IEEE P1003.2a Draft 8 – December 1991

5.34.5.3 Environment Variables

The following environment variables shall affect the execution of `uuencode`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and data).

LC_MESSAGES This variable shall determine the language in which messages should be written.

5.34.5.4 Asynchronous Events

Default.

5.34.6 External Effects

5.34.6.1 Standard Output

The standard output shall be a text file (encoded in the character set of the 7
current locale) that begins with the line: 7

`"beginΔ%sΔ%s\n"`, <*mode*>, *decode_pathname*

and ends with the line:

`"end\n"`

In both cases, the lines shall have no preceding or trailing `<blank>`s. 7

The algorithm that shall be used for lines in between `begin` and `end` takes three
octets as input and writes four characters of output by splitting the input at six- 7
bit intervals into four octets, containing data in the lower six bits only. These 7
octets shall be converted to characters by adding a value of 0x20 to each octet, so 7
that each octet is in the range 0x20–0x5f, and then it shall be assumed to 7
represent a printable character in the ISO/IEC 646 {1} encoded character set. It 7
then shall be translated into the corresponding character codes for the code set in 7
use in the current locale. (For example, the octet 0x41, representing `A`, would be 7
translated to `A` in the current code set, such as 0xc1 if it were EBCDIC.) 7

Where the bits of two octets are combined, the least significant bits of the first octet shall be shifted left and combined with the most significant bits of the second octet shifted right. Thus the three octets A, B, C are converted into the four octets:

```
0x20 + (( A >> 2                    ) & 0x3F)
0x20 + (((A << 4) | ((B >> 4) & 0xF)) & 0x3F)         7
0x20 + (((B << 2) | ((C >> 6) & 0x3)) & 0x3F)         7
0x20 + (( C                         ) & 0x3F)
```

These octets are then translated into the local character set. 7

Each encoded line shall contain a length character, equal to the number of charac- 7
ters to be decoded plus 0x20 translated to the local character set as described 7
above, followed by the encoded characters. The maximum number of octets to be 7
encoded on each line shall be 45. 7

5.34.6.2 Standard Error

Used only for diagnostic messages.

5.34.6.3 Output Files

None.

5.34.7 Extended Description

None.

5.34.8 Exit Status

The `uuencode` utility shall exit with one of the following values:

- 0 Successful completion.
- >0 An error occurred.

5.34.9 Consequences of Errors

Default.

5.34.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

The file is expanded by 35 percent (each three octets become four, plus control information) causing it to take longer to transmit.

History of Decisions Made

Since this utility is intended to create files to be used for data interchange between systems with possibly different code sets, and to represent binary data as a text file, ISO/IEC 646 {1} was chosen for a midpoint in the algorithm as a known reference point. The output from `uuencode` is a text file on the local system. If the output were in the ISO/IEC 646 {1} code set, it might not be a text file (at least because the `<newline>`s might not match), and the goal of creating a text file would be defeated. If this text file was then carried to another machine with the same code set, it would be perfectly compatible with that system's `uudecode`. If it was transmitted over a mail system or sent to a machine with a different code set, it is assumed that, as for every other text file, some translation mechanism would convert it (by the time it reached a user on the other system) into an appropriate code set. This translation only makes sense from the local code set, not if the file has been put into an ISO/IEC 646 {1} representation first. Similarly, `uuencode`d files can be placed in `pax` archives, intermixed with other text files in the same code set.

The algorithm is described in terms of 8-bit quantities, or octets. Since no byte alignment is implied, it will encode data from machines with any number of bits per byte. However, unless that encoded data is then decoded on a machine with the same number of bits per byte, the output might not be useful.

The term ''source file'' in previous drafts was changed to ''file'' to avoid any implied relationship to computer-language source code files.

5.35 `vi` — Screen-oriented (visual) display editor

This utility shall be provided on systems that both support the User Portability 7
Utilities Option and define the {POSIX2_CHAR_TERM} symbol. On other systems 7
it is optional. 7

5.35.1 Synopsis

`vi` `[−rR]` `[−c` *command* `]` `[−t` *tagstring* `]` `[−w` *size* `]` [*file* . . .]

Obsolescent Version :

`vi` `[−rR]` `[+`*command* `]` `[−t` *tagstring* `]` `[−w` *size* `]` [*file* . . .]

5.35.2 Description

The `vi` (visual) utility is a screen-oriented text editor. The user can switch back and forth between `vi` and the line editor `ex` (see 5.10) and execute `ex` commands from within `vi`.

When using `vi`, the terminal screen acts as a window into the editing buffer. Changes made to the editing buffer shall be reflected in the screen display; the position of the cursor on the screen shall indicate the position within the editing buffer.

Certain block-mode terminals do not have all the capabilities necessary to support the complete `vi` definition. When these commands cannot be supported on such terminals, this condition shall not produce an error message such as ‘‘not an editor command’’ or report a syntax error. The implementation may either accept the commands and produce results on the screen that are the result of an unsuccessful attempt to meet the requirements of this standard or report an error describing the terminal-related deficiency.

5.35.3 Options

The `vi` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except for the obsolescent +*command* ‘‘option.’’

The following options shall be supported by the implementation:

`−c` *command*
+*command* (Obsolescent.)
: Begin editing by executing the specified `ex` command-mode command(s). As with normal `ex` command-line entries, the *command* option-argument can consist of multiple `ex` commands separated by vertical-line characters (`|`). The use of commands that enter input mode in this manner produces undefined results.

`−r` Attempt to recover the named *file*s after an editor or system crash, after the editor has been terminated by a signal or after the use of a `pre` (`ex`) editor command.

If no *file* operands are given, all other options, the **EXINIT** variable, and any `.exrc` files shall be ignored; a list of all recoverable files available to the invoking user shall be written; and `vi` shall exit without reading files or processing user commands.

`−R` Set read-only mode, preventing accidental overwriting of the files. Any command that would write to a file shall require the ! suffix (see, for example, 5.10.7.2.37) to be effective in this mode.

`−t` *tagstring*
Edit the file containing the specified *tagstring* and set the initial position within the edit buffer to the point of definition of the tag. (See `ctags` in 5.7.) The tags feature represented by `−t` *tagstring* and the `ta` command (see 5.10.7.2.32) is optional. It shall be provided on any system that also provides a conforming implementation of `ctags`; otherwise, the use of `−t` produces undefined results.

`−w` *size* Set the value of the *window* editor option to *size*. See 5.10.7.5.28.

If both the `−t` *tagstring* and `−c` *command* (or the obsolescent +*command*) options are given, the `−t` *tagstring* shall be processed first; i.e., the file containing the tag is selected by `−t` and then the command is executed.

5.35.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to be edited.

5.35.5 External Influences

5.35.5.1 Standard Input

The standard input consists of a series of commands and input text, as described in 5.35.7.

5.35.5.2 Input Files

Input files shall be text files or files that would be text files except for an incomplete last line that is not longer than {LINE_MAX} − 1 bytes in length and contains no NUL characters. The editing of other forms of files may optionally be allowed by `vi` implementations.

The `.exrc` files (see 5.35.7) shall be text files consisting of `ex` commands.

By default, `vi` shall read lines from the files to be edited without interpreting any 8
of those lines as any form of editor command. 8

5.35.5.3 Environment Variables

The following environment variables shall affect the execution of `vi`:

COLUMNS This variable shall override the system-selected horizontal 7
screen size. See 2.6 for valid values and results when it is 7
unset or null. 7

EXINIT This variable shall be interpreted to contain a list of `ex` commands that shall be executed on editor startup, before reading the first file. The list can contain multiple commands by separating them using a vertical-line (`|`) character. See also 5.35.7 for more details of the initialization phase.

HOME This variable shall be interpreted as a pathname of a directory that shall be searched for an editor startup file named `.exrc`; see 5.35.7 for details.

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_COLLATE This variable shall determine the locale for character collation information in regular expressions.

LC_CTYPE This variable shall determine the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files), the behavior of character classes within regular expressions, the classification of characters as upper- or lowercase letters, the case conversion of letters, and the detection of word boundaries.

LC_MESSAGES This variable shall determine the language in which messages should be displayed.

LINES This variable shall override the system-selected vertical 7
screen size, used as the number of lines in a screenful and 7
the vertical screen size in visual mode. See 2.6 for valid 7
values and results when it is unset or null. 7

PATH This variable shall determine the search path for the shell command specified in the editor commands `shell`, `read`, and `write` and the visual-mode command `!`; see the description of command search and execution in 3.9.1.1.

SHELL This variable shall be interpreted as the preferred command-line interpreter for use in `!`, `shell`, `read`, and other commands with an operand of the form `!`*string*. For the `shell` command the program shall be invoked with the single argument `−i`, for all others it shall be invoked with the two arguments `−c` and *string*. If this variable is null or not set, the `sh` (see 4.56) utility shall be used.

TERM This variable shall be interpreted as the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.

5.35.5.4 Asynchronous Events

The following actions shall be taken upon receipt of signals:

SIGINT The current editor command shall be aborted. The editor shall prompt for another command.

SIGCONT The screen shall be refreshed (if in *visual* mode).

SIGHUP If the current buffer has changed since the last `e` or `w` command, `vi` shall attempt to save the current file in a state such that it can be recovered later by an `ex` or `vi −r` command.

The action taken for all other signals is unspecified.

5.35.6 External Effects

5.35.6.1 Standard Output

If standard output is a terminal device, it may be used for writing prompts to the user, for informational messages, and for writing lines from the file. If standard output is not a terminal device, undefined results occur.

5.35.6.2 Standard Error

Used only for diagnostic messages.

5.35.6.3 Output Files

The output from `vi` shall be text files that shall be identical to the input files if no changes have been made to those files by commands, with the exception that in all cases where a forced session termination (the `ex` command `q!`; see 5.10.7.2.23) has not been issued prior to any write of the file, a trailing `<newline>` shall be added to the last line of the file if one was not present in the input.

5.35.7 Extended Description

Only the visual mode of the editor is described in this subclause. See 5.10 for additional editing capabilities used in `vi`.

The pathname of the file being edited by `vi` is the *current* file. The text of the file shall be read into a *buffer*, and all editing changes shall be performed in this buffer; changes shall have no effect on the file until the buffer is written out explicitly. (The use of the term *buffer* does not attempt to imply a specific implementation.) Lines in the buffer may each be limited to {LINE_MAX} bytes and an error message may be displayed if the limit is exceeded during editing.

The *alternate* pathname is the name of the last file mentioned in an editor command, or the previous current pathname if the last file mentioned became the current file. When the character `%` appears in a pathname entered as part of a command argument, the character shall be replaced by the current pathname; the character `#` shall be replaced by the alternate pathname. The characters `%` and `#` can be escaped by preceding them with a backslash.

During initialization, before the first file is read or any user commands from the terminal are processed, if the environment variable **EXINIT** is set, the editor shall execute the `ex` commands contained in that variable. If the variable is not set, `vi` shall attempt to read `ex` commands from the file `$HOME/.exrc` (the file `.exrc` in the directory referred to by the **HOME** environment variable). If and only if **EXINIT** or `$HOME/.exrc` sets the editor option `exrc`, `vi` finally shall attempt to read `ex` commands from a file `.exrc` in the current directory. In the event that **EXINIT** is not set and the current directory is the user's home directory, any `.exrc` file shall only be processed once. No `.exrc` file shall be read unless it is owned by the same user ID as the effective user ID of the process. After any `.exrc` files are processed, any commands specified by the `−c` option shall be processed.

After initialization, `vi` shall be in *command* mode; *input* mode can be entered by several commands used to insert or change text. In input mode, `<ESC>` can be used to return to command mode; other uses of `<ESC>` are described in 5.35.7.1.17. The results of entering `<newline>`s in input mode are affected by the setting of the `autoindent` editor variable; see 5.10.7.5.1.

The last (bottom) line of the screen shall be used to display the input for search commands (`/` and `?`), for `ex` commands (`:`), and system commands (`!`). It shall be also used to report errors or display informational messages. The editor shall prompt for input for commands by displaying the command (`/`, `?`, `:`, or `!`) at the beginning of the last line of the screen. All subsequent characters shall be then taken as input until a line terminator is entered.

An interrupt typed during text input, or during the input of a command on the bottom line, shall terminate the input (or cancel the command) and shall return the editor to command mode. During command mode, an interrupt shall alert the terminal. In general, an alert indicates an error (such as unrecognized key).

Lines displayed on the screen containing only a tilde (~) shall indicate that the last line above them is the last line in the editing buffer (the ~ lines are past the end of the editing buffer).

There may be lines on the screen marked with an @. These indicate space on the screen not corresponding to lines in the file. This shall happen if there is space at the bottom of the screen for some text, but the next line of the buffer will not fit. (It may also happen on a terminal with limited local intelligence. In this case, these lines can be removed by entering a `control-R`, forcing the editor to refresh the screen.)

5.35.7.1 `vi` Command Descriptions

The cursor, in general, shall be placed on the current line and in the current column position as noted for each command described below. If the current line is not in the display window, then the display window shall be either scrolled or refreshed to cause the current line to be in the display window. If the screen is refreshed, the current line shall be positioned as close to the center of the display window as possible. If the current line is less than one-half window lines from the beginning of the editing buffer, the first line of the buffer shall be displayed on the first line of the display window. If the current line is less than one-half window lines from the end of the editing buffer, the remaining lines of the display window after the last line of the file shall contain only a ~ character in column position 1. If the screen is scrolled rather than refreshed, the current line shall be placed at the top of the display window if the current line is before the first line displayed. If the current line is after the last line displayed, the display window shall be scrolled up and the current line shall be placed on the last line of the display window.

As stated previously, the cursor is generally placed on the current column position of the current line. The one exception to this is if the current column position is beyond the end of the current line. In this case, the cursor shall be placed on the last character of the current line. However, the current column position value shall not be altered by this. Thus, if the current line changes to a longer line, the cursor shall be moved back out to the current column position or to the end of the line if it is shorter than the current column.

Unless otherwise specified, the commands shall be interpreted in command mode and have no special effect in input mode.

Some of the following command descriptions (such as `a`, `A`, `c`, etc.) move the cursor 7
and enter input mode. The indications of *Current line* and *Current column* are for 7
the cursor movement up to the beginning of input mode. Depending on the text 7
inserted, the cursor will generally wind up in a totally different location by the 7
time command mode is reentered. The cursor movement that accompanies the 7
transition from input mode to command mode is described in 5.35.7.1.17. 7

The following symbols are used in this section to represent arguments to commands, to describe commands, or to specify the new values of the current line and column indicators following execution of the command.

bigword A maximal sequence of nonblank characters preceded and followed by <blank> characters or the beginning or end of a line or the file.

buffer One of a number of named areas for saving text. Commands that change or delete text can be preceded by a buffer specification argument *buffer*. It is specified in the form ‘‘"*c*’’, where *c* represents the name of a buffer, one of 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’’ commands (`p` and `P`) to specify the buffer that shall provide the text to insert. When a command synopsis shows both [*buffer*] and [*count*] preceding the command letter, either can precede the other.

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.

column The (previous) value of the current column indicator.

count A positive integer used as an optional argument to most commands, either to give a size or a position (for display or movement commands), or as a repeat count (for commands that change text). This argument is optional and shall default to 1 unless otherwise noted in the individual command description.

end-of-line A description of the current column indicator value meaning that the current column indicator shall be set always to indicate the last character of the current line. Until the current column indicator is changed, the cursor shall be always positioned to the last character of whatever line is current.

line The (previous) value of the current line indicator.

nonblank A description of the current column indicator value meaning that the current column indicator shall be set to the position of the first nonblank character of the current line. If the current line has no nonblank characters, the indicator shall be set to the position of the last character of the line.

motion A command used as an optional trailing argument to some commands, indicating the extent of text to be affected by the command. The *motion* argument can be either the command character repeated or a cursor movement command. In the former case, exactly the current line shall be affected. In the latter case, the region specified shall be from the current cursor position to just before the cursor position indicated by the motion command. If the command operates on lines only, then all the lines that fall partly or wholly within this region shall be affected. Otherwise, the exact region as described above shall be affected. The following commands are considered to be cursor motion commands:

$	^	e	M
%	‘‘	control-F	control-N
''	‘*letter*	F	N
'*letter*	{	f	n
(	|	G	control-P
)	}	control-H	<space>
,	0	H	control-T
-	control-B	control-J	T
/	B	j	t
;	b	k	control-U
?	control-D	L	W
[[	control-E	l	w
]]	E	control-M	control-Y

The optional *count* prefix available for some of the motion commands can be included and is considered part of the *motion* argument. For example, in `c2w`, the `2w` is the *motion* argument.

previous context
A state set whenever a nonrelative motion command is executed in `vi`.

paragraph An area of text that begins with either the beginning of a file, an empty line, or section boundary and continues until either an empty line, section boundary, or the end of the editing buffer. Additional paragraph boundaries can be defined by the `paragraph` option. (See 5.10.7.5.12.)

section — An area of text that starts with a line whose first character is either a `<form-feed>` or an open brace (`{`) and continues until the next section or the end of the editing buffer. The beginning and end of the editing buffer are also considered section boundaries. Additional section boundaries can be defined by the `sections` option. (See 5.10.7.5.18.)

sentence — An area of text that begins with either the beginning of the editing buffer or the first nonblank character following the previous sentence, paragraph, or section boundary and continues until the end of the editing buffer or a period, exclamation point, or question mark (`.`, `!`, `?`) character followed by either an end of line or two `<space>`s. Any number of closing parentheses, brackets, or double-quote (`)`, `]`, `"`) characters can appear between the period, exclamation point, or question mark and the `<space>`s or end of line.

window — The current value of the editor option `window`.

word — In the POSIX Locale, `vi` recognizes two kinds of words: 7

— A maximal sequence of letters, digits and underscores, delim- 7
ited at both ends by: characters other than letters, digits, or 7
underscores; the beginning or end of a line; or the end of the 7
editing buffer. 7

— A maximal sequence of characters other than letters, digits, 7
underscores, or white space, delimited at both ends by: a 7
letter, digit, underscore, or white space; the beginning or end 7
of a line; or the end of the editing buffer. 7

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

After processing a command, if the current line and column indicators have been changed to specify a character not currently on the screen, the editor shall scroll or page the text to bring the needed character onto the screen. If the new current line is within *window* lines of the previous current line, and the terminal is capable of scrolling in the indicated direction, the text shall be scrolled the minimum number of lines necessary to present the needed character. Otherwise, the screen shall be redisplayed with the new current line positioned in the middle of the screen.

The following rules specify how exceptions shall be handled. When a movement command would cause a window to display beyond the beginning of the editing buffer, the window displayed is the window beginning with the first line of the editing buffer. When a movement command would cause a window to be displayed beyond the end of the editing buffer, the terminal shall be alerted and the command aborted. When the current line is empty, the cursor shall be placed in the first position of the line on the screen. When the current column indicator value is beyond the end of the current line, the cursor shall be placed on the last character of the current line, but the current column indicator shall be unchanged.

The following rules specify how position exceptions shall be handled. When the current column indicator is at the beginning (end) of a line and would be set to a position before the beginning (after the end) of the current line, the command shall not be executed, the terminal shall be alerted, and the current position shall remain unchanged. Otherwise, when the current column indicator would be set to a position before the beginning (after the end) of the current line, it shall be set instead to the position of the first (last) character of that line. When the current line indicator would be set to a position before the beginning (after the end) of the editing buffer, the command shall not be executed, the terminal shall be alerted, and the current position shall remain unchanged.

5.35.7.1.1 Page backwards

Synopsis : [*count*] `control-B`

Page backward in the file, allowing two lines of overlap, by displaying the window starting at line [*line* − (*count* ∗ (*window* −2))]. 7

Current line : The last line displayed.

Current column : Move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.2 Scroll forward

Synopsis : [*count*] `control-D`

Scroll forward *count* lines in the file. If *count* is not specified, the same number of lines shall be scrolled as by the previous `control-D` or `control-U` command. On the first `control-D` or `control-U` command, the amount scrolled shall be one-half the number of lines in a screenful. See 5.10.7.5.17.

Current line : (*line* + amount scrolled).

Current column : Move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.3 Scroll forward by line

Synopsis : [*count*] `control-E`

Scroll forward *count* lines, leaving the current line and column as is if possible.

Current line : Unchanged unless the current line scrolls off the screen; otherwise, move to the first line displayed.

Current column : Unchanged unless the current line scrolls off the screen; otherwise, move to the first character if it is a blank line or the last character of the line if the position of the current column is further into the line than the position of the last column of the line.

5.35.7.1.4 Page forward

Synopsis : [*count*] `control-F`

Page forward in the file, allowing two lines of overlap, by displaying the window starting at line [*line* + (*count* ∗ (*window* −2))]. 7

Current line : The first line displayed.

Current column : Move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.5 Display information

Synopsis : `control-G`

Display an informational message listing the current pathname, current line, number of lines, and other unspecified information, as does the `ex` command *file* .

Current line : Unchanged.

Current column : Unchanged.

5.35.7.1.6 Move cursor backwards

Synopsis : [*count*] `control-H`
Synopsis : [*count*] `h`

Move the cursor back *count* characters on the current line. No characters shall be erased from the screen by this command. In input mode, `control-H` shall have the exact same function.

Current line : Unchanged.

Current column : Set to (*column* − the number of columns occupied by *count* characters ending with the previous current column) or the beginning of the line if *count* is greater than the number of characters preceding the current character in the current line.

5.35.7.1.7 Move down in column

Synopsis : [*count*] `control-J`
Synopsis : [*count*] `j`
Synopsis : [*count*] `control-N`

Move the cursor down *count* lines without changing the current column.

Current line : *current line* + *count* . Unchanged if *count* is greater than the number of lines in the buffer following the current line.

Current column : Unchanged.

5.35.7.1.8 Clear and redisplay

Synopsis : `control-L`

Clear and redisplay the screen.

Current line : Unchanged.

Current column : Unchanged.

5.35.7.1.9 Move cursor down to nonblank

Synopsis : [*count*] `control-M`
Synopsis : [*count*] `+`

Move the cursor down *count* lines to the first nonblank character of that line.

Current line : *line* + *count* .

Current column : Move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.10 Move up in column

Synopsis : [*count*] `control-P`
Synopsis : [*count*] `k`

Move the cursor up *count* lines without changing the current column.

Current line : *line* − *count* .

Current column : Unchanged.

5.35.7.1.11 Redraw screen

Synopsis : `control-R`

Redraw the current screen. If any lines have been deleted from the logical screen and flagged as deleted on the terminal using the ‘‘@’’ convention (see 5.35.7), they shall be deleted, and the logical screen shall be redisplayed. Lines flagged with @ because they do not fit on the terminal display shall not be affected.

Current line : Unchanged.

Current column : Unchanged.

5.35.7.1.12 Move forward with tabs

Synopsis : `control-T`

Move the cursor forward *shiftwidth* (see 5.10.7.5.20) positions in insert mode by inserting `<tab>` characters, followed by space characters, as necessary, if the current cursor position is at the beginning of a line or preceded only by `<blank>`s.

Current line : Unchanged.

Current column : *column* + *shiftwidth* .

5.35.7.1.13 Scroll backward

Synopsis : [*count*] `control-U`

Scroll backward *count* lines in the file. If *count* is not specified, the same number of lines shall be scrolled as by the previous `control-D` or `control-U` command. On the first `control-D` or `control-U` command, the amount scrolled shall be the value of the `scroll` editor option; see 5.10.7.5.17.

Current line : (*line* − amount scrolled).

Current column : Move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.14 Escape next character

Synopsis : `control-V` *character*
Synopsis : `control-Q` *character* 8

Allow the entry of a subsequent *character* , removing any special meaning to the editor it has in input mode (e.g., `<ESC>`). The character ˆ shall be displayed in the current cursor position until the subsequent character is typed, which then replaces the character ˆ.

Current line : Unchanged.

Current column : Unchanged.

5.35.7.1.15 Delete word

Synopsis : `control-W`

Delete the word preceding the cursor (including any `<blank>`s between the end of the word and the current cursor position) in input mode by moving the cursor back to the beginning of the word. No characters shall be erased from the screen by this command. Only text entered during the current input mode session and on the current line can be deleted with this command.

Current line : Unchanged.

Current column : Moved back as described above.

5.35.7.1.16 Scroll backward by line

Synopsis : [*count*] `control-Y`

Scroll backward *count* lines, leaving the current line and column as is, if possible.

Current line : Unchanged unless the current line scrolls off the screen; otherwise, the last line displayed.

Current column : Unchanged unless the current line scrolls off the screen; otherwise, move to the first character if it is a blank line or the last character of the line if the position of the current column is further into the line than the position of the last column of the line.

5.35.7.1.17 Terminate command or input mode

Synopsis : `<ESC>`

The effects of `<ESC>` depend on the mode and the command being entered, as follows:

— In command mode:

- If a line-oriented command (`/`, `?`, `:`, or `!`) is being entered, terminate the line-oriented command and execute it.
- If only part of a command has been entered, cancel the partial command. A command is not considered to be partially entered until at least one non-*count* character has been entered. For example, '`33c<ESC>`'' is partially entered and silently canceled, whereas '`33<ESC>`''alerts the terminal (see the next item).
- Otherwise, alert the terminal.

— In input mode: Terminate input mode and return to command mode. At this point any text that was deleted in input mode, but that has not yet been erased from the screen, shall be erased from the screen; the current line shall be redisplayed to match exactly the changes made in input mode.

Current line : When a line-oriented command is being terminated, the current line indicator shall be set as given for that command. Otherwise, unchanged.

Current column : When a line-oriented command is being terminated, the current column indicator shall be set as given for that command; When terminating input mode, if the cursor is not at the beginning of a line, the current column position shall be set to (*column* − the width of the last character inserted). Otherwise, unchanged.

5.35.7.1.18 Move cursor forward

Synopsis : `[`*count*`]` `<space>`
Synopsis : `[`*count*`]` `l` (ell)

Move the cursor forward *count* characters without changing the current line.

Current line : Unchanged.

Current column : (*current column* + the width of the next *count* characters) or the end of the line if *count* is greater than the number of characters in the line following the current column.

5.35.7.1.19 Replace text with results from shell command

Synopsis : [*count*] ! *motion* *shell-command(s)* `<newline>`

Pass the lines specified by *count* and *motion* as standard input to a shell command, for the program named in the **SHELL** environment variable, and replace those lines with the standard output of the shell command. An implementation may consider a *motion* command that does not result in an integral number of lines to be an error. After the *motion* is entered, the text of the command shall be prompted for on the last line of the display as described in 5.35.7. A warning shall be issued if the buffer has been changed since the last write.

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 `!` or `:!` command. (Thus, `!!` shall repeat the previous `!` command.) If no `!` or `:!` command has yet been executed, an informational message shall be displayed stating the problem.

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.

Current line : The first line in *range* .

Current column : The first column of the replaced text.

5.35.7.1.20 Move cursor to end-of-line

Synopsis : [*count*] `$`

Move the cursor forward to the end of a line. The *count* argument specifies the number of lines, including the current line, to move forward.

Current line : *line* + *count* − 1.

Current column : *end-of-line* .

5.35.7.1.21 Move to matching character

Synopsis : `%`

Move the cursor to the parenthesis or curly brace matching the parenthesis or curly brace at the current position or on the current line forward from the current position. Matching shall be determined as follows: for a left parenthesis (respectively curly brace) the editing buffer shall be searched forward until either a left or right parenthesis is found. If a left parenthesis is found, a counter is incremented by one, and if a right parenthesis is found, a counter is decremented by one, and the search continues (until the end of the editing buffer is found). If a right parenthesis is found when the count is less than or equal to zero, it is the matching parenthesis. For a right parenthesis (respectively curly brace) the editing buffer shall be searched backward (to the beginning of the editing buffer), and each right parenthesis increments the counter, and each left parenthesis decrements it. When searching for parentheses, curly braces shall not be counted, and vice versa. If there is no matching character, the terminal shall be alerted and the current position shall remain unchanged.

Current line : Move to the line containing the matching character, as described above.

Current column : Move to the column containing the matching character, as described above.

5.35.7.1.22 Repeat substitution

Synopsis : &

Repeat the previous substitution command (equivalent to the ex & command; see 5.10.7.2.44) using the current line as the target. Flags specified on the previous substitution command shall be ignored by this command.

Current line : Unchanged.

Current column : Unchanged if the previous current column indicator value was *end-of-line* ; otherwise, move to the first nonblank character of the current line or the first character if the line is a blank line.

5.35.7.1.23 Return to previous context (at beginning of line)

Synopsis : ''
Synopsis : ' *letter*

Return to the previous context when followed by a ' character, if the context still exists, placing the cursor at the first nonblank character of the line or the last character if the line is a blank line. If the previous context no longer exists, the ' command shall have no effect. When followed by a lowercase letter from the POSIX Locale, return to the line marked with that letter (see the m command), at the first nonblank character in the line.

When used with an operator such as d to specify an extent of text, the operation takes place over complete lines. (See also 5.35.7.1.24.)

Current line : Move to the line from the previous context.

Current column : Move to the first nonblank character of the current line or the last character if the line is a blank line.

5.35.7.1.24 Return to previous context

Synopsis : ``
Synopsis : ` *letter*

When followed by a ` character, return to the previous context, placing the cursor at the character position marked. (The previous context shall be set whenever a nonrelative move is made.) When followed by a lowercase letter from the POSIX Locale, return to the line marked with that letter (see the m command), at the character position marked.

When used with an operator such as d to specify an extent of text, the operation takes place from the exact marked place to the current position within the line.

(See also 5.35.7.1.23.)

Current line : Move to the line from the previous context.

Current column : Set to the position of the character marked from the previous context.

5.35.7.1.25 Return to previous section

Synopsis : `[[`

Back up to the previous section boundary. This command shall be affected by the `sections` option; see 5.10.7.5.18. Note that the brackets in this command are part of the command syntax itself and are not metacharacters.

Current line : Move to the previous line that is a section boundary or to the first line of the editing buffer if no more section boundaries exist preceding the current line.

Current column : Move to the first nonblank character of the current line or the last character if the line is a blank line.

5.35.7.1.26 Move to next section

Synopsis : `]]`

Move forward to a section boundary. This command shall be affected by the `sections` option; see 5.10.7.5.18. Note that the brackets in this command are part of the command syntax itself and are not metacharacters.

Current line : Move to the next line that is a section boundary or to the last line of the editing buffer if no more section boundaries exist following the current line.

Current column : Move to the first nonblank character of the current line or the last character if the line is a blank line.

5.35.7.1.27 Move to first nonblank position on current line

Synopsis : `^`

Move to the first nonblank position on the current line.

Current line : Unchanged.

Current column : Move to the first nonblank character of the current line or the last character if the line is a blank line.

5.35.7.1.28 Move back to beginning of sentence

Synopsis : `[`*count*`] (`

Move backward to the beginning of a sentence. A *count* shall move back that many sentences.

Current line : Move to the line containing the beginning of the sentence.