Manuals

Manual: vi
PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

vi - screen-oriented (visual) display editor

SYNOPSIS

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

DESCRIPTION

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

The vi (visual) utility is a screen-oriented text editor. Only the open and visual modes of the editor are described in IEEE Std 1003.1-2001; see the line editor ex for additional editing capabilities used in vi. The user can switch back and forth between vi and ex and execute ex commands from within vi.

This reference page uses the term edit buffer to describe the current working text. No specific implementation is implied by this term. All editing changes are performed on the edit buffer, and no changes to it shall affect any file until an editor command writes the file.

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 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 volume of IEEE Std 1003.1-2001 or report an error describing the terminal-related deficiency.

OPTIONS

The vi utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-c command
: See the ex command description of the **-c** option.

-r
: See the ex command description of the **-r** option.

-R
: See the ex command description of the **-R** option.

-t tagstring
: See the ex command description of the **-t** option.

-w size
: See the ex command description of the **-w** option.

OPERANDS

See the OPERANDS section of the ex command for a description of the operands supported by the vi command.

STDIN

If standard input is not a terminal device, the results are undefined. The standard input consists of a series of commands and input text, as described in the EXTENDED DESCRIPTION section.

If a read from the standard input returns an error, or if the editor detects an end-of-file condition from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

INPUT FILES

See the INPUT FILES section of the ex command for a description of the input files supported by the vi command.

ENVIRONMENT VARIABLES

See the ENVIRONMENT VARIABLES section of the ex command for the environment variables that affect the execution of the vi command.

ASYNCHRONOUS EVENTS

See the ASYNCHRONOUS EVENTS section of the ex for the asynchronous events that affect the execution of the vi command.

STDOUT

If standard output is not a terminal device, undefined results occur.

Standard output may be used for writing prompts to the user, for informational messages, and for writing lines from the file.

STDERR

If standard output is not a terminal device, undefined results occur.

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

See the OUTPUT FILES section of the ex command for a description of the output files supported by the vi command.

EXTENDED DESCRIPTION

If the terminal does not have the capabilities necessary to support an unspecified portion of the vi definition, implementations shall start initially in ex mode or open mode. Otherwise, after

initialization, vi shall be in command mode; text input mode can be entered by one of several commands used to insert or change text. In text input mode, <ESC> can be used to return to command mode; other uses of <ESC> are described later in this section; see Terminate Command or Input Mode .

Initialization in ex and vi

See Initialization in ex and vi for a description of ex and vi initialization for the vi utility.

Command Descriptions in vi

The following symbols are used in this reference page to represent arguments to commands.

buffer
: See the description of buffer in the EXTENDED DESCRIPTION section of the ex utility; see Command Descriptions in ex .

In open and visual mode, when a command synopsis shows both [buffer] and [count] preceding the command name, they can be specified in either order.

count
: A positive integer used as an optional argument to most commands, either to give a repeat count or as a size. This argument is optional and shall default to 1 unless otherwise specified.

The Synopsis lines for the vi commands <control>-G, <control>-L, <control>-R, <control>-], **%**, **&**, **^**, **D**, **m**, **M**, **Q**, **u**, **U**, and **ZZ** do not have count as an optional argument. Regardless, it shall not be an error to specify a count to these commands, and any specified count shall be ignored.

motion
: An optional trailing argument used by the **!**, **<**, **>**, **c**, **d**, and **y** commands, which is used to indicate the region of text that shall be affected by the command. The motion can be either one of the command characters repeated or one of several other vi commands (listed in the following table). Each of the applicable commands specifies the region of text matched by repeating the command; each command that can be used as a motion command specifies the region of text it affects.

Commands that take motion arguments operate on either lines or characters, depending on the circumstances. When operating on lines, all lines that fall partially or wholly within the text region specified for the command shall be affected. When operating on characters, only the exact characters in the specified text region shall be affected. Each motion command specifies this individually.

When commands that may be motion commands are not used as motion commands, they shall set the current position to the current line and column as specified.

The following commands shall be valid cursor motion commands:

<apostrophe>	**(**	**-**	**j**	**H**
<carriage-return>	**)**	**$**	**k**	**L**
<comma>	**[[**	**%**	**l**	**M**
<control>-H	**]]**	**_**	**n**	**N**
<control>-N	**{**	**;**	**t**	**T**
<control>-P	**}**	**?**	**w**	**W**
<grave accent>	**^**	**b**	**B**	
<newline>	**+**	**e**	**E**	
<space>	**|**	**f**	**F**	
<zero>	**/**	**h**	**G**	

Any count that is specified to a command that has an associated motion command shall be applied to the motion command. If a count is applied to both the command and its associated motion command, the effect shall be multiplicative.

The following symbols are used in this section to specify locations in the edit buffer:

current character

The character that is currently indicated by the cursor.

end of a line

The point located between the last non- <newline> (if any) and the terminating <newline> of a line. For an empty line, this location coincides with the beginning of the line.

end of the edit buffer

The location corresponding to the end of the last line in the edit buffer.

The following symbols are used in this section to specify command actions:

bigword

In the POSIX locale, vi shall recognize four kinds of bigwords:

1. A maximal sequence of non- <blank>s preceded and followed by <blank>s or the beginning or end of a line or the edit buffer
2. One or more sequential blank lines
3. The first character in the edit buffer
4. The last non- <newline> in the edit buffer

word

In the POSIX locale, vi shall recognize five kinds of words:

1. A maximal sequence of letters, digits, and underscores, delimited at both ends by:
    * Characters other than letters, digits, or underscores
    * The beginning or end of a line
    * The beginning or end of the edit buffer
2. A maximal sequence of characters other than letters, digits, underscores, or <blank>s, delimited at both ends by:
    * A letter, digit, underscore
    * <blank>s
    * The beginning or end of a line
    * The beginning or end of the edit buffer
3. One or more sequential blank lines
4. The first character in the edit buffer

5. The last non- <newline> in the edit buffer

section boundary

A section boundary is one of the following:

1. A line whose first character is a <form-feed>

2. A line whose first character is an open curly brace ('**{**')

3. A line whose first character is a period and whose second and third characters match a two-character pair in the **sections** edit option (see ed)

4. A line whose first character is a period and whose only other character matches the first character of a two-character pair in the **sections** edit option, where the second character of the two-character pair is a <space>

5. The first line of the edit buffer

6. The last line of the edit buffer if the last line of the edit buffer is empty or if it is a **]]** or **}** command; otherwise, the last non- <newline> of the last line of the edit buffer

paragraph boundary

A paragraph boundary is one of the following:

1. A section boundary

2. A line whose first character is a period and whose second and third characters

match a two-character pair in the **paragraphs** edit option (see ed)

3. A line whose first character is a period and whose only other character matches the first character of a two-character pair in the paragraphs edit option, where the second character of the two-character pair is a <space>

4. One or more sequential blank lines

remembered search direction

See the description of remembered search direction in ed.

sentence boundary

A sentence boundary is one of the following:

1. A paragraph boundary

2. The first non- <blank> that occurs after a paragraph boundary

3. The first non- <blank> that occurs after a period (**'.'**), exclamation mark (**'!'**), or question mark (**'?'**), followed by two <space>s or the end of a line; any number of closing parenthesis (**')'**), closing brackets (**']'**), double quote (**'"'**), or single quote (**'''**) characters can appear between the punctuation mark and the two <space>s or end-of-line

In the remainder of the description of the vi utility, the term "buffer line" refers to a line in the edit buffer and the term "display line" refers to the line or lines on the display screen used to display one buffer line. The term "current line" refers to a specific "buffer line".

If there are display lines on the screen for which there are no corresponding buffer lines because they correspond to lines that would be after the end of the file, they shall be displayed as a single tilde (**'~'**) character, plus the terminating <newline>.

The last line of the screen shall be used to report errors or display informational messages. It shall also be used to display the input for "line-oriented commands" (**/**, **?**, **:**, and **!**). When a line-oriented command is executed, the editor shall enter text input mode on the last line on the screen, using the

respective command characters as prompt characters. (In the case of the **!** command, the associated motion shall be entered by the user before the editor enters text input mode.) The line entered by the user shall be terminated by a <newline>, a non- <control>-V-escaped <carriage-return>, or unescaped <ESC>. It is unspecified if more characters than require a display width minus one column number of screen columns can be entered.

If any command is executed that overwrites a portion of the screen other than the last line of the screen (for example, the ex **suspend** or **!** commands), other than the ex **shell** command, the user shall be prompted for a character before the screen is refreshed and the edit session continued.

<tab>s shall take up the number of columns on the screen set by the **tabstop** edit option (see ed), unless there are less than that number of columns before the display margin that will cause the displayed line to be folded; in this case, they shall only take up the number of columns up to that boundary.

The cursor shall be placed on the current line and relative to the current column as specified by each command described in the following sections.

In open mode, if the current line is not already displayed, then it shall be displayed.

In visual mode, if the current line is not displayed, then the lines that are displayed shall be expanded, scrolled, or redrawn to cause an unspecified portion of the current line to be displayed. If the screen is redrawn, no more than the number of display lines specified by the value of the **window** edit option shall be displayed (unless the current line cannot be completely displayed in the number of display lines specified by the **window** edit option) and the current line shall be positioned as close to the center of the displayed lines as possible (within the constraints imposed by the distance of the line from the beginning or end of the edit buffer). If the current line is before the first line in the display and the screen is scrolled, an unspecified portion of the current line shall be placed on the first line of the display. If the current line is after the last line in the display and the screen is scrolled, an unspecified portion of the current line shall be placed on the last line of the display.

In visual mode, if a line from the edit buffer (other than the current line) does not entirely fit into the lines at the bottom of the display that are available for its presentation, the editor may choose not to display any portion of the line. The lines of the display that do not contain text from the edit buffer for this reason shall each consist of a single **'@'** character.

In visual mode, the editor may choose for unspecified reasons to not update lines in the display to correspond to the underlying edit buffer text. The lines of the display that do not correctly correspond to text from the edit buffer for this reason shall consist of a single **'@'** character (plus the terminating <newline>), and the <control>-R command shall cause the editor to update the screen to correctly represent the edit buffer.

Open and visual mode commands that set the current column set it to a column position in the display, and not a character position in the line. In this case, however, the column position in the display shall be calculated for an infinite width display; for example, the column related to a character that is part of a line that has been folded onto additional screen lines will be offset from the display line column where the buffer line begins, not from the beginning of a particular display line.

The display cursor column in the display is based on the value of the current column, as follows, with each rule applied in turn:

1. If the current column is after the last display line column used by the displayed line, the display cursor column shall be set to the last display line column occupied by the last non-<newline> in the current line; otherwise, the display cursor column shall be set to the current column.

2.
If the character of which some portion is displayed in the display line column specified by the display cursor column requires more than a single display line column:

a.
If in text input mode, the display cursor column shall be adjusted to the first display line column in which any portion of that character is displayed.

b.
Otherwise, the display cursor column shall be adjusted to the last display line column in which any portion of that character is displayed.

The current column shall not be changed by these adjustments to the display cursor column.

If an error occurs during the parsing or execution of a vi command:

*
The terminal shall be alerted. Execution of the vi command shall stop, and the cursor (for example, the current line and column) shall not be further modified.

*
Unless otherwise specified by the following command sections, it is unspecified whether an informational message shall be displayed.

*
Any partially entered vi command shall be discarded.

*
If the vi command resulted from a **map** expansion, all characters from that **map** expansion shall be discarded, except as otherwise specified by the **map** command (see ed).

*
If the vi command resulted from the execution of a buffer, no further commands caused by the execution of the buffer shall be executed.

Page Backwards

Synopsis:

[*count***] <control>-B**

If in open mode, the <control>-B command shall behave identically to the **z** command. Otherwise, if the current line is the first line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the last line of the display shall be some portion of:

(*current first line***) -1**

otherwise, display a screen where the first line of the display shall be some portion of:

(*current first line***) -** *count* **x ((window edit option) -2)**

If this calculation would result in a line that is before the first line of the edit buffer, the first line of the display shall display some portion of the first line of the edit buffer.

Current line: If no lines from the previous display remain on the screen, set to the last line of the display; otherwise, set to (line - the number of new lines displayed on this screen).

Current column: Set to non- <blank>.

Scroll Forward

Synopsis:

[*count***] <control>-D**

If the current line is the last line of the edit buffer, it shall be an error.

If no count is specified, count shall default to the count associated with the previous <control>-D or <control>-U command. If there was no previous <control>-D or <control>-U command, count shall default to the value of the **scroll** edit option.

If in open mode, write lines starting with the line after the current line, until count lines or the last line of the file have been written.

Current line: If the current line + count is past the last line of the edit buffer, set to the last line of the edit buffer; otherwise, set to the current line + count.

Current column: Set to non- <blank>.

Scroll Forward by Line

Synopsis:

[*count***] <control>-E**

Display the line count lines after the last line currently displayed.

If the last line of the edit buffer is displayed, it shall be an error. If there is no line count lines after the last line currently displayed, the last line of the display shall display some portion of the last line of the edit buffer.

Current line: Unchanged if the previous current character is displayed; otherwise, set to the first line displayed.

Current column: Unchanged.

Page Forward

Synopsis:

[*count***] <control>-F**

If in open mode, the <control>-F command shall behave identically to the **z** command. Otherwise, if the current line is the last line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the first line of the display shall be some portion of:

(*current last line***) +1**

otherwise, display a screen where the first line of the display shall be some portion of:

(*current first line***) + *count* x ((window edit option) -2)**

If this calculation would result in a line that is after the last line of the edit buffer, the last line of the display shall display some portion of the last line of the edit buffer.

Current line: If no lines from the previous display remain on the screen, set to the first line of the display; otherwise, set to (line + the number of new lines displayed on this screen).

Current column: Set to non- <blank>.

Display Information

Synopsis:

<control>-G

This command shall be equivalent to the ex **file** command.

Move Cursor Backwards

Synopsis:

[*count***] <control>-H**

[*count***] h**

the current *erase* character (see stty)

If there are no characters before the current character on the current line, it shall be an error. If there are less than count previous characters on the current line, count shall be adjusted to the number of previous characters on the line.

If used as a motion command:

1. The text region shall be from the character before the starting cursor up to and including the countth character before the starting cursor.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to (column - the number of columns occupied by count characters ending with the previous current column).

Move Down

Synopsis:

[*count***] <newline>**

[*count***] <control>-J**

[*count***] <control>-M**

[*count***] <control>-N**

[*count***] j**

[*count***] <carriage-return>**

[*count***] +**

If there are less than count lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the next count - 1 lines.

2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

Current line: Set to current line+ count.

Current column: Set to non- <blank> for the <carriage-return>, <control>-M, and **+** commands; otherwise, unchanged.

Clear and Redisplay

Synopsis:

```
<control>-L
```

If in open mode, clear the screen and redisplay the current line. Otherwise, clear and redisplay the screen.

Current line: Unchanged.

Current column: Unchanged.

Move Up

Synopsis:

```
[count] <control>-P

[count] k

[count] -
```

If there are less than count lines before the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the previous count lines.

2.
Any text copied to a buffer shall be in line mode.

If not used as a motion command:

Current line: Set to current line - count.

Current column: Set to non- <blank> for the **-** command; otherwise, unchanged.

Redraw Screen

Synopsis:

<control>-R

If any lines have been deleted from the display screen and flagged as deleted on the terminal using the @ convention (see the beginning of the EXTENDED DESCRIPTION section), they shall be redisplayed to match the contents of the edit buffer.

It is unspecified whether lines flagged with @ because they do not fit on the terminal display shall be affected.

Current line: Unchanged.

Current column: Unchanged.

Scroll Backward

Synopsis:

[*count***] <control>-U**

If the current line is the first line of the edit buffer, it shall be an error.

If no count is specified, count shall default to the count associated with the previous <control>-D or <control>-U command. If there was no previous <control>-D or <control>-U command, count shall default to the value of the **scroll** edit option.

Current line: If count is greater than the current line, set to 1; otherwise, set to the current line -

count.

Current column: Set to non- <blank>.

Scroll Backward by Line

Synopsis:

[*count***] <control>-Y**

Display the line count lines before the first line currently displayed.

If the current line is the first line of the edit buffer, it shall be an error. If this calculation would result in a line that is before the first line of the edit buffer, the first line of the display shall display some portion of the first line of the edit buffer.

Current line: Unchanged if the previous current character is displayed; otherwise, set to the first line displayed.

Current column: Unchanged.

Edit the Alternate File

Synopsis:

<control>-^

This command shall be equivalent to the ex **edit** command, with the alternate pathname as its argument.

Terminate Command or Input Mode

Synopsis:

<ESC>

If a partial vi command (as defined by at least one, non- count character) has been entered, discard the count and the command character(s).

Otherwise, if no command characters have been entered, and the <ESC> was the result of a map expansion, the terminal shall be alerted and the <ESC> character shall be discarded, but it shall not be an error.

Otherwise, it shall be an error.

Current line: Unchanged.

Current column: Unchanged.

Search for tagstring

Synopsis:

<control>-]

If the current character is not a word or <blank>, it shall be an error.

This command shall be equivalent to the ex **tag** command, with the argument to that command defined as follows.

If the current character is a <blank>:

1. Skip all <blank>s after the cursor up to the end of the line.

2. If the end of the line is reached, it shall be an error.

Then, the argument to the ex **tag** command shall be the current character and all subsequent characters, up to the first non-word character or the end of the line.

Move Cursor Forward

Synopsis:

[*count***] <space>**

[*count***] l** (ell)

If there are less than count non- <newline>s after the cursor on the current line, count shall be adjusted to the number of non- <newline>s after the cursor on the line.

If used as a motion command:

1. If the current or countth character after the cursor is the last non- <newline> in the line, the text region shall be comprised of the current character up to and including the last non- <newline> in the line. Otherwise, the text region shall be from the current character up to, but not including, the countth character after the cursor.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

If there are no non- <newline>s after the current character on the current line, it shall be an error.

Current line: Unchanged.

Current column: Set to the last column that displays any portion of the countth character after the current character.

Replace Text with Results from Shell Command

Synopsis:

[*count***] !** *motion shell-commands* **<newline>**

If the motion command is the **!** command repeated:

1. If the edit buffer is empty and no count was supplied, the command shall be the equivalent of the ex **:read !** command, with the text input, and no text shall be copied to any buffer.

2. Otherwise:

    a.

If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

b.
The text region shall be from the current line up to and including the next count -1 lines.

Otherwise, the text region shall be the lines in which any character of the text region specified by the motion command appear.

Any text copied to a buffer shall be in line mode.

This command shall be equivalent to the ex **!** command for the specified lines.

Move Cursor to End-of-Line

Synopsis:

[*count***] $**

It shall be an error if there are less than (count -1) lines after the current line in the edit buffer.

If used as a motion command:

1.
If count is 1:

a.
It shall be an error if the line is empty.

b.
Otherwise, the text region shall consist of all characters from the starting cursor to the last non- <newline> in the line, inclusive, and any text copied to a buffer shall be in character mode.

2.

Otherwise, if the starting cursor position is at or before the first non- <blank> in the line, the text region shall consist of the current and the next count -1 lines, and any text saved to a buffer shall be in line mode.

3.

Otherwise, the text region shall consist of all characters from the starting cursor to the last non- <newline> in the line that is count -1 lines forward from the current line, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the current line + count-1.

Current column: The current column is set to the last display line column of the last non- <newline> in the line, or column position 1 if the line is empty.

The current column shall be adjusted to be on the last display line column of the last non- <newline> of the current line as subsequent commands change the current line, until a command changes the current column.

Move to Matching Character

Synopsis:

%

If the character at the current position is not a parenthesis, bracket, or curly brace, search forward in the line to the first one of those characters. If no such character is found, it shall be an error.

The matching character shall be the parenthesis, bracket, or curly brace matching the parenthesis, bracket, or curly brace, respectively, that was at the current position or that was found on the current line.

Matching shall be determined as follows, for an open parenthesis:

1.

Set a counter to 1.

2.

Search forwards until a parenthesis is found or the end of the edit buffer is reached.

3.

If the end of the edit buffer is reached, it shall be an error.

4. If an open parenthesis is found, increment the counter by 1.

5. If a close parenthesis is found, decrement the counter by 1.

6. If the counter is zero, the current character is the matching character.

Matching for a close parenthesis shall be equivalent, except that the search shall be backwards, from the starting character to the beginning of the buffer, a close parenthesis shall increment the counter by 1, and an open parenthesis shall decrement the counter by 1.

Matching for brackets and curly braces shall be equivalent, except that searching shall be done for open and close brackets or open and close curly braces. It is implementation-defined whether other characters are searched for and matched as well.

If used as a motion command:

1. If the matching cursor was after the starting cursor in the edit buffer, and the starting cursor position was at or before the first non- <blank> non- <newline> in the starting line, and the matching cursor position was at or after the last non- <blank> non- <newline> in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.

2. If the matching cursor was before the starting cursor in the edit buffer, and the starting cursor position was at or after the last non- <blank> non- <newline> in the starting line, and the matching cursor position was at or before the first non- <blank> non- <newline> in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.

3. Otherwise, the text region shall consist of the starting character to the matching character, inclusive, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line where the matching character is located.

Current column: Set to the last column where any portion of the matching character is displayed.

Repeat Substitution

Synopsis:

```
&
```

Repeat the previous substitution command. This command shall be equivalent to the ex **&** command with the current line as its addresses, and without options, count, or flags.

Return to Previous Context at Beginning of Line

Synopsis:

```
' character
```

It shall be an error if there is no line in the edit buffer marked by character.

If used as a motion command:

1. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.

2. The text region shall consist of the starting line up to and including the marked line, and any text copied to a buffer shall be in line mode.

If not used as a motion command:

Current line: Set to the line referenced by the mark.

Current column: Set to non- <blank>.

Return to Previous Context

Synopsis:

```
` character
```

It shall be an error if the marked line is no longer in the edit buffer. If the marked line no longer contains a character in the saved numbered character position, it shall be as if the marked position is the first non- <blank>.

If used as a motion command:

1. It shall be an error if the marked cursor references the same character in the edit buffer as the starting cursor.

2. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.

3. If the starting line is empty or the starting cursor is at or before the first non- <blank> non- <newline> of the starting line, and the marked cursor line is empty or the marked cursor references the first character of the marked cursor line, the text region shall consist of all lines containing characters from the starting cursor to the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in line mode.

4. Otherwise, if the marked cursor line is empty or the marked cursor references a character at or before the first non- <blank> non- <newline> of the marked cursor line, the region of text shall be from the starting cursor to the last non- <newline> of the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in character mode.

5. Otherwise, the region of text shall be from the starting cursor (inclusive), to the marked cursor (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line referenced by the mark.

Current column: Set to the last column in which any portion of the character referenced by the mark is displayed.

Return to Previous Section

Synopsis:

[*count***] [[**

Move the cursor backward through the edit buffer to the first character of the previous section boundary, count times.

If used as a motion command:

1. If the starting cursor was at the first character of the starting line or the starting line was empty, and the first character of the boundary was the first character of the boundary line, the text region shall consist of the current line up to and including the line where the countth next boundary starts, and any text copied to a buffer shall be in line mode.

2. If the boundary was the last line of the edit buffer or the last non- <newline> of the last line of the edit buffer, the text region shall consist of the last character in the edit buffer up to and including the starting character, and any text saved to a buffer shall be in character mode.

3. Otherwise, the text region shall consist of the starting character up to but not including the first character in the countth next boundary, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line where the countth next boundary in the edit buffer starts.

Current column: Set to the last column in which any portion of the first character of the countth next boundary is displayed, or column position 1 if the line is empty.

Move to Next Section

Synopsis:

```
[count] ]]
```

Move the cursor forward through the edit buffer to the first character of the next section boundary, count times.

If used as a motion command:

1. If the starting cursor was at the first character of the starting line or the starting line was empty, and the first character of the boundary was the first character of the boundary line, the text region shall consist of the current line up to and including the line where the countth previous boundary starts, and any text copied to a buffer shall be in line mode.

2. If the boundary was the first line of the edit buffer, the text region shall consist of the first character in the edit buffer up to but not including the starting character, and any text copied to a buffer shall be in character mode.

3. Otherwise, the text region shall consist of the first character in the countth previous section boundary up to but not including the starting character, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line where the countth previous boundary in the edit buffer starts.

Current column: Set to the last column in which any portion of the first character of the countth previous boundary is displayed, or column position 1 if the line is empty.

Move to First Non-<blank> Position on Current Line

Synopsis:

```
^
```

If used as a motion command:

1. If the line has no non- <blank> non- <newline>s, or if the cursor is at the first non- <blank> non- <newline> of the line, it shall be an error.

2. If the cursor is before the first non- <blank> non- <newline> of the line, the text region shall be comprised of the current character, up to, but not including, the first non- <blank> non- <newline> of the line.

3. If the cursor is after the first non- <blank> non- <newline> of the line, the text region shall be from the character before the starting cursor up to and including the first non- <blank> non- <newline> of the line.

4.
Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to non- <blank>.

Current and Line Above

Synopsis:

```
[count] _
```

If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1.
If count is less than 2, the text region shall be the current line.

2.
Otherwise, the text region shall include the starting line and the next count -1 lines.

3.
Any text copied to a buffer shall be in line mode.

If not used as a motion command:

Current line: Set to current line + count -1.

Current column: Set to non- <blank>.

Move Back to Beginning of Sentence

Synopsis:

[*count***] (**

Move backward to the beginning of a sentence. This command shall be equivalent to the **[[** command, with the exception that sentence boundaries shall be used instead of section boundaries.

Move Forward to Beginning of Sentence

Synopsis:

[*count***])**

Move forward to the beginning of a sentence. This command shall be equivalent to the **]]** command, with the exception that sentence boundaries shall be used instead of section boundaries.

Move Back to Preceding Paragraph

Synopsis:

[*count***] {**

Move back to the beginning of the preceding paragraph. This command shall be equivalent to the **[[** command, with the exception that paragraph boundaries shall be used instead of section boundaries.

Move Forward to Next Paragraph

Synopsis:

[*count***] }**

Move forward to the beginning of the next paragraph. This command shall be equivalent to the **]]** command, with the exception that paragraph boundaries shall be used instead of section boundaries.

Move to Specific Column Position

Synopsis:

[*count***] |**

For the purposes of this command, lines that are too long for the current display and that have been folded shall be treated as having a single, 1-based, number of columns.

If there are less than count columns in which characters from the current line are displayed on the screen, count shall be adjusted to be the last column in which any portion of the line is displayed on the screen.

If used as a motion command:

1. If the line is empty, or the cursor character is the same as the character on the countth column of the line, it shall be an error.

2. If the cursor is before the countth column of the line, the text region shall be comprised of the current character, up to but not including the character on the countth column of the line.

3. If the cursor is after the countth column of the line, the text region shall be from the character before the starting cursor up to and including the character on the countth column of the line.

4. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to the last column in which any portion of the character that is displayed in the count column of the line is displayed.

Reverse Find Character

Synopsis:

```
[count] ,
```

If the last **F**, **f**, **T**, or **t** command was **F**, **f**, **T**, or **t**, this command shall be equivalent to an **f**, **F**, **t**, or **T** command, respectively, with the specified count and the same search character.

If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

Repeat

Synopsis:

```
[count] .
```

Repeat the last **!**, **<**, **>**, **A**, **C**, **D**, **I**, **J**, **O**, **P**, **R**, **S**, **X**, **Y**, **a**, **c**, **d**, **i**, **o**, **p**, **r**, **s**, **x**, **y**, or **~** command. It shall be an error if none of these commands have been executed. Commands (other than commands that enter text input mode) executed as a result of map expansions, shall not change the value of the last repeatable command.

Repeated commands with associated motion commands shall repeat the motion command as well; however, any specified count shall replace the count(s) that were originally specified to the repeated command or its associated motion command.

If the motion component of the repeated command is **f**, **F**, **t**, or **T**, the repeated command shall not set the remembered search character for the **;** and **,** commands.

If the repeated command is **p** or **P**, and the buffer associated with that command was a numeric buffer named with a number less than 9, the buffer associated with the repeated command shall be set to be the buffer named by the name of the previous buffer logically incremented by 1.

If the repeated character is a text input command, the input text associated with that command is repeated literally:

*
Input characters are neither macro or abbreviation-expanded.

*
Input characters are not interpreted in any special way with the exception that <newline>,

<carriage-return>, and <control>-T behave as described in Input Mode Commands in vi .

Current line: Set as described for the repeated command.

Current column: Set as described for the repeated command.

Find Regular Expression

Synopsis:

/

If the input line contains no non- <newline>s, it shall be equivalent to a line containing only the last regular expression encountered. The enhanced regular expressions supported by vi are described in Regular Expressions in ex .

Otherwise, the line shall be interpreted as one or more regular expressions, optionally followed by an address offset or a vi **z** command.

If the regular expression is not the last regular expression on the line, or if a line offset or **z** command is specified, the regular expression shall be terminated by an unescaped **'/'** character, which shall not be used as part of the regular expression. If the regular expression is not the first regular expression on the line, it shall be preceded by zero or more <blank>s, a semicolon, zero or more <blank>s, and a leading **'/'** character, which shall not be interpreted as part of the regular expression. It shall be an error to precede any regular expression with any characters other than these.

Each search shall begin from the character after the first character of the last match (or, if it is the first search, after the cursor). If the **wrapscan** edit option is set, the search shall continue to the character before the starting cursor character; otherwise, to the end of the edit buffer. It shall be an error if any search fails to find a match, and an informational message to this effect shall be displayed.

An optional address offset (see Addressing in ex) can be specified after the last regular expression by including a trailing **'/'** character after the regular expression and specifying the address offset. This offset will be from the line containing the match for the last regular expression specified. It shall be an error if the line offset would indicate a line address less than 1 or greater than the last line in the edit buffer. An address offset of zero shall be supported. It shall be an error to follow the address offset with any other characters than <blank>s.

If not used as a motion command, an optional **z** command (see Redraw Window) can be specified after the last regular expression by including a trailing **'/'** character after the regular expression, zero or more <blank>s, a **'z'** , zero or more <blank>s, an optional new **window** edit option value, zero or more <blank>s, and a location character. The effect shall be as if the **z** command was executed after the **/** command. It shall be an error to follow the **z** command with any other characters than <blank>s.

The remembered search direction shall be set to forward.

If used as a motion command:

1. It shall be an error if the last match references the same character in the edit buffer as the starting cursor.

2. If any address offset is specified, the last match shall be adjusted by the specified offset as described previously.

3. If the starting cursor is after the last match, then the locations of the starting cursor and the last match in the edit buffer shall be logically swapped.

4. If any address offset is specified, the text region shall consist of all lines containing characters from the starting cursor to the last match line, inclusive, and any text copied to a buffer shall be in line mode.

5. Otherwise, if the starting line is empty or the starting cursor is at or before the first non- <blank> non- <newline> of the starting line, and the last match line is empty or the last match starts at the first character of the last match line, the text region shall consist of all lines containing characters from the starting cursor to the line before the last match line, inclusive, and any text copied to a buffer shall be in line mode.

6. Otherwise, if the last match line is empty or the last match begins at a character at or before the first non- <blank> non- <newline> of the last match line, the region of text shall be from the current cursor to the last non- <newline> of the line before the last match line, inclusive, and any text copied to a buffer shall be in character mode.

7. Otherwise, the region of text shall be from the current cursor (inclusive), to the first character of the last match (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: If a match is found, set to the last matched line plus the address offset, if any; otherwise, unchanged.

Current column: Set to the last column on which any portion of the first character in the last matched string is displayed, if a match is found; otherwise, unchanged.

Move to First Character in Line

Synopsis:

0 (zero)

Move to the first character on the current line. The character **'0'** shall not be interpreted as a command if it is immediately preceded by a digit.

If used as a motion command:

1. If the cursor character is the first character in the line, it shall be an error.

2. The text region shall be from the character before the cursor character up to and including the first character in the line.

3. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: The last column in which any portion of the first character in the line is displayed, or if the line is empty, unchanged.

Execute an ex Command

Synopsis:

:

Execute one or more ex commands.

If any portion of the screen other than the last line of the screen was overwritten by any ex command (except **shell**), vi shall display a message indicating that it is waiting for an input from the user, and shall then read a character. This action may also be taken for other, unspecified reasons.

If the next character entered is a **:** , another ex command shall be accepted and executed. Any other character shall cause the screen to be refreshed and vi shall return to command mode.

Current line: As specified for the ex command.

Current column: As specified for the ex command.

Repeat Find

Synopsis:

[*count***] ;**

This command shall be equivalent to the last **F**, **f**, **T**, or **t** command, with the specified count, and with the same search character used for the last **F**, **f**, **T**, or **t** command. If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

Shift Left

Synopsis:

[*count***] <** *motion*

If the motion command is the **<** command repeated:

1. If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

2. The text region shall be from the current line, up to and including the next count -1 lines.

Shift any line in the text region specified by the count and motion command one shiftwidth (see the ex **shiftwidth** option) toward the start of the line, as described by the ex **<** command. The unshifted lines shall be copied to the unnamed buffer in line mode.

Current line: If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

Current column: Set to non- <blank>

Shift Right

Synopsis:

[*count***] >** *motion*

If the motion command is the **>** command repeated:

1. If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

2. The text region shall be from the current line, up to and including the next count -1 lines.

Shift any line with characters in the text region specified by the count and motion command one shiftwidth (see the ex **shiftwidth** option) away from the start of the line, as described by the ex **>** command. The unshifted lines shall be copied into the unnamed buffer in line mode.

Current line: If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

Current column: Set to non- <blank>.

Scan Backwards for Regular Expression

Synopsis:

?

Scan backwards; the **?** command shall be equivalent to the **/** command (see Find Regular Expression) with the following exceptions:

1. The input prompt shall be a **'?'** .

2. Each search shall begin from the character before the first character of the last match (or, if

it is the first search, the character before the cursor character).

3.
The search direction shall be from the cursor toward the beginning of the edit buffer, and the **wrapscan** edit option shall affect whether the search wraps to the end of the edit buffer and continues.

4.
The remembered search direction shall be set to backward.

Execute

Synopsis:

```
@buffer
```

If the buffer is specified as **@**, the last buffer executed shall be used. If no previous buffer has been executed, it shall be an error.

Behave as if the contents of the named buffer were entered as standard input. After each line of a line-mode buffer, and all but the last line of a character mode buffer, behave as if a <newline> were entered as standard input.

If an error occurs during this process, an error message shall be written, and no more characters resulting from the execution of this command shall be processed.

If a count is specified, behave as if that count were entered as user input before the characters from the **@** buffer were entered.

Current line: As specified for the individual commands.

Current column: As specified for the individual commands.

Reverse Case

Synopsis:

```
[count] ~
```

Reverse the case of the current character and the next count -1 characters, such that lowercase characters that have uppercase counterparts shall be changed to uppercase characters, and uppercase characters that have lowercase counterparts shall be changed to lowercase characters, as prescribed by the current locale. No other characters shall be affected by this command.

If there are less than count -1 characters after the cursor in the edit buffer, count shall be adjusted to the number of characters after the cursor in the edit buffer minus 1.

For the purposes of this command, the next character after the last non- <newline> on the line shall be the next character in the edit buffer.

Current line: Set to the line including the (count-1)th character after the cursor.

Current column: Set to the last column in which any portion of the (count-1)th character after the cursor is displayed.

Append

Synopsis:

[*count***] a**

Enter text input mode after the current cursor position. No characters already in the edit buffer shall be affected by this command. A count shall cause the input text to be appended count -1 more times to the end of the input.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Append at End-of-Line

Synopsis:

[*count***] A**

This command shall be equivalent to the vi command:

$ [*count* **] a**

(see Append).

Move Backward to Preceding Word

Synopsis:

[*count***] b**

With the exception that words are used as the delimiter instead of bigwords, this command shall be equivalent to the **B** command.

Move Backward to Preceding Bigword

Synopsis:

[*count***] B**

If the edit buffer is empty or the cursor is on the first character of the edit buffer, it shall be an error. If less than count bigwords begin between the cursor and the start of the edit buffer, count shall be adjusted to the number of bigword beginnings between the cursor and the start of the edit buffer.

If used as a motion command:

1. The text region shall be from the first character of the countth previous bigword beginning up to but not including the cursor character.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line containing the current column.

Current column: Set to the last column upon which any part of the first character of the countth previous bigword is displayed.

Change

Synopsis:

[*buffer***][***count***] c** *motion*

If the motion command is the **c** command repeated:

1. The buffer text shall be in line mode.

2. If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

3. The text region shall be from the current line up to and including the next count -1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

The replaced text shall be copied into buffer, if specified, and into the unnamed buffer. If the text to be replaced contains characters from more than a single line, or the buffer text is in line mode, the replaced text shall be copied into the numeric buffers as well.

If the buffer text is in line mode:

1. Any lines that contain characters in the region shall be deleted, and the editor shall enter text input mode at the beginning of a new line which shall replace the first line deleted.

2. If the **autoindent** edit option is set, **autoindent** characters equal to the **autoindent** characters on the first line deleted shall be inserted as if entered by the user.

Otherwise, if characters from more than one line are in the region of text:

1. The text shall be deleted.

2. Any text remaining in the last line in the text region shall be appended to the first line in the region, and the last line in the region shall be deleted.

3.
The editor shall enter text input mode after the last character not deleted from the first line in the text region, if any; otherwise, on the first column of the first line in the region.

Otherwise:

1.
If the glyph for **'$'** is smaller than the region, the end of the region shall be marked with a **'$'** .

2.
The editor shall enter text input mode, overwriting the region of text.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Change to End-of-Line

Synopsis:

[*buffer***][***count***] C**

This command shall be equivalent to the vi command:

[*buffer***][***count***] c$**

See the **c** command.

Delete

Synopsis:

[*buffer***][***count***] d** *motion*

If the motion command is the **d** command repeated:

1. The buffer text shall be in line mode.

2. If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

3. The text region shall be from the current line up to and including the next count -1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

If in open mode, and the current line is deleted, and the line remains on the display, an **'@'** character shall be displayed as the first glyph of that line.

Delete the region of text into buffer, if specified, and into the unnamed buffer. If the text to be deleted contains characters from more than a single line, or the buffer text is in line mode, the deleted text shall be copied into the numeric buffers, as well.

Current line: Set to the first text region line that appears in the edit buffer, unless that line has been deleted, in which case it shall be set to the last line in the edit buffer, or line 1 if the edit buffer is empty.

Current column:

1. If the line is empty, set to column position 1.

2. Otherwise, if the buffer text is in line mode or the motion was from the cursor toward the end of the edit buffer:

    a. If a character from the current line is displayed in the current column, set to the last column that displays any portion of that character.

    b. Otherwise, set to the last column in which any portion of any character in the line is displayed.

3.

Otherwise, if a character is displayed in the column that began the text region, set to the last column that displays any portion of that character.

4.
Otherwise, set to the last column in which any portion of any character in the line is displayed.

Delete to End-of-Line

Synopsis:

[*buffer***] D**

Delete the text from the current position to the end of the current line; equivalent to the vi command:

[*buffer***] d$**

Move to End-of-Word

Synopsis:

[*count***] e**

With the exception that words are used instead of bigwords as the delimiter, this command shall be equivalent to the **E** command.

Move to End-of-Bigword

Synopsis:

[*count***] E**

If the edit buffer is empty it shall be an error. If less than count bigwords end between the cursor and the end of the edit buffer, count shall be adjusted to the number of bigword endings between the cursor and the end of the edit buffer.

If used as a motion command:

1. The text region shall be from the last character of the countth next bigword up to and including the cursor character.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Set to the line containing the current column.

Current column: Set to the last column upon which any part of the last character of the countth next bigword is displayed.

Find Character in Current Line (Forward)

Synopsis:

[*count***] f** *character*

It shall be an error if count occurrences of the character do not occur after the cursor in the line.

If used as a motion command:

1. The text range shall be from the cursor character up to and including the countth occurrence of the specified character after the cursor.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to the last column in which any portion of the countth occurrence of the specified character after the cursor appears in the line.

Find Character in Current Line (Reverse)

Synopsis:

```
[count] F character
```

It shall be an error if count occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1. The text region shall be from the countth occurrence of the specified character before the cursor, up to, but not including the cursor character.

2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to the last column in which any portion of the countth occurrence of the specified character before the cursor appears in the line.

Move to Line

Synopsis:

```
[count] G
```

If count is not specified, it shall default to the last line of the edit buffer. If count is greater than the

last line of the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall be from the cursor line up to and including the specified line.

2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

Current line: Set to count if count is specified; otherwise, the last line.

Current column: Set to non- <blank>.

Move to Top of Screen

Synopsis:

[*count***] H**

If the beginning of the line count greater than the first line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.

2. Otherwise, the text region shall be from the starting line up to and including (the first line of the display + count -1).

3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

If in open mode, this command shall set the current column to non- <blank> and do nothing else.

Otherwise, it shall set the current line and current column as follows.

Current line: Set to (the first line of the display + count -1).

Current column: Set to non- <blank>.

Insert Before Cursor

Synopsis:

[*count***] i**

Enter text input mode before the current cursor position. No characters already in the edit buffer shall be affected by this command. A count shall cause the input text to be appended count -1 more times to the end of the input.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Insert at Beginning of Line

Synopsis:

[*count***] I**

This command shall be equivalent to the vi command ^[count] **i**.

Join

Synopsis:

[*count***] J**

If the current line is the last line in the edit buffer, it shall be an error.

This command shall be equivalent to the ex **join** command with no addresses, and an ex command count value of 1 if count was not specified or if a count of 1 was specified, and an ex command count value of count -1 for any other value of count, except that the current line and column shall be set as follows.

Current line: Unchanged.

Current column: The last column in which any portion of the character following the last character in the initial line is displayed, or the last non- <newline> in the line if no characters were appended.

Move to Bottom of Screen

Synopsis:

[*count***] L**

If the beginning of the line count less than the last line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.

2. Otherwise, the text region shall include all lines from the starting cursor line to (the last line of the display -(count -1)).

3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

1. If in open mode, this command shall set the current column to non- <blank> and do nothing else.

2. Otherwise, it shall set the current line and current column as follows.

Current line: Set to (the last line of the display -(count -1)).

Current column: Set to non- <blank>.

Mark Position

Synopsis:

```
m letter
```

This command shall be equivalent to the ex **mark** command with the specified character as an argument.

Move to Middle of Screen

Synopsis:

```
M
```

The middle line of the display shall be calculated as follows:

```
(the top line of the display) + (((number of lines displayed) +1) /2) -1
```

If used as a motion command:

1. If in open mode, the text region shall be the current line.

2. Otherwise, the text region shall include all lines from the starting cursor line up to and including the middle line of the display.

3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

If in open mode, this command shall set the current column to non- <blank> and do nothing else.

Otherwise, it shall set the current line and current column as follows.

Current line: Set to the middle line of the display.

Current column: Set to non- <blank>.

Repeat Regular Expression Find (Forward)

Synopsis:

```
n
```

If the remembered search direction was forward, the **n** command shall be equivalent to the vi **/** command with no characters entered by the user. Otherwise, it shall be equivalent to the vi **?** command with no characters entered by the user.

If the **n** command is used as a motion command for the **!** command, the editor shall not enter text input mode on the last line on the screen, and shall behave as if the user entered a single **'!'** character as the text input.

Repeat Regular Expression Find (Reverse)

Synopsis:

```
N
```

Scan for the next match of the last pattern given to **/** or **?**, but in the reverse direction; this is the reverse of **n**.

If the remembered search direction was forward, the **N** command shall be equivalent to the vi **?** command with no characters entered by the user. Otherwise, it shall be equivalent to the vi **/** command with no characters entered by the user. If the **N** command is used as a motion command for the **!** command, the editor shall not enter text input mode on the last line on the screen, and shall behave as if the user entered a single **!** character as the text input.

Insert Empty Line Below

Synopsis:

o

Enter text input mode in a new line appended after the current line. A count shall cause the input text to be appended count -1 more times to the end of the already added text, each time starting on a new, appended line.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Insert Empty Line Above

Synopsis:

0

Enter text input mode in a new line inserted before the current line. A count shall cause the input text to be appended count -1 more times to the end of the already added text, each time starting on a new, appended line.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Put from Buffer Following

Synopsis:

[*buffer***] p**

If no buffer is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be appended below the current line, and each line of the buffer shall become a new line in the edit buffer. A count shall cause the buffer text to be appended count -1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be appended into the current line after the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A count shall cause the buffer text to be appended count -1 more times to the end of the

already added text, each time starting after the last added character.

Current line: If the buffer text is in line mode, set the line to line +1; otherwise, unchanged.

Current column: If the buffer text is in line mode:

1. If there is a non- <blank> in the first line of the buffer, set to the last column on which any portion of the first non- <blank> in the line is displayed.

2. If there is no non- <blank> in the first line of the buffer, set to the last column on which any portion of the last non- <newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the text in the buffer is from more than a single line, then set to the last column on which any portion of the first character from the buffer is displayed.

2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.

3. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

Put from Buffer Before

Synopsis:

[*buffer***] P**

If no buffer is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be inserted above the current line, and each line of the buffer shall become a new line in the edit buffer. A count shall cause the buffer text to be appended count -1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be inserted into the current line before the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A count shall cause the buffer text to be appended count -1 more times to the end of the already added text, each time starting after the last added character.

Current line: Unchanged.

Current column: If the buffer text is in line mode:

1. If there is a non- <blank> in the first line of the buffer, set to the last column on which any portion of that character is displayed.

2. If there is no non- <blank> in the first line of the buffer, set to the last column on which any portion of the last non- <newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.

2. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

Enter ex Mode

Synopsis:

```
Q
```

Leave visual or open mode and enter ex command mode.

Current line: Unchanged.

Current column: Unchanged.

Replace Character

Synopsis:

[*count***] r** *character*

Replace the count characters at and after the cursor with the specified character. If there are less than count non- <newline>s at and after the cursor on the line, it shall be an error.

If character is <control>-V, any next character other than the <newline> shall be stripped of any special meaning and used as a literal character.

If character is <ESC>, no replacement shall be made and the current line and current column shall be unchanged.

If character is <carriage-return> or <newline>, count new lines shall be appended to the current line. All but the last of these lines shall be empty. count characters at and after the cursor shall be discarded, and any remaining characters after the cursor in the current line shall be moved to the last of the new lines. If the **autoindent** edit option is set, they shall be preceded by the same number of **autoindent** characters found on the line from which the command was executed.

Current line: Unchanged unless the replacement character is a <carriage-return> or <newline>, in which case it shall be set to line + count.

Current column: Set to the last column position on which a portion of the last replaced character is displayed, or if the replacement character caused new lines to be created, set to non- <blank>.

Replace Characters

Synopsis:

R

Enter text input mode at the current cursor position possibly replacing text on the current line. A count shall cause the input text to be appended count -1 more times to the end of the input.

Current line/column: As specified for the text input commands (see Input Mode Commands in vi).

Substitute Character

Synopsis:

[*buffer***][***count***] s**

This command shall be equivalent to the vi command:

[*buffer***][***count***] c<space>**

Substitute Lines

Synopsis:

[*buffer***][***count***] S**

This command shall be equivalent to the vi command:

[*buffer***][***count***] c_**

Move Cursor to Before Character (Forward)

Synopsis:

[*count***] t** *character*

It shall be an error if count occurrences of the character do not occur after the cursor in the line.

If used as a motion command:

1. The text region shall be from the cursor up to but not including the countth occurrence of the specified character after the cursor.

2.
Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to the last column in which any portion of the character before the countth occurrence of the specified character after the cursor appears in the line.

Move Cursor to After Character (Reverse)

Synopsis:

```
[count] T character
```

It shall be an error if count occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1.
If the character before the cursor is the specified character, it shall be an error.

2.
The text region shall be from the character before the cursor up to but not including the countth occurrence of the specified character before the cursor.

3.
Any text copied to a buffer shall be in character mode.

If not used as a motion command:

Current line: Unchanged.

Current column: Set to the last column in which any portion of the character after the countth occurrence of the specified character before the cursor appears in the line.

Undo

Synopsis:

u

This command shall be equivalent to the ex **undo** command except that the current line and current column shall be set as follows:

Current line: Set to the first line added or changed if any; otherwise, move to the line preceding any deleted text if one exists; otherwise, move to line 1.

Current column: If undoing an ex command, set to the first non- <blank>.

Otherwise, if undoing a text input command:

1. If the command was a **C**, **c**, **O**, **o**, **R**, **S**, or **s** command, the current column shall be set to the value it held when the text input command was entered.

2. Otherwise, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non- <newline>s follow the text deleted from this line, set to the last column in which any portion of the last non- <newline> in the line is displayed, or 1 if the line is empty.

Otherwise, if a single line was modified (that is, not added or deleted) by the **u** command:

1. If text was added or changed, set to the last column in which any portion of the first character added or changed is displayed.

2. If text was deleted, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non- <newline>s follow the deleted text, set to the last column in which any portion of the last non- <newline> in the line is displayed, or 1 if the line is empty.

Otherwise, set to non- <blank>.

Undo Current Line

Synopsis:

U

Restore the current line to its state immediately before the most recent time that it became the current line.

Current line: Unchanged.

Current column: Set to the first column in the line in which any portion of the first character in the line is displayed.

Move to Beginning of Word

Synopsis:

```
[count] w
```

With the exception that words are used as the delimiter instead of bigwords, this command shall be equivalent to the **W** command.

Move to Beginning of Bigword

Synopsis:

```
[count] W
```

If the edit buffer is empty, it shall be an error. If there are less than count bigwords between the cursor and the end of the edit buffer, count shall be adjusted to move the cursor to the last bigword in the edit buffer.

If used as a motion command:

1. If the associated command is **c**, count is 1, and the cursor is on a <blank>, the region of text shall be the current character and no further action shall be taken.

2. If there are less than count bigwords between the cursor and the end of the edit buffer, then

the command shall succeed, and the region of text shall include the last character of the edit buffer.

3. If there are <blank>s or an end-of-line that precede the countth bigword, and the associated command is **c**, the region of text shall be up to and including the last character before the preceding <blank>s or end-of-line.

4. If there are <blank>s or an end-of-line that precede the bigword, and the associated command is **d** or **y**, the region of text shall be up to and including the last <blank> before the start of the bigword or end-of-line.

5. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

1. If the cursor is on the last character of the edit buffer, it shall be an error.

Current line: Set to the line containing the current column.

Current column: Set to the last column in which any part of the first character of the countth next bigword is displayed.

Delete Character at Cursor

Synopsis:

[*buffer***][***count***] x**

Delete the count characters at and after the current character into buffer, if specified, and into the unnamed buffer.

If the line is empty, it shall be an error. If there are less than count non- <newline>s at and after the cursor on the current line, count shall be adjusted to the number of non- <newline>s at and after the cursor.

Current line: Unchanged.

Current column: If the line is empty, set to column position 1. Otherwise, if there were count or less non- <newline>s at and after the cursor on the current line, set to the last column that displays any part of the last non- <newline> of the line. Otherwise, unchanged.

Delete Character Before Cursor

Synopsis:

[*buffer***][***count***] X**

Delete the count characters before the current character into buffer, if specified, and into the unnamed buffer.

If there are no characters before the current character on the current line, it shall be an error. If there are less than count previous characters on the current line, count shall be adjusted to the number of previous characters on the line.

Current line: Unchanged.

Current column: Set to (current column - the width of the deleted characters).

Yank

Synopsis:

[*buffer***][***count***] y** *motion*

Copy (yank) the region of text into buffer, if specified, and into the unnamed buffer.

If the motion command is the **y** command repeated:

1. The buffer shall be in line mode.

2. If there are less than count -1 lines after the current line in the edit buffer, it shall be an error.

3. The text region shall be from the current line up to and including the next count -1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

Current line: If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

Current column:

1. If the motion was from the current cursor position toward the end of the edit buffer, unchanged.

2. Otherwise, if the current line is empty, set to column position 1.

3. Otherwise, set to the last column that displays any part of the first character in the file that is part of the text region specified by the motion command.

Yank Current Line

Synopsis:

[*buffer***][***count***] Y**

This command shall be equivalent to the vi command:

[*buffer***][***count***] y_**

Redraw Window

If in open mode, the **z** command shall have the Synopsis:

Synopsis:

[*count***] z**

If count is not specified, it shall default to the **window** edit option -1. The **z** command shall be equivalent to the ex **z** command, with a type character of **=** and a count of count -2, except that the current line and current column shall be set as follows, and the **window** edit option shall not be affected. If the calculation for the count argument would result in a negative number, the count argument to the ex **z** command shall be zero. A blank line shall be written after the last line is written.

Current line: Unchanged.

Current column: Unchanged.

If not in open mode, the **z** command shall have the following Synopsis:

Synopsis:

[*line***] z [***count***]** *character*

If line is not specified, it shall default to the current line. If line is specified, but is greater than the number of lines in the edit buffer, it shall default to the number of lines in the edit buffer.

If count is specified, the value of the **window** edit option shall be set to count (as described in the ex **window** command), and the screen shall be redrawn.

line shall be placed as specified by the following characters:

<newline>, <carriage-return>

Place the beginning of the line on the first line of the display.

.

Place the beginning of the line in the center of the display. The middle line of the display shall be calculated as described for the **M** command.

-

Place an unspecified portion of the line on the last line of the display.

+

If line was specified, equivalent to the <newline> case. If line was not specified, display a screen where the first line of the display shall be (current last line) +1. If there are no lines after the last line in the display, it shall be an error.

^

If line was specified, display a screen where the last line of the display shall contain an unspecified portion of the first line of a display that had an unspecified portion of the specified line on the last line of the display. If this calculation results in a line before the

beginning of the edit buffer, display the first screen of the edit buffer.

Otherwise, display a screen where the last line of the display shall contain an unspecified portion of (current first line -1). If this calculation results in a line before the beginning of the edit buffer, it shall be an error.

Current line: If line and the **'^'** character were specified:

1. If the first screen was displayed as a result of the command attempting to display lines before the beginning of the edit buffer: if the first screen was already displayed, unchanged; otherwise, set to (current first line -1).

2. Otherwise, set to the last line of the display.

If line and the **'+'** character were specified, set to the first line of the display.

Otherwise, if line was specified, set to line.

Otherwise, unchanged.

Current column: Set to non- <blank>.

Exit

Synopsis:

ZZ

This command shall be equivalent to the ex **xit** command with no addresses, trailing **!**, or filename (see the ex **xit** command).

Input Mode Commands in vi

In text input mode, the current line shall consist of zero or more of the following categories, plus the terminating <newline>:

1. Characters preceding the text input entry point

Characters in this category shall not be modified during text input mode.

2.

autoindent characters

autoindent characters shall be automatically inserted into each line that is created in text input mode, either as a result of entering a <newline> or <carriage-return> while in text input mode, or as an effect of the command itself; for example, **O** or **o** (see the ex **autoindent** command), as if entered by the user.

It shall be possible to erase **autoindent** characters with the <control>-D command; it is unspecified whether they can be erased by <control>-H, <control>-U, and <control>-W characters. Erasing any **autoindent** character turns the glyph into erase-columns and deletes the character from the edit buffer, but does not change its representation on the screen.

3.

Text input characters

Text input characters are the characters entered by the user. Erasing any text input character turns the glyph into erase-columns and deletes the character from the edit buffer, but does not change its representation on the screen.

Each text input character entered by the user (that does not have a special meaning) shall be treated as follows:

a.

The text input character shall be appended to the last character in the edit buffer from the first, second, or third categories.

b.

If there are no erase-columns on the screen, the text input command was the **R** command, and characters in the fifth category from the original line follow the cursor, the next such character shall be deleted from the edit buffer. If the **slowopen** edit option is not set, the corresponding glyph on the screen shall become erase-columns.

c.

If there are erase-columns on the screen, as many columns as they occupy, or as are necessary, shall be overwritten to display the text input character. (If only part of a multi-column glyph is overwritten, the remainder shall be left on the screen, and continue to be treated as erase-columns; it is unspecified whether the remainder of the glyph is modified in any way.)

d.

If additional display line columns are needed to display the text input character:

1.

If the **slowopen** edit option is set, the text input characters shall be displayed on subsequent display line columns, overwriting any characters displayed in those columns.

2. Otherwise, any characters currently displayed on or after the column on the display line where the text input character is to be displayed shall be pushed ahead the number of display line columns necessary to display the rest of the text input character.

4. Erase-columns

Erase-columns are not logically part of the edit buffer, appearing only on the screen, and may be overwritten on the screen by subsequent text input characters. When text input mode ends, all erase-columns shall no longer appear on the screen.

Erase-columns are initially the region of text specified by the **c** command (see Change); however, erasing **autoindent** or text input characters causes the glyphs of the erased characters to be treated as erase-columns.

5. Characters following the text region for the **c** command, or the text input entry point for all other commands

Characters in this category shall not be modified during text input mode, except as specified in category 3.b. for the **R** text input command, or as <blank>s deleted when a <newline> or <carriage-return> is entered.

It is unspecified whether it is an error to attempt to erase past the beginning of a line that was created by the entry of a <newline> or <carriage-return> during text input mode. If it is not an error, the editor shall behave as if the erasing character was entered immediately after the last text input character entered on the previous line, and all of the non- <newline>s on the current line shall be treated as erase-columns.

When text input mode is entered, or after a text input mode character is entered (except as specified for the special characters below), the cursor shall be positioned as follows:

1. On the first column that displays any part of the first erase-column, if one exists

2. Otherwise, if the **slowopen** edit option is set, on the first display line column after the last character in the first, second, or third categories, if one exists

3. Otherwise, the first column that displays any part of the first character in the fifth category, if one exists

4.
Otherwise, the display line column after the last character in the first, second, or third categories, if one exists

5.
Otherwise, on column position 1

The characters that are updated on the screen during text input mode are unspecified, other than that the last text input character shall always be updated, and, if the **slowopen** edit option is not set, the current cursor character shall always be updated.

The following specifications are for command characters entered during text input mode.

NUL

Synopsis:

```
NUL
```

If the first character of the text input is a NUL, the most recently input text shall be input as if entered by the user, and then text input mode shall be exited. The text shall be input literally; that is, characters are neither macro or abbreviation expanded, nor are any characters interpreted in any special manner. It is unspecified whether implementations shall support more than 256 bytes of remembered input text.

<control>-D

Synopsis:

```
<control>-D
```

The <control>-D character shall have no special meaning when in text input mode for a line-oriented command (see Command Descriptions in vi).

This command need not be supported on block-mode terminals.

If the cursor does not follow an **autoindent** character, or an **autoindent** character and a **'0'** or **'^'**

character:

1. If the cursor is in column position 1, the <control>-D character shall be discarded and no further action taken.

2. Otherwise, the <control>-D character shall have no special meaning.

If the last input character was a **'0'** , the cursor shall be moved to column position 1.

Otherwise, if the last input character was a **'^'** , the cursor shall be moved to column position 1. In addition, the **autoindent** level for the next input line shall be derived from the same line from which the **autoindent** level for the current input line was derived.

Otherwise, the cursor shall be moved back to the column after the previous shiftwidth (see the ex **shiftwidth** command) boundary.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in Input Mode Commands in vi .

Current line: Unchanged.

Current column: Set to 1 if the <control>-D was preceded by a **'^'** or **'0'** ; otherwise, set to (column -1) -((column -2) % **shiftwidth**).

<control>-H

Synopsis:

<control>-H

If in text input mode for a line-oriented command, and there are no characters to erase, text input mode shall be terminated, no further action shall be done for this command, and the current line and column shall be unchanged.

If there are characters other than **autoindent** characters that have been input on the current line before the cursor, the cursor shall move back one character.

Otherwise, if there are **autoindent** characters on the current line before the cursor, it is implementation-defined whether the <control>-H command is an error or if the cursor moves back one **autoindent** character.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-H command is an error or if it is equivalent to entering <control>-H after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in Input Mode Commands in vi .

The current erase character (see stty) shall cause an equivalent action to the <control>-H command, unless the previously inserted character was a backslash, in which case it shall be as if the literal current erase character had been inserted instead of the backslash.

Current line: Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

Current column: Set to the first column that displays any portion of the character backed up over.

<newline>

Synopsis:

<newline>

<carriage-return>

<control>-J

<control>-M

If input was part of a line-oriented command, text input mode shall be terminated and the command shall continue execution with the input provided.

Otherwise, terminate the current line. If there are no characters other than **autoindent** characters on the line, all characters on the line shall be discarded. Otherwise, it is unspecified whether the **autoindent** characters in the line are modified by entering these characters.

Continue text input mode on a new line appended after the current line. If the **slowopen** edit option is set, the lines on the screen below the current line shall not be pushed down, but the first of them shall be cleared and shall appear to be overwritten. Otherwise, the lines of the screen below the current line shall be pushed down.

If the **autoindent** edit option is set, an appropriate number of **autoindent** characters shall be added as a prefix to the line as described by the ex **autoindent** edit option.

All columns after the cursor that are erase-columns (as described in Input Mode Commands in vi) shall be discarded.

If the **autoindent** edit option is set, all <blank>s immediately following the cursor shall be discarded.

All remaining characters after the cursor shall be transferred to the new line, positioned after any

autoindent characters.

Current line: Set to current line +1.

Current column: Set to the first column that displays any portion of the first character after the **autoindent** characters on the new line, if any, or the first column position after the last **autoindent** character, if any, or column position 1.

<control>-T

Synopsis:

```
<control>-T
```

The <control>-T character shall have no special meaning when in text input mode for a line-oriented command (see Command Descriptions in vi).

This command need not be supported on block-mode terminals.

Behave as if the user entered the minimum number of <blank>s necessary to move the cursor forward to the column position after the next **shiftwidth** (see the ex **shiftwidth** command) boundary.

Current line: Unchanged.

Current column: Set to column + **shiftwidth** - ((column -1) % **shiftwidth**).

<control>-U

Synopsis:

```
<control>-U
```

If there are characters other than **autoindent** characters that have been input on the current line before the cursor, the cursor shall move to the first character input after the **autoindent** characters.

Otherwise, if there are **autoindent** characters on the current line before the cursor, it is implementation-defined whether the <control>-U command is an error or if the cursor moves to the first column position on the line.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-U command is an error or if it is equivalent to entering <control>-U after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in Input Mode Commands in vi .

The current kill character (see stty) shall cause an equivalent action to the <control>-U command, unless the previously inserted character was a backslash, in which case it shall be as if the literal current kill character had been inserted instead of the backslash.

Current line: Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

Current column: Set to the first column that displays any portion of the last character backed up over.

<control>-V

Synopsis:

<control>-V

<control>-Q

Allow the entry of any subsequent character, other than <control>-J or the <newline>, as a literal character, removing any special meaning that it may have to the editor in text input mode. If a <control>-V or <control>-Q is entered before a <control>-J or <newline>, the <control>-V or <control>-Q character shall be discarded, and the <control>-J or <newline> shall behave as described in the <newline> command character during input mode.

For purposes of the display only, the editor shall behave as if a **'^'** character was entered, and the cursor shall be positioned as if overwriting the **'^'** character. When a subsequent character is entered, the editor shall behave as if that character was entered instead of the original <control>-V or <control>-Q character.

Current line: Unchanged.

Current column: Unchanged.

<control>-W

Synopsis:

<control>-W

If there are characters other than **autoindent** characters that have been input on the current line before the cursor, the cursor shall move back over the last word preceding the cursor (including any <blank>s between the end of the last word and the current cursor); the cursor shall not move to before the first character after the end of any **autoindent** characters.

Otherwise, if there are **autoindent** characters on the current line before the cursor, it is implementation-defined whether the <control>-W command is an error or if the cursor moves to the first column position on the line.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-W command is an error or if it is equivalent to entering <control>-W after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in Input Mode Commands in vi .

Current line: Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

Current column: Set to the first column that displays any portion of the last character backed up over.

<ESC>

Synopsis:

<ESC>

If input was part of a line-oriented command:

1. If interrupt was entered, text input mode shall be terminated and the editor shall return to command mode. The terminal shall be alerted.

2. If <ESC> was entered, text input mode shall be terminated and the command shall continue execution with the input provided.

Otherwise, terminate text input mode and return to command mode.

Any **autoindent** characters entered on newly created lines that have no other non- <newline>s shall be deleted.

Any leading **autoindent** and <blank>s on newly created lines shall be rewritten to be the minimum

number of <blank>s possible.

The screen shall be redisplayed as necessary to match the contents of the edit buffer.

Current line: Unchanged.

Current column:

1. If there are text input characters on the current line, the column shall be set to the last column where any portion of the last text input character is displayed.

2. Otherwise, if a character is displayed in the current column, unchanged.

3. Otherwise, set to column position 1.

EXIT STATUS

The following exit values shall be returned:

0
: Successful completion.

>0
: An error occurred.

CONSEQUENCES OF ERRORS

When any error is encountered and the standard input is not a terminal device file, vi shall not write the file or return to command or text input mode, and shall terminate with a non-zero exit status.

Otherwise, when an unrecoverable error is encountered it shall be equivalent to a SIGHUP asynchronous event.

Otherwise, when an error is encountered, the editor shall behave as specified in Command Descriptions in vi .

The following sections are informative.

APPLICATION USAGE

None.

EXAMPLES

None.

RATIONALE

See the RATIONALE for ex for more information on vi. Major portions of the vi utility specification point to ex to avoid inadvertent divergence. While ex and vi have historically been implemented as a single utility, this is not required by IEEE Std 1003.1-2001.

It is recognized that portions of vi would be difficult, if not impossible, to implement satisfactorily on a block-mode terminal, or a terminal without any form of cursor addressing, thus it is not a mandatory requirement that such features should work on all terminals. It is the intention, however, that a vi implementation should provide the full set of capabilities on all terminals capable of supporting them.

Historically, vi exited immediately if the standard input was not a terminal. IEEE Std 1003.1-2001 permits, but does not require, this behavior. An end-of-file condition is not equivalent to an end-of-file character. A common end-of-file character, <control>-D, is historically a vi command.

The text in the STDOUT section reflects the usage of the verb display in this section; some implementations of vi use standard output to write to the terminal, but IEEE Std 1003.1-2001 does not require that to be the case.

Historically, implementations reverted to open mode if the terminal was incapable of supporting full visual mode. IEEE Std 1003.1-2001 requires this behavior. Historically, the open mode of vi behaved roughly equivalently to the visual mode, with the exception that only a single line from the edit buffer (one "buffer line") was kept current at any time. This line was normally displayed on the next-to-last line of a terminal with cursor addressing (and the last line performed its normal visual functions for line-oriented commands and messages). In addition, some few commands behaved differently in open mode than in visual mode. IEEE Std 1003.1-2001 requires conformance to historical practice.

Historically, ex and vi implementations have expected text to proceed in the usual European/Latin order of left to right, top to bottom. There is no requirement in IEEE Std 1003.1-2001 that this be the case. The specification was deliberately written using words like "before", "after", "first", and "last" in order to permit implementations to support the natural text order of the language.

Historically, lines past the end of the edit buffer were marked with single tilde (**'~'**) characters; that is, if the one-based display was 20 lines in length, and the last line of the file was on line one, then lines 2-20 would contain only a single **'~'** character.

Historically, the vi editor attempted to display only complete lines at the bottom of the screen (it did display partial lines at the top of the screen). If a line was too long to fit in its entirety at the bottom of the screen, the screen lines where the line would have been displayed were displayed as single **'@'** characters, instead of displaying part of the line. IEEE Std 1003.1-2001 permits, but does not require, this behavior. Implementations are encouraged to attempt always to display a complete line at the bottom of the screen when doing scrolling or screen positioning by buffer lines.

Historically, lines marked with **'@'** were also used to minimize output to dumb terminals over slow lines; that is, changes local to the cursor were updated, but changes to lines on the screen that were not close to the cursor were simply marked with an **'@'** sign instead of being updated to match the current text. IEEE Std 1003.1-2001 permits, but does not require this feature because it is used ever less frequently as terminals become smarter and connections are faster.

Initialization in ex and vi

Historically, vi always had a line in the edit buffer, even if the edit buffer was "empty". For example:

1.

The ex command **=** executed from visual mode wrote "1" when the buffer was empty.

2. Writes from visual mode of an empty edit buffer wrote files of a single character (a <newline>), while writes from ex mode of an empty edit buffer wrote empty files.

3. Put and read commands into an empty edit buffer left an empty line at the top of the edit buffer.

For consistency, IEEE Std 1003.1-2001 does not permit any of these behaviors.

Historically, vi did not always return the terminal to its original modes; for example, ICRNL was modified if it was not originally set. IEEE Std 1003.1-2001 does not permit this behavior.

Command Descriptions in vi

Motion commands are among the most complicated aspects of vi to describe. With some exceptions, the text region and buffer type effect of a motion command on a vi command are described on a case-by-case basis. The descriptions of text regions in IEEE Std 1003.1-2001 are not intended to imply direction; that is, an inclusive region from line n to line n+5 is identical to a region from line n +5 to line n. This is of more than academic interest-movements to marks can be in either direction, and, if the **wrapscan** option is set, so can movements to search points. Historically, lines are always stored into buffers in text order; that is, from the start of the edit buffer to the end.
IEEE Std 1003.1-2001 requires conformance to historical practice.

Historically, command counts were applied to any associated motion, and were multiplicative to any supplied motion count. For example, **2cw** is the same as **c2w**, and **2c3w** is the same as **c6w**.
IEEE Std 1003.1-2001 requires this behavior. Historically, vi commands that used bigwords, words, paragraphs, and sentences as objects treated groups of empty lines, or lines that contained only <blank>s, inconsistently. Some commands treated them as a single entity, while others treated each line separately. For example, the **w**, **W**, and **B** commands treated groups of empty lines as individual words; that is, the command would move the cursor to each new empty line. The **e** and **E** commands treated groups of empty lines as a single word; that is, the first use would move past the group of lines. The **b** command would just beep at the user, or if done from the start of the line as a motion command, fail in unexpected ways. If the lines contained only (or ended with) <blank>s, the **w** and **W** commands would just beep at the user, the **E** and **e** commands would treat the group as a single word, and the **B** and **b** commands would treat the lines as individual words. For consistency and simplicity of specification, IEEE Std 1003.1-2001 requires that all vi commands treat groups of empty or blank lines as a single entity, and that movement through lines ending with <blank>s be consistent with other movements.

Historically, vi documentation indicated that any number of double quotes were skipped after punctuation marks at sentence boundaries; however, implementations only skipped single quotes. IEEE Std 1003.1-2001 requires both to be skipped.

Historically, the first and last characters in the edit buffer were word boundaries. This historical practice is required by IEEE Std 1003.1-2001.

Historically, vi attempted to update the minimum number of columns on the screen possible, which could lead to misleading information being displayed. IEEE Std 1003.1-2001 makes no requirements other than that the current character being entered is displayed correctly, leaving all other decisions in this area up to the implementation.

Historically, lines were arbitrarily folded between columns of any characters that required multiple column positions on the screen, with the exception of tabs, which terminated at the right-hand margin. IEEE Std 1003.1-2001 permits the former and requires the latter. Implementations that do not arbitrarily break lines between columns of characters that occupy multiple column positions should not permit the cursor to rest on a column that does not contain any part of a character.

The historical vi had a problem in that all movements were by buffer lines, not by display or screen lines. This is often the right thing to do; for example, single line movements, such as **j** or **k**, should work on buffer lines. Commands like **dj**, or **j.**, where **.** is a change command, only make sense for buffer lines. It is not, however, the right thing to do for screen motion or scrolling commands like <control>-D, <control>-F, and **H**. If the window is fairly small, using buffer lines in these cases can result in completely random motion; for example, **1** <control>-D can result in a completely changed screen, without any overlap. This is clearly not what the user wanted. The problem is even worse in the case of the **H**, **L**, and **M** commands-as they position the cursor at the first non- <blank> of the line, they may all refer to the same location in large lines, and will result in no movement at all.

In addition, if the line is larger than the screen, using buffer lines can make it impossible to display parts of the line-there are not any commands that do not display the beginning of the line in historical vi, and if both the beginning and end of the line cannot be on the screen at the same time, the user suffers. Finally, the page and half-page scrolling commands historically moved to the first non- <blank> in the new line. If the line is approximately the same size as the screen, this is inadequate because the cursor before and after a <control>-D command will refer to the same location on the screen.

Implementations of ex and vi exist that do not have these problems because the relevant commands (<control>-B, <control>-D, <control>-F, <control>-U, <control>-Y, <control>-E, **H**, **L**, and **M)** operate on display (screen) lines, not (edit) buffer lines.

IEEE Std 1003.1-2001 does not permit this behavior by default because the standard developers believed that users would find it too confusing. However, historical practice has been relaxed. For example, ex and vi historically attempted, albeit sometimes unsuccessfully, to never put part of a line on the last lines of a screen; for example, if a line would not fit in its entirety, no part of the line was displayed, and the screen lines corresponding to the line contained single '**@**' characters. This behavior is permitted, but not required by IEEE Std 1003.1-2001, so that it is possible for implementations to support long lines in small screens more reasonably without changing the commands to be oriented to the display (instead of oriented to the buffer). IEEE Std 1003.1-2001 also permits implementations to refuse to edit any edit buffer containing a line that will not fit on the screen in its entirety.

The display area (for example, the value of the **window** edit option) has historically been "grown", or expanded, to display new text when local movements are done in displays where the number of lines displayed is less than the maximum possible. Expansion has historically been the first choice, when the target line is less than the maximum possible expansion value away. Scrolling has historically been the next choice, done when the target line is less than half a display away, and otherwise, the screen was redrawn. There were exceptions, however, in that ex commands generally always caused the screen to be redrawn. IEEE Std 1003.1-2001 does not specify a standard behavior because there may be external issues, such as connection speed, the number of characters necessary to redraw as opposed to scroll, or terminal capabilities that implementations will have to accommodate.

The current line in IEEE Std 1003.1-2001 maps one-to-one to a buffer line in the file. The current column does not. There are two different column values that are described by IEEE Std 1003.1-2001. The first is the current column value as set by many of the vi commands. This value is remembered for the lifetime of the editor. The second column value is the actual position on the screen where the cursor rests. The two are not always the same. For example, when the cursor is backed by a multi-column character, the actual cursor position on the screen has historically been the last column of the character in command mode, and the first column of the character in input mode.

Commands that set the current line, but that do not set the current cursor value (for example, **j** and **k**) attempt to get as close as possible to the remembered column position, so that the cursor tends

to restrict itself to a vertical column as the user moves around in the edit buffer.
IEEE Std 1003.1-2001 requires conformance to historical practice, requiring that the display location of the cursor on the display line be adjusted from the current column value as necessary to support this historical behavior.

Historically, only a single line (and for some terminals, a single line minus 1 column) of characters could be entered by the user for the line-oriented commands; that is, **:**, **!**, **/**, or **?**.
IEEE Std 1003.1-2001 permits, but does not require, this limitation.

Historically, "soft" errors in vi caused the terminal to be alerted, but no error message was displayed. As a general rule, no error message was displayed for errors in command execution in vi, when the error resulted from the user attempting an invalid or impossible action, or when a searched-for object was not found. Examples of soft errors included **h** at the left margin, <control>-B or **[[** at the beginning of the file, **2G** at the end of the file, and so on. In addition, errors such as **%**, **]]**, **}**, **)**, **N**, **n**, **f**, **F**, **t**, and **T** failing to find the searched-for object were soft as well. Less consistently, **/** and **?** displayed an error message if the pattern was not found, **/**, **?**, **N**, and **n** displayed an error message if no previous regular expression had been specified, and **;** did not display an error message if no previous **f**, **F**, **t**, or **T** command had occurred. Also, behavior in this area might reasonably be based on a runtime evaluation of the speed of a network connection. Finally, some implementations have provided error messages for soft errors in order to assist naive users, based on the value of a verbose edit option. IEEE Std 1003.1-2001 does not list specific errors for which an error message shall be displayed. Implementations should conform to historical practice in the absence of any strong reason to diverge.

Page Backwards

The <control>-B and <control>-F commands historically considered it an error to attempt to page past the beginning or end of the file, whereas the <control>-D and <control>-U commands simply moved to the beginning or end of the file. For consistency, IEEE Std 1003.1-2001 requires the latter behavior for all four commands. All four commands still consider it an error if the current line is at the beginning (<control>-B, <control>-U) or end (<control>-F, <control>-D) of the file.
Historically, the <control>-B and <control>-F commands skip two lines in order to include overlapping lines when a single command is entered. This makes less sense in the presence of a count, as there will be, by definition, no overlapping lines. The actual calculation used by historical implementations of the vi editor for <control>-B was:

```
((current first line) - count x (window edit option)) +2
```

and for <control>-F was:

```
((current first line) + count x (window edit option)) -2
```

This calculation does not work well when intermixing commands with and without counts; for example, **3** <control>-F is not equivalent to entering the <control>-F command three times, and is not reversible by entering the <control>-B command three times. For consistency with other vi commands that take counts, IEEE Std 1003.1-2001 requires a different calculation.

Scroll Forward

The 4BSD and System V implementations of vi differed on the initial value used by the **scroll** command. 4BSD used:

```
((window edit option) +1) /2
```

while System V used the value of the **scroll** edit option. The System V version is specified by IEEE Std 1003.1-2001 because the standard developers believed that it was more intuitive and permitted the user a method of setting the scroll value initially without also setting the number of lines that are displayed.

Scroll Forward by Line

Historically, the <control>-E and <control>-Y commands considered it an error if the last and first lines, respectively, were already on the screen. IEEE Std 1003.1-2001 requires conformance to historical practice. Historically, the <control>-E and <control>-Y commands had no effect in open mode. For simplicity and consistency of specification, IEEE Std 1003.1-2001 requires that they behave as usual, albeit with a single line screen.

Clear and Redisplay

The historical <control>-L command refreshed the screen exactly as it was supposed to be currently displayed, replacing any **'@'** characters for lines that had been deleted but not updated on the screen with refreshed **'@'** characters. The intent of the <control>-L command is to refresh when the screen has been accidentally overwritten; for example, by a **write** command from another user, or modem noise.

Redraw Screen

The historical <control>-R command redisplayed only when necessary to update lines that had been deleted but not updated on the screen and that were flagged with **'@'** characters. There is no requirement that the screen be in any way refreshed if no lines of this form are currently displayed. IEEE Std 1003.1-2001 permits implementations to extend this command to refresh lines on the screen flagged with **'@'** characters because they are too long to be displayed in the current framework; however, the current line and column need not be modified.

Search for tagstring

Historically, the first non- <blank> at or after the cursor was the first character, and all subsequent characters that were word characters, up to the end of the line, were included. For example, with the cursor on the leading space or on the **'#'** character in the text **"#bar@"** , the tag was **"#bar"** . On the character **'b'** it was **"bar"** , and on the **'a'** it was **"ar"** . IEEE Std 1003.1-2001 requires this behavior.

Replace Text with Results from Shell Command

Historically, the **<**, **>**, and **!** commands considered most cursor motions other than line-oriented motions an error; for example, the command **>/foo<CR>** succeeded, while the command **>l** failed, even though the text region described by the two commands might be identical. For consistency, all three commands only consider entire lines and not partial lines, and the region is defined as any line that contains a character that was specified by the motion.

Move to Matching Character

Other matching characters have been left implementation-defined in order to allow extensions such as matching **'<'** and **'>'** for searching HTML, or **#ifdef**, **#else**, and **#endif** for searching C source.

Repeat Substitution

IEEE Std 1003.1-2001 requires that any **c** and **g** flags specified to the previous substitute command be ignored; however, the **r** flag may still apply, if supported by the implementation.

Return to Previous (Context or Section)

The **[[**, **]]**, **(**, **)**, **{**, and **}** commands are all affected by "section boundaries", but in some historical implementations not all of the commands recognize the same section boundaries. This is a bug, not a feature, and a unique section-boundary algorithm was not described for each command. One special case that is preserved is that the sentence command moves to the end of the last line of the edit buffer while the other commands go to the beginning, in order to preserve the traditional character cut semantics of the sentence command. Historically, vi section boundaries at the beginning and end of the edit buffer were the first non- <blank> on the first and last lines of the edit buffer if one exists; otherwise, the last character of the first and last lines of the edit buffer if one exists. To increase consistency with other section locations, this has been simplified by IEEE Std 1003.1-2001 to the first character of the first and last lines of the edit buffer, or the first and the last lines of the edit buffer if they are empty.

Sentence boundaries were problematic in the historical vi. They were not only the boundaries as defined for the section and paragraph commands, but they were the first non- <blank> that occurred after those boundaries, as well. Historically, the vi section commands were documented as taking an optional window size as a count preceding the command. This was not implemented in historical versions, so IEEE Std 1003.1-2001 requires that the count repeat the command, for consistency with other vi commands.

Repeat

Historically, mapped commands other than text input commands could not be repeated using the **period** command. IEEE Std 1003.1-2001 requires conformance to historical practice.

The restrictions on the interpretation of special characters (for example, <control>-H) in the repetition of text input mode commands is intended to match historical practice. For example, given the input sequence:

```
iab<control>-H<control>-H<control>-Hdef<escape>
```

the user should be informed of an error when the sequence is first entered, but not during a command repetition. The character <control>-T is specifically exempted from this restriction. Historical implementations of vi ignored <control>-T characters that were input in the original command during command repetition. IEEE Std 1003.1-2001 prohibits this behavior.

Find Regular Expression

Historically, commands did not affect the line searched to or from if the motion command was a search (**/**, **?**, **N**, **n**) and the final position was the start/end of the line. There were some special cases and vi was not consistent. IEEE Std 1003.1-2001 does not permit this behavior, for consistency. Historical implementations permitted but were unable to handle searches as motion commands that wrapped (that is, due to the edit option **wrapscan**) to the original location. IEEE Std 1003.1-2001 requires that this behavior be treated as an error.

Historically, the syntax **"/RE/0"** was used to force the command to cut text in line mode.
IEEE Std 1003.1-2001 requires conformance to historical practice.

Historically, in open mode, a **z** specified to a search command redisplayed the current line instead of displaying the current screen with the current line highlighted. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Historically, trailing **z** commands were permitted and ignored if entered as part of a search used as a motion command. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Execute an ex Command

Historically, vi implementations restricted the commands that could be entered on the colon command line (for example, **append** and **change**), and some other commands were known to cause them to fail catastrophically. For consistency, IEEE Std 1003.1-2001 does not permit these restrictions. When executing an ex command by entering **:**, it is not possible to enter a <newline> as part of the command because it is considered the end of the command. A different approach is to enter ex command mode by using the vi **Q** command (and later resuming visual mode with the ex **vi** command). In ex command mode, the single-line limitation does not exist. So, for example, the following is valid:

```
Q
s/break here/break\
here/
vi
```

IEEE Std 1003.1-2001 requires that, if the ex command overwrites any part of the screen that would be erased by a refresh, vi pauses for a character from the user. Historically, this character could be any character; for example, a character input by the user before the message appeared, or even a mapped character. This is probably a bug, but implementations that have tried to be more rigorous by requiring that the user enter a specific character, or that the user enter a character after the message was displayed, have been forced by user indignation back into historical behavior.
IEEE Std 1003.1-2001 requires conformance to historical practice.

Shift Left (Right)

Refer to the Rationale for the **!** and **/** commands. Historically, the **<** and **>** commands sometimes moved the cursor to the first non- <blank> (for example if the command was repeated or with **_** as the motion command), and sometimes left it unchanged. IEEE Std 1003.1-2001 does not permit this inconsistency, requiring instead that the cursor always move to the first non- <blank>. Historically, the **<** and **>** commands did not support buffer arguments, although some implementations allow the specification of an optional buffer. This behavior is neither required nor disallowed by IEEE Std 1003.1-2001.

Execute

Historically, buffers could execute other buffers, and loops, infinite and otherwise, were possible. IEEE Std 1003.1-2001 requires conformance to historical practice. The * buffer syntax of ex is not required in vi, because it is not historical practice and has been used in some vi implementations to support additional scripting languages.

Reverse Case

Historically, the ~ command ignored any associated count, and acted only on the characters in the current line. For consistency with other vi commands, IEEE Std 1003.1-2001 requires that an associated count act on the next count characters, and that the command move to subsequent lines if warranted by count, to make it possible to modify large pieces of text in a reasonably efficient manner. There exist vi implementations that optionally require an associated motion command for the ~ command. Implementations supporting this functionality are encouraged to base it on the **tildedop** edit option and handle the text regions and cursor positioning identically to the **yank** command.

Append

Historically, counts specified to the **A**, **a**, **I**, and **i** commands repeated the input of the first line count times, and did not repeat the subsequent lines of the input text. IEEE Std 1003.1-2001 requires that the entire text input be repeated count times.

Move Backward to Preceding Word

Historically, vi became confused if word commands were used as motion commands in empty files. IEEE Std 1003.1-2001 requires that this be an error. Historical implementations of vi had a large number of bugs in the word movement commands, and they varied greatly in behavior in the presence of empty lines, "words" made up of a single character, and lines containing only <blank>s. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Change to End-of-Line

Some historical implementations of the **C** command did not behave as described by IEEE Std 1003.1-2001 when the **$** key was remapped because they were implemented by pushing the **$** key onto the input queue and reprocessing it. IEEE Std 1003.1-2001 does not permit this behavior. Historically, the **C**, **S**, and **s** commands did not copy replaced text into the numeric buffers. For consistency and simplicity of specification, IEEE Std 1003.1-2001 requires that they behave like their respective **c** commands in all respects.

Delete

Historically, lines in open mode that were deleted were scrolled up, and an @ glyph written over the beginning of the line. In the case of terminals that are incapable of the necessary cursor motions, the editor erased the deleted line from the screen. IEEE Std 1003.1-2001 requires conformance to historical practice; that is, if the terminal cannot display the '@' character, the line cannot remain on the screen.

Delete to End-of-Line

Some historical implementations of the **D** command did not behave as described by IEEE Std 1003.1-2001 when the **$** key was remapped because they were implemented by pushing the **$** key onto the input queue and reprocessing it. IEEE Std 1003.1-2001 does not permit this behavior.

Join

An historical oddity of vi is that the commands **J**, **1J**, and **2J** are all equivalent. IEEE Std 1003.1-2001 requires conformance to historical practice. The vi **J** command is specified in terms of the ex **join** command with an ex command count value. The address correction for a count that is past the end of the edit buffer is necessary for historical compatibility for both ex and vi.

Mark Position

Historical practice is that only lowercase letters, plus '`' and '" , could be used to mark a cursor

position. IEEE Std 1003.1-2001 requires conformance to historical practice, but encourages implementations to support other characters as marks as well.

Repeat Regular Expression Find (Forward and Reverse)

Historically, the **N** and **n** commands could not be used as motion components for the **c** command. With the exception of the **cN** command, which worked if the search crossed a line boundary, the text region would be discarded, and the user would not be in text input mode. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Insert Empty Line (Below and Above)

Historically, counts to the **O** and **o** commands were used as the number of physical lines to open, if the terminal was dumb and the **slowopen** option was not set. This was intended to minimize traffic over slow connections and repainting for dumb terminals. IEEE Std 1003.1-2001 does not permit this behavior, requiring that a count to the open command behave as for other text input commands. This change to historical practice was made for consistency, and because a superset of the functionality is provided by the **slowopen** edit option.

Put from Buffer (Following and Before)

Historically, counts to the **p** and **P** commands were ignored if the buffer was a line mode buffer, but were (mostly) implemented as described in IEEE Std 1003.1-2001 if the buffer was a character mode buffer. Because implementations exist that do not have this limitation, and because pasting lines multiple times is generally useful, IEEE Std 1003.1-2001 requires that count be supported for all **p** and **P** commands.

Historical implementations of vi were widely known to have major problems in the **p** and **P** commands, particularly when unusual regions of text were copied into the edit buffer. The standard developers viewed these as bugs, and they are not permitted for consistency and simplicity of specification.

Historically, a **P** or **p** command (or an ex **put** command executed from open or visual mode) executed in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Replace Character

Historically, the **r** command did not correctly handle the erase and word erase characters as arguments, nor did it handle an associated count greater than 1 with a <carriage-return> argument, for which it replaced count characters with a single <newline>. IEEE Std 1003.1-2001 does not permit these inconsistencies.

Historically, the **r** command permitted the <control>-V escaping of entered characters, such as <ESC> and the <carriage-return>; however, it required two leading <control>-V characters instead of one. IEEE Std 1003.1-2001 requires that this be changed for consistency with the other text input commands of vi.

Historically, it is an error to enter the **r** command if there are less than count characters at or after the cursor in the line. While a reasonable and unambiguous extension would be to permit the **r** command on empty lines, it would require that too large a count be adjusted to match the number of characters at or after the cursor for consistency, which is sufficiently different from historical practice to be avoided. IEEE Std 1003.1-2001 requires conformance to historical practice.

Replace Characters

Historically, if there were **autoindent** characters in the line on which the **R** command was run, and **autoindent** was set, the first <newline> would be properly indented and no characters would be

replaced by the <newline>. Each additional <newline> would replace n characters, where n was the number of characters that were needed to indent the rest of the line to the proper indentation level. This behavior is a bug and is not permitted by IEEE Std 1003.1-2001.

Undo

Historical practice for cursor positioning after undoing commands was mixed. In most cases, when undoing commands that affected a single line, the cursor was moved to the start of added or changed text, or immediately after deleted text. However, if the user had moved from the line being changed, the column was either set to the first non- <blank>, returned to the origin of the command, or remained unchanged. When undoing commands that affected multiple lines or entire lines, the cursor was moved to the first character in the first line restored. As an example of how inconsistent this was, a search, followed by an **o** text input command, followed by an **undo** would return the cursor to the location where the **o** command was entered, but a **cw** command followed by an **o** command followed by an **undo** would return the cursor to the first non- <blank> of the line. IEEE Std 1003.1-2001 requires the most useful of these behaviors, and discards the least useful, in the interest of consistency and simplicity of specification.

Yank

Historically, the **yank** command did not move to the end of the motion if the motion was in the forward direction. It moved to the end of the motion if the motion was in the backward direction, except for the _ command, or for the **G** and ' commands when the end of the motion was on the current line. This was further complicated by the fact that for a number of motion commands, the **yank** command moved the cursor but did not update the screen; for example, a subsequent command would move the cursor from the end of the motion, even though the cursor on the screen had not reflected the cursor movement for the **yank** command. IEEE Std 1003.1-2001 requires that all **yank** commands associated with backward motions move the cursor to the end of the motion for consistency, and specifically, to make ' commands as motions consistent with search patterns as motions.

Yank Current Line

Some historical implementations of the **Y** command did not behave as described by IEEE Std 1003.1-2001 when the '_' key was remapped because they were implemented by pushing the '_' key onto the input queue and reprocessing it. IEEE Std 1003.1-2001 does not permit this behavior.

Redraw Window

Historically, the **z** command always redrew the screen. This is permitted but not required by IEEE Std 1003.1-2001, because of the frequent use of the **z** command in macros such as **map n nz.** for screen positioning, instead of its use to change the screen size. The standard developers believed that expanding or scrolling the screen offered a better interface for users. The ability to redraw the screen is preserved if the optional new window size is specified, and in the <control>-L and <control>-R commands.

The semantics of **z^** are confusing at best. Historical practice is that the screen before the screen that ended with the specified line is displayed. IEEE Std 1003.1-2001 requires conformance to historical practice.

Historically, the **z** command would not display a partial line at the top or bottom of the screen. If the partial line would normally have been displayed at the bottom of the screen, the command worked, but the partial line was replaced with '@' characters. If the partial line would normally have been displayed at the top of the screen, the command would fail. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Historically, the **z** command with a line specification of 1 ignored the command. For consistency and

simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Historically, the **z** command did not set the cursor column to the first non- <blank> for the character if the first screen was to be displayed, and was already displayed. For consistency and simplicity of specification, IEEE Std 1003.1-2001 does not permit this behavior.

Input Mode Commands in vi

Historical implementations of vi did not permit the user to erase more than a single line of input, or to use normal erase characters such as line erase, worderase, and erase to erase **autoindent** characters. As there exist implementations of vi that do not have these limitations, both behaviors are permitted, but only historical practice is required. In the case of these extensions, vi is required to pause at the **autoindent** and previous line boundaries.

Historical implementations of vi updated only the portion of the screen where the current cursor character was displayed. For example, consider the vi input keystrokes:

```
iabcd<escape>0C<tab>
```

Historically, the <tab> would overwrite the characters **"abcd"** when it was displayed. Other implementations replace only the **'a'** character with the <tab>, and then push the rest of the characters ahead of the cursor. Both implementations have problems. The historical implementation is probably visually nicer for the above example; however, for the keystrokes:

```
iabcd<ESC>0R<tab><ESC>
```

the historical implementation results in the string **"bcd"** disappearing and then magically reappearing when the <ESC> character is entered. IEEE Std 1003.1-2001 requires the former behavior when overwriting erase-columns-that is, overwriting characters that are no longer logically part of the edit buffer-and the latter behavior otherwise.

Historical implementations of vi discarded the <control>-D and <control>-T characters when they were entered at places where their command functionality was not appropriate.
IEEE Std 1003.1-2001 requires that the <control>-T functionality always be available, and that <control>-D be treated as any other key when not operating on **autoindent** characters.

NUL

Some historical implementations of vi limited the number of characters entered using the NUL input character to 256 bytes. IEEE Std 1003.1-2001 permits this limitation; however, implementations are encouraged to remove this limit.

<control>-D

See also Rationale for the input mode command <newline>. The hidden assumptions in the <control>-D command (and in the vi **autoindent** specification in general) is that <space>s take up a single column on the screen and that <tab>s are comprised of an integral number of <space>s.

<newline>

Implementations are permitted to rewrite **autoindent** characters in the line when <newline>, <carriage-return>, <control>-D, and <control>-T are entered, or when the **shift** commands are used, because historical implementations have both done so and found it necessary to do so. For example, a <control>-D when the cursor is preceded by a single <tab>, with **tabstop** set to 8, and **shiftwidth** set to 3, will result in the <tab> being replaced by several <space>s.

<control>-T

See also the Rationale for the input mode command <newline>. Historically, <control>-T only worked if no non- <blank>s had yet been input in the current input line. In addition, the characters inserted by <control>-T were treated as **autoindent** characters, and could not be erased using normal user erase characters. Because implementations exist that do not have these limitations, and as moving to a column boundary is generally useful, IEEE Std 1003.1-2001 requires that both limitations be removed.

<control>-V

Historically, vi used **^V**, regardless of the value of the literal-next character of the terminal. IEEE Std 1003.1-2001 requires conformance to historical practice.

The uses described for <control>-V can also be accomplished with <control>-Q, which is useful on terminals that use <control>-V for the down-arrow function. However, most historical implementations use <control>-Q for the termios START character, so the editor will generally not receive the <control>-Q unless **stty ixon** mode is set to off. (In addition, some historical implementations of vi explicitly set **ixon** mode to on, so it was difficult for the user to set it to off.) Any of the command characters described in IEEE Std 1003.1-2001 can be made ineffective by their selection as termios control characters, using the stty utility or other methods described in the System Interfaces volume of IEEE Std 1003.1-2001.

<ESC>

Historically, SIGINT alerted the terminal when used to end input mode. This behavior is permitted, but not required, by IEEE Std 1003.1-2001.

FUTURE DIRECTIONS

None.

SEE ALSO

ed , ex , stty

COPYRIGHT

Index

Manuals

Unique solution ID: #1013
Author: n/a
Last update: 2014-12-16 21:38