IEEE P1003.2a Draft 8 – December 1991

5.10.7.3.)

(6) A RE enclosed in question marks (`?`) addresses the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. The second question mark can be omitted at the end of a command line. If the `wrapscan` option is set, the search shall wrap around from the beginning of the buffer to the end of the buffer and continue up to and including the current line, so that the entire buffer is searched. (See 5.10.7.3.)

(7) An address followed by a plus sign (`+`) or a minus sign (`–`) followed by a decimal number is an offset address, and refers to the first address plus (respectively minus) the indicated number of lines. If the address is omitted, the addition or subtraction shall be taken with respect to the current line.

(8) An address of `+` or `–` followed by a number shall be taken with respect to the current line number; for example, `–5` is understood to mean `.–5`.

(9) An address ending with `+` or `–` shall have 1 added to or subtracted from the address, respectively. As a consequence of this rule and of rule (8) above, the address `–` refers to the line preceding the current line. Moreover, trailing `+` and `–` characters have a cumulative effect; for example, `––` refers to the current line less 2.

(10) A percent sign (`%`) shall stand for the address pair `1,$`.

Commands require zero, one, or two addresses. See the descriptions of *line* and 7
range in 5.10.7.2. Commands that require zero addresses shall regard the pres- 7
ence of an address as an error. 7

Adjacent addresses in a *range* shall be separated from each other by a comma (`,`) 7
or a semicolon (`;`). In the latter case, the current line (`.`) shall be set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches [see rules (5) and (6) above]. The second address of any two-address sequence shall correspond to a line that follows, in the buffer, the line corresponding to the first address. The first address shall be less than or equal to the second address. The first address shall be greater than or equal to the first line of the editing buffer and the last address shall be less than or equal to the last line of the editing buffer. Any other case shall be an error.

5.10.7.2 Command Descriptions

The following symbols are used in this subclause to represent optional modifiers. Any or all can be omitted; the defaults are shown.

line A single line address, given in any of the forms described in 5.10.7.1; the default for *line* shall be the current line.

range A *line* , or a pair of line addresses, separated by a comma or semicolon (see 5.10.7.1 for the difference between the two); the default for *range* shall be the current line only (`.,.`). A percent sign (%) shall stand for the range `(1,$)`. If the range specified is such that the starting address exceeds the ending address, the range is invalid and the command shall not be performed. If more than the expected number of addresses are given in a range, the greatest valid number of the last one(s) given shall be used. For example, `1,3,5p` prints lines 3 through 5 (because two is the greatest valid number in the range accepted by `print`).

count A positive integer, specifying the number of lines to be affected by the command; the default for *count* shall be 1.

flags One of the characters `#`, `p`, or `l` (ell), or both `#` and `l` to add numbers to list-format output. When a command with such a flag completes, the addressed line(s) shall be written out as if by the corresponding `#`, `p`, or `l` command. The use of *flags* shall apply to all lines written by the `list`, `number`, `open`, `print`, `substitute`, `visual`, &, and `z` commands; for other commands, it shall apply to the current line at the completion of the command. In addition, any number of `+` or `–` characters can also be given after the flags, in which case the line written shall not be the one affected by the command, but rather the line addressed by the offset address as described above. The default for *flags* shall be null.

buffer One of a number of named areas for saving text. The named buffers are specified by the lowercase letters of the POSIX Locale. Specifying *buffer* shall cause the area of text affected by the command to be stored into the buffer as it was before the command took effect. This argument is also used on the `put` command and the visual mode ''put'' commands (`p` and `P`) to specify the buffer that shall provide the text to insert.

If the buffer name is specified in uppercase, and the buffer is to be modified (as with a deletion or yanking command), the buffer shall be appended to rather than being overwritten. If the buffer is not to be modified (as in a visual mode ''put'' commands) the buffer name can be specified in lowercase or uppercase with the same results. There shall be also one unnamed buffer, which is the repository for all text deleted (with the `delete` or visual mode `d` command) or yanked (with the `yank` or visual mode `y` command) when no buffer is specified.

There are also numbered buffers, 1 through 9, which shall be accessible only from visual mode. These buffers are special in that, in visual mode, when deleted text is placed in the unnamed buffer, it also shall be placed in buffer 1, the previous contents of buffer 1 shall be placed in buffer 2, and so on. Any text in buffer 9 shall be lost. Text that is yanked (or otherwise copied) into the unnamed buffer shall not modify the numbered buffers. Text cannot be placed directly into the numbered buffers although it can be retrieved from them by using a visual mode ''put'' command with the buffer name given as a number. When the *buffer* modifier is not used in the commands below, the unnamed buffer shall be the default.

file A pattern used to derive a pathname; the default shall be the current file, as defined above. If no current file has yet been established, a warning shall be written and the command shall be aborted, except where specifically noted in the individual command descriptions that follow. The pattern shall be subjected to the process of shell word expansions (see 3.6); if more than a single pathname results and the command is expecting one file, the effects are unspecified.

word In the POSIX Locale, a *word* consists of a maximal sequence of letters, digits, and underscores, delimited at both ends by characters other than letters, digits, or underscores, or by the beginning or end of a line or the file.

! A character that can be appended to the command to modify its operation, as detailed in the individual command descriptions.

If both a *count* and a *range* are specified for a command that uses them, the number of lines affected shall be taken from the *count* value rather than the *range*. The starting line for the command shall be taken to be the first line addressed by the *range*.

When only a *line* or *range* is specified with no command, the implied command shall be either a `print`, `list`, or `number` (`p`, `l`, or `#`). The command selected shall be the last of these three commands to be used, including use as a *flag*. When no range or count is specified and the command line is a blank line, the current line shall be written, and the current line shall be set to `.+1`.

Zero or more `<blank>` characters can precede or follow the addresses, *count*, *flags*, or command name. Any object following a command name (such as *buffer*, *file*, etc.) that begins with an alphabetic character shall be separated from the command name with at least one `<blank>`.

For each of the commands listed below, the command can be entered as the abbreviation (those characters in the Synopsis command word preceding the [ ), the full command (all characters shown for the command word, omitting the [and]), or any subset of the characters of the full command down to the abbreviation. For example, the `args` command (shown as `ar[gs]` in the Synopsis) can be entered as `ar`, `arg`, or `args`.

5.10.7.2.1 abbrev

Synopsis : `ab[brev]` *word rhs*

Add the named abbreviation to the current abbreviation list. In visual mode, if *word* is typed so that it is preceded and followed by characters that cannot be part of a *word* (as defined previously), it shall be replaced by the string *rhs* .

5.10.7.2.2 append

Synopsis : [*line*] `a[ppend][!]`

Enter input mode; the input text shall be placed after the specified line. If line 0 is specified, the text shall be placed at the beginning of the buffer. The current line indicator shall be set to the last input line; if no lines are input, it shall be set to the target line, or to the first line of the file if a target of 0 was specified. Following the command name with ! shall cause the `autoindent` editor option setting to be toggled for the duration of this command only.

5.10.7.2.3 args

Synopsis : `ar[gs]`

Write the argument list with the current argument inside `[` and `]`. The argument list is the list of operands on startup, which can subsequently be replaced by the operands of the `next` command.

5.10.7.2.4 change

Synopsis : [*range*] `c[hange][!]` [*count*]

Enter input mode; the input text shall replace the specified lines (*range*). The current line indicator shall be set to the last line input; if no lines were input, it shall be set to the line before the target line, or to the first line of the file if there are no lines preceding the target. Following the command name with ! shall cause the `autoindent` editor option setting to be toggled for the duration of this command only.

5.10.7.2.5 chdir

Synopsis : `chd[ir][!]` [*directory*]
Synopsis : `cd[!]` [*directory*]

Change the current working directory to *directory* . If the current buffer has been modified since the last write, a warning shall be written and the command shall fail. This warning can be overridden by appending ! to the command name, which shall allow the command to complete.

5.10.7.2.6 copy

Synopsis : [*range*] `co[py]` *line* [*flags*]
Synopsis : [*range*] `t` *line* [*flags*]

Place a copy of the specified lines (*range*) after the specified destination *line* ; line 0 specifies that the lines shall be placed at the beginning of the buffer.

5.10.7.2.7 **delete**

Synopsis : [*range*] `d[elete]` [*buffer*] [*count*] [*flags*] 7

Delete the specified lines from the buffer. If a named *buffer* is specified, the deleted text shall be saved in it; otherwise, the deleted text shall be saved in the
unnamed buffer. If the command name is followed by a letter that could be inter- 7
preted as either a buffer name or a *flag* value (because neither a *count* nor an 7
additional *flags* value was given), `ex` shall consider the letter to be a *flags* value if 7
the letter directly follows the command name, without any `<blank>` separation; if 7
the letter is preceded by `<blank>`(s), it shall be considered a buffer name. 7

For example: 7

`1dp` or `1deletep`	Deletes the first line and prints the line that was second.	8
`1d p`	Deletes the first line, saving it in buffer `p`.	8
`1d p1l`	(Pee-one-ell.) Deletes the first line, saving it in buffer `p`, and listing the line that was second.	8

The current line indicator shall be set to the line following the deleted lines, or to the last line if the deleted lines were at the end.

5.10.7.2.8 **edit**

Synopsis : `e[dit][!]` [`+`*line*] [*file*]
Synopsis : `ex[!]` [`+`*line*] [*file*]

Begin editing *file* . If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. This action can be overridden by appending the character `!` to the command name (`e!` *file*). The current line indicator shall be the last line of the buffer; however, if this command is executed from within visual mode, the current line indicator shall be the first line of the buffer. If the `+`*line* option is specified, the current line indicator shall be set to the specified position, where *line* can be a number (or `$`) or specified as "`/`*pattern* " or "`?`*pattern* ". Preceding the pattern with a `/` shall start a search from the beginning of the file. Preceding the pattern with a `?` shall start a search from the end of the file. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 for details.

5.10.7.2.9 **file**

Synopsis : `f[ile]` [*file*]

Write the current pathname, the number of lines, the current position when no *file* argument has been specified; `ex` may write other unspecified information. If no current file has yet been established, an unspecified message shall be written to indicate that no file is being edited. With *file* , `ex` shall change the current file name to *file* , without changing the contents of the buffer or the previous current file.

5.10.7.2.10 `global`

Synopsis : [*range*] `g[lobal]` /*pattern* / [*cmds*] 7
Synopsis : [*range*] `v` /*pattern* / [*cmds*] 7

Mark the lines within the given range that match (`g`) or do not match (`v`) the given pattern. Then execute the `ex` command(s) given by *cmds* with the current line (`.`) set to each marked line.

Multiple *cmds* can be specified, one per line, by escaping each `<newline>` with a backslash. If *cmds* are omitted, each line shall be written. For the `append`, `change`, and `insert` commands, the input text shall be included as part of the `global` command line; in this case the terminating period can be omitted if it ends *cmds* . The `visual` command can be specified as one of the *cmds* . In this mode, input shall be taken from the terminal. Entering a `Q` from visual mode shall cause the next line matching the pattern to be selected and visual mode to be reentered, until the list is exhausted.

The `global` command itself and the `undo` command cannot be used in *cmds* . The editor options `autoprint`, `autoindent`, and `report` shall be inhibited for the duration of the `g` or `v` command.

5.10.7.2.11 `insert`

Synopsis : [*line*] `i[nsert][!]`

Enter input mode; the input text shall be placed before the specified line. The 7
current line indicator shall be set to the last line input; if no lines were input, it 7
shall be set to the line before the target line, or to the first line of the file if there 7
are no lines preceding the target. Following the command name with the charac- 7
ter ! shall cause the `autoindent` editor option setting to be toggled for the dura- 8
tion of this command only. 8

5.10.7.2.12 `join`

Synopsis : [*range*] `j[oin][!]` [*count*] [*flags*]

Join the text from the specified lines together into one line. In the POSIX Locale, when the last character on the first line of a pair of lines to be joined is a period, two `<space>`s shall be added following the period; when the last character of the first line is a `<blank>` or when the first character on the second line of the pair is a `)`, no `<space>`s shall be added; otherwise, one `<space>` shall be added following the last character of the first line. Extra `<blank>`s at the start of a line shall be discarded.

Appending a character ! to the `join` command name shall cause a simpler join with no white-space processing, independent of the current locale.

5.10.7.2.13 `list`

Synopsis : [*range*] `l[ist]` [*count*] [*flags*]

Write the addressed lines in a way that should be unambiguous: nonprintable
characters shall be written as implementation-defined multicharacter sequences; 7
the end of the line shall be marked with a `$`.

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. The only useful flag is #, for line numbers. The current line indicator shall be set to the last line written.

5.10.7.2.14 `map`

Synopsis : `map[!]` [*x rhs*]

Define macros for use in visual mode. The first argument shall be a single character. When this character is typed in visual mode, the action shall be as if the corresponding *rhs* had been typed. If the character ! is appended to the command name `map`, the mapping shall be effective during input mode rather than com-
mand mode. This allows *x* to have two different macro definitions at the same 7
time: one for command mode and one for input mode. Nonprintable characters, 7
except for `<tab>`, require escaping with a `control-V` (or `control-Q`) to be 8
entered in the arguments. On certain block-mode terminals, the mapping need not occur immediately (for example, it may occur after the terminal transmits a group of characters to the system), but it shall achieve the same results of modifying the file as if it occurred immediately. Implementations may restrict the set of commands accepted within *rhs*; the list of restrictions is implementation defined.

The `map` command with no arguments shall write all of the macros currently defined. If ! is appended to the command, only the macros effective during input mode shall be written; otherwise, only the macros effective during command mode shall be written.

5.10.7.2.15 `mark`

Synopsis : [*line*] `ma[rk]` *x*

Synopsis : [*line*] `k` *x*

Give the specified line the specified mark *x*, which shall be a single lowercase letter of the POSIX Locale. The current line position shall not be affected. The expression `'`*x* can then be used as an address in any command requiring one. For example ``` `.,'``` *x*`d`'' shall delete all the lines from the current one to the marked line. Also see 5.35.7.1.23 and 5.35.7.1.24 for uses of the mark in visual mode. If the `'`*x* command is used in nonvisual mode, the character marked shall be the first `<non->`blank character of the current line. Otherwise, the character marked shall be the character at the current column of the current line.

5.10.7.2.16 **move**

Synopsis : [*range*] m[ove] *line*

Move the specified lines (*range*) to be after the target line (*line*). The current line indicator shall be set to the first of the moved lines.

5.10.7.2.17 **next**

Synopsis : n[ext][!] [*file* . . .]

Edit the next file from the argument list. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. This action can be overridden by appending the character ! to the command name (n!). The argument list can be replaced by specifying a new one as operands to this command. Editing then starts with the first file on this new list. The current line indicator shall be reset as described for the edit command. This command shall be affected by the editor options autowrite and writeany; see 5.10.7.5 for details.

5.10.7.2.18 **number**

Synopsis : [*range*] nu[mber] [*count*] [*flags*]
Synopsis : [*range*] # [*count*] [*flags*]

Write the selected lines, each preceded with its line number in decimal. Nonprintable characters, except for <tab>, shall be expanded as specified by the 7
print command. See 5.10.7.2.21. The format shall be as follows:

"%+6d\t%s", *<line number>* , *<line text>*

The only meaningful flag is l, which shall cause the additional expanded writing of <tab>s and end-of-lines done by the list command to be performed. The current line indicator shall be set to the last line written.

5.10.7.2.19 **open**

Synopsis : [*line*] o[pen] /*pattern* / [*flags*]

Enter open mode, which shall be equivalent to visual mode with a one-line window. All the visual mode commands shall be available. If a match is found for the optional regular expression in *line* , the cursor shall be placed at the start of the matching pattern. The visual mode command Q (see 5.35.7.1.71) shall exit open mode. This command need not be supported on block-mode terminals. 7

5.10.7.2.20 **preserve**

Synopsis : pre[serve]

Save the current buffer in a form that can later be recovered by using ex −r or by using the recover command. After the file has been preserved, a mail message shall be sent to the user. This message shall be readable by invoking the mailx utility (see 4.40). The message shall contain the name of the file, the time of preservation, and an `ex` command that could be used to recover the file. Additional information may be included in the mail message.

5.10.7.2.21 **print**

Synopsis : [*range*] `p[rint]` [*count*] [*flags*]

Write the addressed lines. Nonprintable characters, except for `<tab>`, shall be 7
written as implementation-defined multicharacter sequences.

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. The only meaningful *flags* are # and `l`. The current line indicator shall be set to the last line written.

5.10.7.2.22 **put**

Synopsis : [*line*] `pu[t]` [*buffer*]

Put back deleted or ''yanked'' lines after line *line* . A buffer can be specified; otherwise, the text in the unnamed buffer (where deleted or yanked text is placed by default) shall be restored. The current line indicator shall be set to the first line put back.

5.10.7.2.23 **quit**

Synopsis : `q[uit][!]`

Terminate the editing session. If the current buffer has been modified since the last write, a warning shall be written and the command shall fail. This warning can be overridden, and an exit forced, discarding changes, by appending the character ! to the command name.

5.10.7.2.24 **read**

Synopsis : [*line*] `r[ead][!]` [*file*]

Place a copy of the specified file in the current buffer after the target line (which can be line 0 to place text at the beginning). If no *file* is named, the current file shall be the default. If there is no current file, then *file* shall become the current file. If there is no current file nor *file* operand, the command shall fail.

The current line indicator shall be set to the last line read. In visual mode, the current line indicator shall be set to the first line read. If *file* is preceded by !, *file* shall be taken to be an operating system command and passed to the program named in the **SHELL** environment variable; the resultant output shall be read in to the buffer. The special meaning of ! can be overridden by escaping it with a backslash character.

5.10.7.2.25 **recover**

Synopsis : `rec[over]` *file*

Attempt to recover *file* if it was saved as the result of a `preserve` command, the receipt of a signal (see 5.10.5.4), or a system or editor crash. The current line indicator shall be reset as described for the `read` editor command.

5.10.7.2.26 `rewind`

Synopsis : `rew[ind][!]`

Rewind the argument list; that is, the current file shall be set to the first file in the argument list. This shall be equivalent to a `next` command with the current argument list as its operands. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. The action can be overridden by appending the character `!` to the command name (`rew!`). The current line indicator shall be reset as described for the `read` editor command. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 for details.

5.10.7.2.27 `set`

Synopsis : `se[t]` [*option* [=[*value*]] . . .] [`no`*option* . . .] [*option* `?` . . .] [`all`] 7

When no arguments are specified, write those options whose values have been 7
changed from the default settings; when the argument `all` is specified, write all 7
of the option values. 7

Giving an option name followed by the character `?` shall cause the current value of that option to be written. The `?` can be separated from the option name by zero or more `<blank>`s. The `?` shall be necessary only for Boolean valued options. Boolean options can be given values by the form `se` *option* to turn them on or `se no`*option* to turn them off; string and numeric options can be assigned by the form `se` *option* =*value* . Any `<blank>`s in strings can be included as is by preceding
each such character with a backslash. More than one option can be set or listed 7
by a single `set` command by specifying multiple arguments, each separated from 7
the next by one or more `<blank>`s.

See 5.10.7.5 for further details about options.

5.10.7.2.28 `shell`

Synopsis : `sh[ell]`

Invoke the program named in the **SHELL** environment variable with the argument `-i` (interactive mode). Editing shall be resumed when the program exits.

5.10.7.2.29 `source`

Synopsis : `so[urce]` *file*

Read and execute commands from the file specified by the mandatory argument *file*. Such `so` commands can be nested. The maximum supported nesting depth is implementation defined, but shall be at least one.

5.10.7.2.30 `substitute`

Synopsis : [*range*] `s[ubstitute]` [/*pattern*/*repl*/[*options*] [*count*] [*flags*]] 7

Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See 5.10.7.3 and 5.10.7.4.) If the /*pattern*/*repl*/ argument is not present, the /*pattern*/*repl*/ from the previous `substitute` command shall be used. If *options* includes the letter `g` (global), all nonoverlapping instances of the pattern in the line shall be substituted. If the option letter `c` (confirm) is included, then before each substitution the line shall be written with ^ characters written on the following line, adjacent to and identifying the pattern to be replaced; an affirmative response shall cause the substitution to be done, while any other input aborts it. An affirmative response shall consist of a line with the affirmative response (as defined by the current locale) at the beginning of the line. Such a line shall be subject to editing in the same way as the command line (the `/` or `:` line at the bottom of the screen). The current line indicator shall be set to the last line substituted. When the `c` option is used, typing the interrupt character or receiving the SIGINT signal shall stop the substitute operation and `ex` shall return to command mode. All substitutions completed before the interrupt occurred shall be retained, and none shall be made after that point. The current line indicator shall be set to the last line substituted. This command shall be affected by the **LC_MESSAGES** environment variable and the `wrapscan` option.

5.10.7.2.31 `suspend`

Synopsis : `su[spend][!]`
Synopsis : `st[op][!]`

Allow control to return to the invoking process; `ex` shall suspend itself as if it had received the SIGTSTP signal. The suspension shall occur only if the system supports job control as described in POSIX.1 {8} and job control is enabled in the invoking shell (see the description of `set −m` in 3.14.11).

If job control is not supported or is disabled in the invoking shell, issuing either the `suspend` or `stop` command shall produce a warning message, but otherwise shall have no effect. Following either `suspend` or `stop` with the character `!` shall affect the operation of the `autowrite` editor option for this command only.

The current *susp* character (see `stty` in 4.59) shall also cause the suspension.

5.10.7.2.32 `tag`

Synopsis : `ta[g][!]` *tagstring*

Search for the *tagstring*, which can be in a different file. If the tag is in a different file, then the new file shall be opened for editing. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. The action can be overridden by appending the character `!` to the command name. The current line indicator shall be reset to the line indicated by the tag. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 and 5.7 (`ctags`) for details. This command shall be affected by the `tags` editor option.

The `tag` command shall search for *tagstring* in the tag file referred to by the `tags` editor option until a reference to *tagstring* is found. The file pointed to by this reference shall be loaded into the buffer, and the current line shall be set to the first occurrence of the pattern specified in the tags file associated with the supplied *tagstring* ; if the tags file contained a line-number reference, the current line shall be set to that line. If the pattern or line number is not found, an error message shall be written. If a file referred to by the `tags` editor option does not exist or is not readable, an error message shall be written. The results are unspecified if the format of a tags file is not as specified by the `ctags` utility description.

5.10.7.2.33 unabbrev

Synopsis : `una[bbrev]` *word*

Delete *word* from the list of abbreviations, as described by the `abbrev` editor command in 5.10.7.2.1.

5.10.7.2.34 undo

Synopsis : `u[ndo]`

Reverse the changes made by the previous editing command (one that changes the contents of the buffer). For this purpose, `global` and `visual` are considered single commands. An `undo` can itself be reversed. Commands that affect the external environment, such as `write`, `edit`, and `next` cannot be undone.

5.10.7.2.35 unmap

Synopsis : `unm[ap][!]` *x*

If no ! is specified, remove the command-mode macro definition for *x*; otherwise, remove the input-mode macro definition for *x*. See 5.10.7.2.14.

5.10.7.2.36 visual

Synopsis : [*line*] `vi[sual]` [*type*] [*count*] [*flags*]

Enter visual mode with the current line indicator set to *line* . The *type* is optional, and can be −, `.`, `+`, or `^`, as in the `z` command, to specify the position of the specified line on the screen window. (The default shall be to place the line at the top of the screen window.) A *count* specifies the number of lines that will initially be written; the default shall be the value of the editor option `window`. The command `Q` shall exit visual mode. (For more information, see `vi` in 5.35). This command need not be supported on block-mode terminals.

5.10.7.2.37 write

Synopsis : [*range*] `w[rite][!] [>>]` [*file*]
Synopsis : [*range*] `w[rite] [!]` [*file*]
Synopsis : [*range*] `wq[!] [>>]` [*file*]

Write the specified lines (the whole buffer, if no *range* is given) out to the file represented by the pathname *file* , writing to standard output the number of lines and bytes written.

If *file* is specified and is not the current file, and the file named by *file* exists, then the write shall fail. If the current file has been changed by the `file` command, and that file exists, the write shall fail. In either case, the write can be forced by appending the character `!` to the command name. An existing file can be appended to by appending `>>` to the command name. If the file does not exist, the result is implementation defined. 7

If the *file* is preceded by `!`, the program named in the **SHELL** environment variable shall be invoked with *file* as its second argument, and the specified lines shall be passed as standard input to the command. The `!` in this usage shall be separated from the `w` command by at least one `<blank>` character. The special meaning of the `!` can be overridden by escaping it with a backslash character. This command shall be affected by the editor options `writeany` and `readonly`.

The command `wq` shall be equivalent to a `w` followed by a `q`; `wq!` shall be equivalent to `w!` followed by `q`. If the current buffer has no pathname associated with it, the `write` command shall fail.

5.10.7.2.38 xit

Synopsis : [*range*] `x[it][!]` [*file*]

Perform a `write` command (see 5.10.7.2.37) if any changes have been made to the current buffer since the last write to any file.

Unless the command fails because an attempt to write lines to a file did not succeed, the `ex` utility shall exit after an `x` command. This command shall be affected by the editor options `writeany` and `readonly`.

5.10.7.2.39 yank

Synopsis : [*range*] `ya[nk]` [*buffer*] [*count*]

Place the specified lines in the named buffer. If no buffer is specified, the unnamed buffer shall be used (where the most recently deleted or yanked text is placed by default).

5.10.7.2.40 Adjust Window

Synopsis : [*line*] `z` [*type*] [*count*] [*flags*]

If *type* is omitted, then *count* lines following the specified line (*line*) shall be written. The default for *count* shall be the value of the editor option `window`. The *type* argument shall change the position at which *line* shall be written on the screen by affecting the number of lines written before and after *line* .

If *type* is specified, it shall be one of the following:

- `–` Place *line* at the bottom of the screen.
- `+` Place *line* at the top of the screen.

- `.` Place *line* in the middle.
- `^` Write out *count* lines starting *count* *2 lines before the addressed line; the net effect of this shall be that a `z^` command following another `z` command writes the previous page.
- `=` Center the addressed line on the screen with a line of hyphens written immediately before and after it. The number of preceding and following lines of text written shall be reduced to account for these lines of hyphens.

In all cases the current line indicator shall be set to the last line written, with the exception of the `=` type, which shall cause the current line indicator to be set to that addressed in the command.

5.10.7.2.41 Escape

Synopsis : `!` *command*

Synopsis : [*range*]`!` *command*

Pass the remainder of the line after the `!` character to the program named in the **SHELL** environment variable for execution. A warning shall be issued if the buffer has been changed since the last write. A single `!` character shall be written when the command completes. The current line position shall not be affected.

Within the text of *command* , `%` and `#` shall be expanded as pathnames (the current and alternate pathnames, respectively), and `!` shall be replaced with the text of the previous `!` command. (Thus, `!!` shall repeat the previous `!` command.) If any such expansion is done, the expanded line shall be echoed.

The special meanings of `%`, `#`, and `!` can be overridden by escaping them with a backslash character. This command shall be affected by the editor options `autowrite` and `writeany`; see 5.10.7.5 for details.

In the second form of the `!` command, the remainder of the line after the `!` shall be passed to the program named in the **SHELL** environment variable, as described above. The specified lines shall be provided to the program as standard input; the resulting output shall replace the specified lines.

5.10.7.2.42 Shift Left

Synopsis : [*range*] `<` [*count*] [*flags*]

Shift the specified lines to the left; the number of character positions to be shifted shall be determined by the editor option `shiftwidth`. Only leading `<blank>`s shall be lost in shifting; other characters shall not be affected. The current line indicator shall be set to the last line changed.

5.10.7.2.43 Shift Right

Synopsis : [*range*] `>` [*count*] [*flags*]

Shift the specified lines to the right, by inserting `<blank>`s, using `<tab>`s where possible, as determined by the editor option `shiftwidth`. Empty lines shall not 7 be changed. The current line indicator shall be set to the last line changed. 7

5.10.7.2.44 Resubstitute

Synopsis : [*range*] & [*options*] [*count*] [*flags*]
Synopsis : [*range*] s[`ubstitute`] [*options*] [*count*] [*flags*]
Synopsis : [*range*] ~ [*options*] [*count*] [*flags*] 8

Repeat the previous substitute command, as if & were replaced by the previous
`s/`*pattern* `/`*repl* `/` command. (The same effect can obtained by omitting the
`/`*pattern* `/`*repl* `/` string in the `substitute` command.) The version of the com- 8
mand using tilde shall be the same as & and `s`, but the *pattern* used shall be the 8
last regular expression used in any command, not necessarily the one used in the 8
last substitute command. For example, in the sequence 8

```
s/red/blue/
/green
~
```
8

the ~ is equivalent to 8

```
s/green/blue/
```
8

5.10.7.2.45 Scroll

Synopsis : *eof*

Write the next *n* lines, where *n* is the value of the editor option `scroll`. The command is invoked with the *eof* character (see the description of the `stty eof` character in 4.59). The current line indicator shall be set to the last line written. This command need not be supported on block-mode terminals.

5.10.7.2.46 Write Line Number

Synopsis : [*line*] = [*flags*]

Write the line number of the specified line (default last line). The current line position shall not be affected.

5.10.7.2.47 Execute

Synopsis : @ *buffer*
Synopsis : * *buffer*

Execute each line of the named buffer as an `ex` command. If no buffer is specified or is specified as @ or *, the last buffer executed shall be used. If there is no last buffer, an error occurs.

5.10.7.3 Regular Expressions

The `ex` utility shall support the basic regular expressions described in 2.8.3. A null RE (`//`) shall be equivalent to the last RE encountered.

Regular expressions can be used in addresses to specify lines and, in some commands (for example, the `substitute` command), to specify portions of a line to be substituted.

The following constructs can be used to enhance the basic regular expressions:

- `\<` Match the beginning of a *word*. (See the definition of *word* at the beginning of 5.10.7.2.)
- `\>` Match the end of a *word*.
- ~ Match the replacement part of the last `substitute` command. The tilde (~) character can be escaped in a regular expression to become a normal character with no special meaning.

When the editor option `nomagic` is set, the only characters with special meanings shall be ^ at the beginning of a pattern, `$` at the end of a pattern, and `\`. The characters `.`, *, `[`, and ~ shall be treated as ordinary characters unless preceded by a \; when preceded by a \ they shall regain their special meaning.

5.10.7.4 Replacement Strings

The character & (\& if the editor option `nomagic` is set) in the replacement string shall stand for the text matched by the pattern to be replaced. The character ~ (\~ if `nomagic` is set) shall be replaced by the replacement part of the previous substitute command. The sequence *n*, where *n* is an integer, shall be replaced by the text matched by the pattern enclosed in the *n*th set of parentheses `\(` and `\)`.

The strings `\l`, `\u`, `\L`, and `\U` can be used to modify the case of elements in the replacement string (using the `\&` or `\`<*digit*>) notation. The string `\l` (`\u`) shall cause the first character (actually inserted by the substitution) that follows the `\l` (`\u`) to be converted to lowercase (uppercase). The strings `\L` (`\U`) shall cause all characters subsequent to them to be converted to lowercase (uppercase) as they are inserted by the substitution until the string `\e` or `\E`, or the end of the replacement string, is encountered.

In visual mode, a `control-Vcontrol-M` (or `control-Qcontrol-M`) sequence in 8
the replacement string shall be mapped to a `<newline>`, and so can be used to 7
split lines. A literal `control-M` requires escaping by preceding it with a 7
backslash (`\^V^M` or `\^Q^M`). 8

5.10.7.5 Edit Options

The `ex` utility has a number of options that modify its behavior. These options have default settings, which can be changed using the `set` command.

Options are Boolean unless otherwise specified.

5.10.7.5.1 **autoindent, ai**

[Default *off*]

If autoindent is set, each line in input mode shall be indented (using first as 7
many <tab>s as possible, as determined by the editor option tabstop, and then
using <space>s) to align with the previous line. (Starting indentation shall be
determined by the line appended after, or the line inserted before, or the first line
changed, with an a, i, or c command, respectively.) When a <newline> charac- 8
ter is inserted in the middle of a line, and when autoindent is on, the first non- 8
<blank> character to the right of the cursor shall be aligned to the current mar- 8
gin on a new line immediately following the current line. Any <blank>s to the 8
left of the cursor position at which the <newline> was entered shall be retained. 8
When autoindent is off, a new line shall be created, but no <blanks> shall be 8
discarded. Additional indentation can be provided as usual; succeeding lines
shall automatically be indented to the new alignment. A line entered during 7
input mode with autoindent that contains no user-entered characters shall be 7
empty, despite the appearance of indentation during entry. 7

Reducing the indent can be achieved when the cursor is at the current left margin 7
by typing control-D one or more times; the cursor shall be moved back to the 7
previous integral number of shiftwidth spaces for each control-D. A ^ fol-
lowed by a control-D shall remove all indentation temporarily (for the current
line); a 0 followed by a control-D shall remove all indentation permanently (for
the current line and subsequent lines until input mode is reentered or until the 7
indentation is specifically set to some other value). Changing the indent with 7
control-D need not be supported on block-mode terminals.

5.10.7.5.2 **autoprint, ap**

[Default *on*] 7

If autoprint is set, the current line shall be written after each command that
changes buffer text. (Autoprint shall be suppressed in the global [g and v] com-
mands and for any command on which print operands [*flags*] are used to write
explicitly the current line.)

5.10.7.5.3 **autowrite, aw**

[Default *off*]

If autowrite is set, when a next, rewind, tag, edit, suspend, stop, or ! command is given, the buffer shall be written (to the current file) if it has been modified. Appending the character ! to the command name for any of these commands except ! shall cause the write not to occur. If the write fails, the command shall be aborted and an error message shall be written.

5.10.7.5.4 beautify, bf

[Default *off*]

If beautify is set, all nonprintable characters, other than <tab>, <newline>, 7
and <form-feed>, shall be discarded from text read in from files. 7

5.10.7.5.5 errorbells, eb

[Default *off*]

If errorbells is set, error messages shall be preceded by an alert action. Setting this option off shall cause the user to be informed of an error even when in visual mode, but rather than using the <alert> character, an error message shall be written, using a standout mode of the terminal (such as inverse video) instead of the the normal effect of the <alert> character, when the effect of the <alert> character is to cause the terminal to ring a bell or make other sounds. The editor should place the error message in a standout mode of the terminal, such as inverse video instead of ringing the bell, when the terminal capabilities allow this.

5.10.7.5.6 exrc

[Default *off*]

If exrc is set, ex shall access any .exrc file in the current directory, as described previously. If exrc is not set, ex shall ignore any .exrc file in the current directory during initialization, unless the current directory is that named by the **HOME** variable.

5.10.7.5.7 ignorecase, ic

[Default *off*]

If ignorecase is set, characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.

5.10.7.5.8 list

[Default *off*]

If list is set, write the addressed lines in a way that should be unambiguous:
nonprintable characters shall be written as implementation-defined multicharac- 7
ter sequences; the end of the line shall be marked with a $.

5.10.7.5.9 magic

[Default *on*]

If magic is set, change the interpretation of characters in regular expressions and substitution replacement strings (see 5.10.7.3 and 5.10.7.4).

5.10.7.5.10 `mesg`

[Default *on*]

If `mesg` is set, the permission for others to use the `write` or `talk` commands to 7
write to the terminal shall be turned on while in visual mode. The shell-level 7
command `mesg n` (see 5.17) shall take precedence over any setting of the `ex mesg` 7
option; i.e., if `mesg y` was issued before `ex` started (or in a shell escape, such as 7
``:!mesg y''), the `mesg` option in `ex` can suppress incoming messages, but the 7
`mesg` option cannot enable incoming messages if `mesg n` was issued. 7

5.10.7.5.11 `number, nu`

[Default *off*]

If `number` is set, lines shall be written with line numbers, as does the `number` command.

5.10.7.5.12 `paragraphs, para`

[Default *implementation defined*]

The `paragraph` option shall define additional paragraph boundaries for the `{` and `}` commands in visual mode. The `paragraph` option can be set to a character string consisting of zero or more character pairs. The default value is implementation defined.

In the text to be edited, the character string `<newline>.`*<char-pair>*, where *<char-pair>* is one of the character pairs found in `paragraph`, shall define a paragraph boundary. For example, if

```
paragraph=LaA ##
```

then all of the following additional paragraph boundaries would be recognized:

```
<newline>.La
<newline>.A<space>
<newline>.##
```

5.10.7.5.13 `prompt`

[Default *on*]

If `prompt` is set, command mode input shall be prompted for with a colon (`:`); when unset, no prompt shall be written.

5.10.7.5.14 `readonly`

[Default *see text*]

If `readonly` is set, read-only mode shall be enabled. Writing to a different file shall be allowed in read-only mode; in addition, the write can be forced by using the character ! (see the editor command `write`). The default setting shall be *off* unless the file lacks write permission or the command-line option `−R` is used.

5.10.7.5.15 `remap`

[Default *on*]

If `remap` is set, macro translation shall allow for macros defined in terms of other macros; translation shall continue until the final product is obtained. If unset, only a one-step translation shall be done.

5.10.7.5.16 `report`

[Default 5]

The value of this option shall give the number of lines that can be changed by an editor command before a report is generated on the number of lines affected.

5.10.7.5.17 `scroll`

[Default `window`/2]

The value of this option shall determine the number of lines scrolled on a *eof* com-
mand (see 5.10.7.2.45 and the description of the `stty eof` character in 4.59). 8
Changing the value of `window` in **EXINIT**, one of the `.exrc` files, or with a `set` 8
command shall not affect the value of `scroll`. This editor option need not be 8
supported on block-mode terminals.

5.10.7.5.18 `sections`

[Default *implementation defined*]

The `sections` option shall define additional section boundaries for the `[[` and `]]` commands in visual mode. The `sections` option can be set to a character string consisting of zero or more character pairs. The default value is implementation defined.

In the text to be edited, the character string `<newline>.`*<char-pair>*, where *<char-pair>* is one of the character pairs found in `sections`, shall define a section boundary in the same manner that paragraph boundaries are defined. (See 5.10.7.5.12.)

5.10.7.5.19 `shell, sh`

[Default from the environment **SHELL**]

The value of this option can be a string representing the pathname of the shell to be invoked for the ! shell escape command, and by the `shell` command. The default shall be taken from the **SHELL** variable in the environment; see 5.10.5.3 for default values for **SHELL**.

5.10.7.5.20 `shiftwidth, sw`

[Default 8]

The value of this option shall give the width of an indentation level used during `autoindent` and by the shift commands.