponsor.
#40: The standard states the behavior for xit in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#41: The standard states the behavior for adjust window in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#42: The standard states the behavior for escape in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#43: The standard states the behavior for shift left in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#44: The standard states the behavior for shift right in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#45: The standard states the behavior for scroll in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#46: The standard states the behavior for execute in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#47: The standard clearly states the behaviour for delete in ex, and conforming implementations must conform to this . #48: The standard clearly states the behaviour for regular expressions in ex, and conforming implementations must conform to this . #49: The standard clearly states the behaviour for autoindent in ex, and conforming implementations must conform to this . #50: The standard states the behavior for list in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#51: The standard states the behavior for paragraphs in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#52: The standard states the behavior for sections in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#53: The standard does not speak to this issue, and as such no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.
#54: The standard states the behavior for file pattern expansion in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#55: The standard clearly states the behaviour for shell in ex, and conforming implementations must conform to this . #56: 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.

#57: 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.
#58: The standard states the behavior for tags in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#59: The standard clearly states the behaviour for warn in ex, and conforming implementations must conform to this . #60: The standard states the behavior for window in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#61: The standard states the behavior for wrapscan in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#62: The standard states the behavior for wrapmargin in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#63: The standard states the behavior for writeany in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#64: The standard does not speak to this issue, and as such no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.
#65: 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.
#66: The standard states the behavior for interrupt in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#67: The standard states the behavior for displaying long lines in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#68: The standard states the behavior for lines and columns in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#69: The standard states the behavior for cursor positioning in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#70: The standard states the definition for paragraphs, sections and sentences in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#71: The standard states the behavior for scroll in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#72: The standard states the behavior for scroll forwards in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#73: 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.
#74: The standard states the behavior for moving the cursor backward in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#75: The standard states the behavior for scroll forwards in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#76: The standard states the behavior for matching special characters in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#77: The standard states the behavior for repeat substitution in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#78: The standard states the behavior for marks in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#79: The standard states the behavior for count and range in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#80: The standard states the behavior for map in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#81: The standard states the behavior for addressing in vi, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#82: The standard states the behavior for global in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#83: The standard states the behavior for abbrev in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#84: The standard states the behavior for open in ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.
#85: The standard states the behavior for recovery in ex, and conforming implementations must conform to this. However. concerns have been raised about this which are being referred to the sponsor.
#86: The standard states the behavior for writing to an output file in ex, and conforming implementations must conform to this. However , concerns have been raised about this which are being referred to the sponsor.
#87: The standard states the behavior ex, and conforming implementations must conform to this. However, concerns have been raised about this which are being referred to the sponsor.

Rationale for Interpretation: None.