## NAME

**ex, vi, view** – text editors

## DESCRIPTION

**Vi** is a screen oriented text editor. **Ex** is a line-oriented text editor. **Ex** and **vi** are different interfaces to the same program, and it is possible to switch back and forth during an edit session. **View** is the equivalent of using the **−R** (read-only) option of **vi**.

This reference manual is the one provided with the **nex/nvi** versions of the **ex/vi** text editors. **Nex/nvi** are intended as bug-for-bug compatible replacements for the original Fourth Berkeley Software Distribution (4BSD) **ex** and **vi** programs. This reference manual is accompanied by a traditional-style manual page. That manual page describes the functionality found in **ex/vi** in far less detail than the description here. In addition, it describes the system interface to **ex/vi**, e.g. command line options, environmental variables, and similar things.

This reference is intended for users already familiar with **ex/vi**. Anyone else should almost certainly read a good tutorial on the editor first. If you're in an unfamiliar environment, and you absolutely have to get work done immediately, see the section entitled FAST STARTUP in the manual page. It's probably enough to get you going.

For the rest of this reference, **nex/nvi** is used only when it's necessary to distinguish it from the historic implementations of **ex/vi**.

## ADDITIONAL FEATURES

There are a few features in **nex/nvi** that are not found in historic versions of **ex/vi**. A list of those features is as follows:

8-bit clean data, large lines, files

> **Nvi/nex** will edit any format file. Line lengths are limited by available memory, and file sizes are limited by available disk space. The command "^Vx[0-9A-Fa-f]∗", in input mode, will insert any legal character value into the text.

Split screens

> The command ":sp[lit] [file ...]" splits the screen in vi mode. The key "^W" switches between the foreground screens, and the ":resize count" command can be used to grow or shrink a particular screen.

Background and foreground screens

> The command ":bg" backgrounds the current screen, and the command ":fg [file]" foregrounds the backgrounded screen that is editing the specified file, or, by default, the first background screen on the queue. The command ":di[splay] s[creens]" lists the background screens.

Shell screens

> The command ":sc[ript] [file ...]" runs a shell in the screen. Editing is unchanged, with the exception that a <carriage-return> enters the current line (stripped of any prompt) as input to the shell.

Tag stacks

> Tags are now maintained in a stack. The command "^T" returns to the previous tag location. The command ":tagpop [number  file]" returns to the most recent tag location by default, or, optionally to a specific tag number in the tag stack, or the most recent tag from the specified file. Use the command ":di[splay] t[ags]" to view the tags stack. The command ":tagtop" returns to the top of the tag stack.

New displays

> The command ":di[splay] b[uffers]  s[creens]  t[ags]" can be used to display, respectively, the

current cut buffers, the backgrounded screens, and the tags stack.

Infinite undo
: The changes made during an edit session may be rolled backward and forward. A '.' command immediately after a 'u' command continues either forward or backward depending on whether the 'u' command was an undo or a redo.

Usage information
: The command ":exu[sage] [cmd]" and "viu[sage] [key]" provide usage information for all of the ex and vi commands by default, or, optionally, for a specific command or key.

Extended regular expressions
: The ":set extended" command treats search and other command regular expressions as extended (egrep(1) style) regular expressions.

Word search
: The command "^A" searches for the word referenced by the cursor.

Number increment
: The command "#" increments the number referenced by the cursor.

Previous file
: The command ":prev[ious][!]" edits the previous file from the argument list.

Left-Right scrolling
: The command ":set leftright" makes **nvi** do left-right screen scrolling, instead of the traditional **vi** line wrapping.

## RECOVERY

There is no recovery program for **nvi**, nor does **nvi** run setuid. Users may recover any file which they may read, and the superuser may recover any edit session.

Edit sessions are backed by files in `/var/tmp/vi.recover`, and are named "vi.XXXX", where "XXXX" is a number related to the process id. When a file is first modified, a second file, which contains an email message for the user, is created, and is named "recover.XXXX", where, again, "XXXX" is associated with the process id. Both files are removed at the end of a normal edit session, but will remain if the edit session is abnormally terminated or the user enters the ex/vi "preserve" command. The use of the `/var/tmp` directory may be changed setting the **recdir** option in the user's or system startup information.

The recovery directory should have the "sticky-bit" set so that only the owners of files may remove them. If this is not possible on the system, then a pseudo-user should own the recovery directory. The recovery directory must be both read and write-able by any user.

The recovery file has all of the necessary information in it to enable the user to recover the edit session. In addition, it has all of the necessary email headers for sendmail. When the system is rebooted, all of the files in `/var/tmp/vi.recover` named "recover.XXXX" should be sent by email, using the **-t** flag of sendmail (or a similar mechanism in other mailers). A simple way to do this is to insert the following script into your `/etc/rc.local` (or other startup) file:

```
virecovery=`echo /var/tmp/vi.recover/recover.*`
if [ "$virecovery" != "/var/tmp/vi.recover/recover.*" ]; then
        echo 'Recovering vi editor sessions'
        for i in $virecovery; do
                sendmail -t < $i
        done
fi
```

If **ex/vi** receives a hangup (SIGHUP) signal, it will email the recovery information to the user itself.

If you don't have the sendmail program on your system, the source file `nvi/recover.c` will have to be modified to use your local mail delivery programs.

## STARTUP INFORMATION

**Ex/vi** interprets one of two possible environmental variables and reads up to three of five possible files during startup. The variables and files are expected to contain **ex** commands, not **vi** commands. In addition, they are interpreted *before* the file to be edited is read, and therefore many **ex** commands may not be used. Generally, any command that requires output to the screen or that needs a file upon which to operate, will cause an error if included in a startup file or environmental variable.

First, the file `/etc/vi.exrc` is read. Second, the environmental variable `NEXINIT` (or the variable `EXINIT`, if `NEXINIT` isn't set) is interpreted. Third, if neither `NEXINIT` or `EXINIT` was set, the file `$HOME/.nexrc` (or the file `$HOME/.exrc`, if `$HOME/.nexrc` doesn't exist) is read. Fourth, the file `.nexrc` (or the file `.exrc`, if `.nexrc` doesn't exist) is read.

Startup files will not be read if they are owned by anyone other than the real user-id of the user running **vi**, (or by "root" in the case of the file `/etc/vi.exrc`) or if they are writable by anyone other than the owner. Home directory startup files (i.e. `$HOME/.nexrc` and `$HOME/.exrc`) will not be read if the "HOME" environmental variable is not set. Local startup files (i.e. `.nexrc` and `.exrc`) will not be read if the **exrc** option is turned off in the `/etc/vi.exrc`, `$HOME/.nexrc`, or `$HOME/.exrc` files, or in the `NEXINIT` or `EXINIT` environmental variables. It is not an error for any of the startup environmental variables or files not to exist.

Because the **ex** command set supported by **nex/nvi** is a superset of the command set supported by most historical implementations of **ex**, **nex/nvi** can use the startup files created for the historical implementations, but the converse is often not true.

## SIZING THE SCREEN

The size of the screen can be set in a number of ways. **Ex/vi** takes the following steps until values are obtained for both the number of rows and number of columns in the screen.

1. If the environmental variable `LINES` exists, it is used to specify the number of rows in the screen.
2. If the environmental variable `COLUMNS` exists, it is used to specify the number of columns in the screen.
3. The TIOCGWINSZ `ioctl`(2) is attempted on the standard error file descriptor.
4. The termcap entry is checked for the "li" entry (rows) and the "co" entry (columns).
5. The number of rows is set to 24, and the number of columns is set to 80.

If a window change size signal (SIGWINCH) is received, the same steps are taken with the exception that the first two steps are skipped.

## REGULAR EXPRESSIONS AND REPLACEMENT STRINGS

Regular expressions are used in line addresses, as the first part of **substitute**, **global**, and **vglobal** commands, and in search patterns.

The regular expressions supported by **ex** and **vi** are, by default, the Basic Regular Expressions (BRE's) described in the IEEE POSIX Standard 1003.2. The **extended** option causes all regular expressions to be interpreted as the Extended Regular Expressions (ERE's) described by the same standard. (See `re_format`(7) for more information. Generally speaking, BRE's are `ed`(1) and `grep`(1) style regular expressions, and ERE's are `egrep`(1) style regular expressions.)

There are some special strings and characters that can be used in RE's:

1. An empty RE (e.g. “//”) is equivalent to the last RE used.
2. The construct “\<” matches the beginning of a word.
3. The construct “\>” matches the end of a word.
4. The character “~” matches the replacement part of the last **substitute** command.

When the **magic** option is *not* set, the only characters with special meanings are “^” at the beginning of an RE, “$” at the end of an RE, and the escaping character “\”. The characters “.”, “*”, “[”, and “~” are treated as ordinary characters unless preceded by a “\”; when preceded by a “\” they regain their special meaning.

Replacement strings are the second part of a **substitute** command.

The character “&” (or “\&” if the **magic** option is *not* set) in the replacement string stands for the text matched by the RE that's being replaced. The character “~” (or “\~” if the **magic** option is *not* set) stands for the replacement part of the previous **substitute** command.

The string “\#”, where “#” is an integer value from 1 to 9, stands for the text matched by the portion of the RE enclosed in the #'th set of escaped parentheses, e.g. “\(” and “\)”. For example, “s/abc\(.*\)def/\1/” deletes the strings “abc” and “def” from the matched pattern.

The strings “\l”, “\u”, “\L”, and “\U” can be used to modify the case of elements in the replacement string. The string “\l” causes the next character to be converted to lowercase; the string “\u” behaves similarly, but converts to uppercase. The strings “\L” causes characters up to the end of the string or the next occurrence of the strings “\e” or “\E” to be converted to lowercase; the string “\U” behaves similarly, but converts to uppercase.

In **vi**, inserting a <control-M> into the replacement string will cause the matched line to be split into two lines at that point.

## SET OPTIONS

There are a large number of options that may be set (or unset) to change the editor's behavior. This section describes the options, their abbreviations and their default values.

In each entry below, the first part of the tag line is the full name of the option, followed by any equivalent abbreviations. (Regardless of the abbreviations, it is only necessary to use the minimum number of characters necessary to distinguish an abbreviation from all other commands for it to be accepted, in **nex/nvi**. Historically, only the full name and the official abbreviations were accepted by **ex/vi**. Using full names in your startup files and environmental variables will probably make them more portable.) The part in square brackets is the default value of the option. Most of the options are boolean, i.e. they are either on or off, and do not have an associated value.

Options apply to both **ex** and **vi** modes, unless otherwise specified.

For information on modifying the options or to display the options and their current values, see the “set” command in the Ex Commands section.

`altwerase [off]`

**Vi** only. Change how **vi** does word erase during text input. When this option is set, text is broken up into three classes: alphabetic, numeric and underscore characters, other non-blank characters, and blank characters. Changing from one class to another marks the end of a word. In addition, the class of the first character erased is ignored (which is exactly what you want when erasing pathname components).

`autoindent, ai [off]`

If this option is set, whenever you create a new line (using the **vi** **A**, **a**, **C**, **c**, **I**, **i**, **O**, **o**, **R**, **r**, **S**, and **s** commands, or the **ex** **append**, **change**, and **insert** commands) the new line is automatically indented to align the cursor with the first non-blank character of the line from which you created it. Lines are indented using tab characters to the extent possible (based on the value of the **tabstop** option) and then using space characters as necessary. For commands inserting text into the middle of a line, any blank characters to the right of the cursor are discarded, and the first non-blank character to the right of the cursor is aligned as described above.

The indent characters are themselves somewhat special. If you do not enter more characters on the new line before moving moving to another line, or entering <escape>, the indent character will be deleted and the line will be empty. For example, if you enter <carriage-return> twice in succession, the line created by the first <carriage-return> will not have any characters in it, regardless of the indentation of the previous or subsequent line.

Indent characters also require that you enter additional erase characters to delete them. For example, if you have an indented line, containing only blanks, the first <word-erase> character you enter will erase up to end of the indent characters, and the second will erase back to the beginning of the line. (Historically, only the **^D** key would erase the indent characters. Both the **^D** key and the usual erase keys work in **nvi .)** In addition, if the cursor is positioned at the end of the indent characters, the keys “0^D” will erase all of the indent characters for the current line, resetting the indentation level to 0. Similarly, the keys “^^D” (i.e. a carat followed by a <control-D>) will erase all of the indent characters for the current line, leaving the indentation level for future created lines unaffected.

Finally, if **autoindent** is set, the **S** and **cc** commands change from the first non-blank of the line to the end of the line, instead of from the beginning of the line to the end of the line.

autoprint, ap [off]

**Ex** only. **Ex** only. Cause the current line to be automatically displayed after the **ex** commands **<**, **>**, **copy**, **delete**, **join**, **move**, **put**, **t**, **Undo**, and **undo**. This automatic display is suppressed during **global** and **vglobal** commands, and for any command where optional flags are used to explicitly display the line.

autowrite, aw [off]

If this option is set, the **vi ! ^^ ^]** and **^Z** commands, and the **ex edit**, **next**, **rewind**, **stop**, **suspend**, **tag**, **tagpop**, and **tagtop** commands automatically write the current file back to the current file name if it has been modified since it was last written. If the write fails, the command fails and goes no further.

Appending the optional force flag “!” to the **ex** commands **next**, **rewind**, **stop**, **suspend**, **tag**, **tagpop**, and **tagtop** stops the automatic write from being attempted.

(Historically, the **next** command ignored the optional force flag.) Note, the **ex** commands **edit**, **quit**, **shell**, and **xit** are *not* affected by the **autowrite** option.

beautify, bf [off]

If this option is set, all control characters that are not currently being specially interpreted, other than <tab>, <newline>, and <form-feed>, are discarded from commands read in by **ex** from command files, and from input text entered to **vi** (either into the file or to the colon command line). Text files read by **ex/vi** are *not* affected by the **beautify** option.

cdpath [environment variable CDPATH, or ``.'']

This option is used to specify a colon separated list of directories which are used as path prefixes for any relative path names used as arguments for the **cd** command. The value of this option defaults to the value of the environmental variable CDPATH if it is set, otherwise to the current directory. For compatibility with the POSIX 1003.2 shell, the **cd** command does *not* check the current directory as a path prefix for relative path names unless it is explicitly specified. It may be so specified by entering an empty string or a “.” into the CDPATH variable or the option value.

columns, co [80]

The number of columns in the screen. Setting this option causes **ex/vi** to set (or reset) the environmental variable `COLUMNS`. See the SCREEN SIZING section for more information.

`comment [off]`
: **Vi** only. If the first non-empty line of the file begins with the string "/*", this option causes **vi** to skip to the end of that C comment (probably a terribly boring legal notice) before displaying the file.

`directory, dir [environment variable TMPDIR, or /tmp]`
: The directory where temporary files are created. The environmental variable `TMPDIR` is used as the default value if it exists, otherwise `/tmp` is used.

`edcompatible, ed [off]`
: This option causes the presence or absence of **g** and **c** suffixes on **substitute** commands to be remembered, and to be toggled by repeating the suffices. The suffix **r** makes the substitution be as in the **~** command, instead of like the **&** command.
*This option is not yet implemented.*

`errorbells, eb [off]`
: **Ex** only. Causes **ex** error messages to be preceded by a bell.
*This option is not yet implemented.*

`exrc, ex [off]`
: If this option is turned off in the system or $HOME startup files, the local startup files are never read (unless they are the same as the system or $HOME startup files). Turning it on has no effect, i.e. the normal checks for local startup files are performed, regardless. See the STARTUP INFORMATION section for more information.

`extended [off]`
: This option causes all regular expressions to be treated as POSIX 1003.2 extended regular expressions (which are similar to historic `egrep`(1) style expressions).

`flash [on]`
: This option causes the screen to flash instead of beeping the keyboard, on error, if the terminal has the capability.

`hardtabs, ht [8]`
: This option defines the spacing between hardware tab settings, i.e. the tab expansion done by the operating system and/or the terminal itself. As **nex/nvi** never writes tabs to the terminal, unlike historic versions of **ex/vi**, this option does not currently have any affect.

`ignorecase, ic [off]`
: This option causes regular expressions, both in **ex** commands and in searches, to be evaluated in a case-insensitive manner.

`keytime [6]`
: The 10th's of a second **ex/vi** waits for a subsequent key to complete a key mapping.

`leftright [off]`
: **Vi** only. This option causes the screen to be scrolled left-right to view lines longer than the screen, instead of the traditional **vi** screen interface which folds long lines at the right-hand margin of the terminal.

`lines, li [24]`
: **Vi** only. The number of lines in the screen. Setting this option causes **ex/vi** to set (or reset) the environmental variable `LINES`. See the Screen Sizing section for more information.

`lisp [off]`
: **Vi** only. This option changes the behavior of the **vi** **(**, **)**, **{**, **}**, **[[** and **]]** commands to match the Lisp language. Also, the **autoindent** option's behavior is changed to be appropriate for Lisp.
*This option is not yet implemented.*

`list [off]`
: This option causes lines to be displayed in an unambiguous fashion. Specifically, tabs are displayed as control characters, i.e. "^I", and the ends of lines are marked with a "$" character.

`magic [on]`
: This option is on by default. Turning the **magic** option off causes all regular expression characters except for "^" and "$", to be treated as ordinary characters. To re-enable characters individually,

when the **magic** option is off, precede them with an "\". See the REGULAR EXPRESSIONS AND REPLACEMENT STRINGS section for more information.

`matchtime [7]`
: **Vi** only. The 10th's of a second **ex/vi** pauses on the matching character when the **showmatch** option is set.

`mesg [on]`
: This option allows other users to contact you using the `talk`(1) and `write`(1) utilities, while you are editing. **Ex/vi** does not turn message on, i.e. if messages were turned off when the editor was invoked, they will stay turned off. This option only permits you to disallow messages for the edit session. See the `mesg`(1) utility for more information.

`modelines, modeline [off]`
: If the **modelines** option is set, **ex/vi** has historically scanned the first and last five lines of each file as it is read for editing, looking for any **ex** commands that have been placed in those lines. After the startup information has been processed, and before the user starts editing the file, any commands embedded in the file are executed. Commands are recognized by the letters "e" or "v" followed by "x" or "i", at the beginning of a line or following a tab or space character, and followed by a ":", an **ex** command, and another ":". This option is a security problem of immense proportions, and should not be used under any circumstances.
*This option will never be implemented.*

`number, nu [off]`
: Precede each line displayed with its current line number.

`open [on]`
: **Ex** only. If this option is not set, the **open** and **visual** commands are disallowed.

`optimize, opt [on]`
: **Vi** only. Throughput of text is expedited by setting the terminal to no do automatic carriage returns when printing more than one (logical) line of output, greatly speeding output on terminals without addressable cursors when text with leading white space is printed.
*This option is not yet implemented.*

`paragraphs, para [IPLPPPQPP LIpplpipbp]`
: **Vi** only. Define additional paragraph boundaries for the **{** and **}** commands. The value of this option must be a character string consisting of zero or more character pairs.

In the text to be edited, the character string <newline>.<char-pair>, (where <char-pair> is one of the character pairs in the option's value) defines a paragraph boundary. For example, if the option were set to "LaA ##", then all of the following additional paragraph boundaries would be recognized:

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

`prompt [on]`
: **Ex** only. This option causes **ex** to prompt for command input with a ":" character; when it's not set, no prompt is displayed.

`readonly, ro [off]`
: This option causes a force flag to be required to attempt to write the file back to the original file name. Setting this option is equivalent to using the **-R** command line option, or editing a file which lacks write permission.

`recdir [/var/tmp/vi.recover]`
: The directory where recovery files are stored.

`redraw, re [off]`
: **Vi** only. The editor simulates (using great amounts of output), an intelligent terminal on a dumb terminal (e.g. during insertions in visual mode the characters to the right of the cursor are refreshed as each input character is typed).

*This option is not yet implemented.*

`remap [on]`

If this option is set, it's possible to define macros in terms of other macros. Otherwise, each key is only remapped up to one time. For example, if "A" is mapped to "B", and "B" is mapped to "C", The keystroke "A" will be mapped to "C" if **remap** is set, and to "B" if it is not set.

`remapmax [on]`

If this option is set, a key may only be remapped 50 times. If it is not set, a key may be remapped an infinite number of times, and the editor can be placed into infinite loops.

`report [5]`

Set the threshold of the number of lines that need to be changed before a message will be displayed to the user. The value is the largest value about which the editor is silent, i.e. by default, 6 lines must change before the user is notified.

`ruler [off]`

**Vi** only. Display a row/column ruler on the colon command line.

`scroll, scr [window / 2]`

Set the number of lines scrolled by the **vi** commands **^D** and **^U**.

Historically, the **ex z** command, when specified without a count, used two times the size of the scroll value; the POSIX 1003.2 standard specified the window size, which is a better choice.

`sections, sect [NHSHH HUnhsh]`

**Vi** only. Define additional section boundaries for the **[[** and **]]** commands. The **sections** option should be set to a character string consisting of zero or more character pairs. In the text to be edited, the character string <newline>.<char-pair>, (where <char-pair> is one of the character pairs in the option's value), defines a section boundary in the same manner that **paragraph** option boundaries are defined.

`shell, sh [environment variable SHELL, or /bin/sh]`

Select the shell used by the editor. The specified path is the pathname of the shell invoked by the **vi !** shell escape command and by the **ex shell** command. This program is also used to resolve any shell meta-characters in **ex** commands.

`shiftwidth, sw [8]`

Set the autoindent and shift command indentation width. This width is used by the **autoindent** option and by the **<**, **>**, and **shift** commands.

`showdirty [off]`

**Vi** only. Display an asterisk on the colon command line if the file has been modified.

`showmatch, sm [off]`

**Vi** only. This option causes **vi**, when a "}" or ")" is entered, to briefly move the cursor the matching "{" or "(". See the **matchtime** option for more information.

`showmode [off]`

**Vi** only. This option causes **vi** to display the strings "Command" or "Input" on the colon command line, based on the current mode of the editor.

`sidescroll [16]`

**Vi** only. Sets the number of columns that are shifted to the left or right, when **vi** is doing left-right scrolling and the left or right margin is crossed. See the **leftright** option for more information.

`slowopen, slow [off]`

This option affects the display algorithm used by **vi**, holding off display updating during input of new text to improve throughput when the terminal in use is slow and unintelligent.
*This option is not yet implemented.*

`sourceany [off]`

If this option is turned on, **vi** historically read startup files that were owned by someone other than the editor user. See the STARTUP INFORMATION section for more information. This option is a security problem of immense proportions, and should not be used under any circumstances.

*This option will never be implemented.*

`tabstop, ts [8]`
: This option sets tab widths for the editor display.

`taglength, tl [0]`
: This option sets the maximum number of characters that are considered significant in a tag name. Setting the value to 0 makes all of the characters in the tag name significant.

`tags, tag [tags /var/db/libc.tags /sys/kern/tags]`
: Sets the list of tags files, in search order, which are used when the editor searches for a tag.

`term, ttytype, tty [environment variable TERM]`
: Set the terminal type. Setting this option causes **ex/vi** to set (or reset) the environmental variable TERM.

`terse [off]`
: This option has historically made editor messages less verbose. It has no effect in this implementation. See the **verbose** option for more information.

`timeout, to [on]`
: If this option is set, **ex/vi** waits for a specific period for a subsequent key to complete a key mapping (see the **keytime** option). If the option is not set, the editor waits until enough keys are entered to resolve the ambiguity, regardless of how long it takes.

`ttywerase [off]`
: **Vi** only. This option changes how **vi** does word erase during text input. If this option is set, text is broken up into two classes, blank characters and non-blank characters. Changing from one class to another marks the end of a word.

`verbose [off]`
: only. **Vi** historically bells the terminal for many obvious mistakes, e.g. trying to move past the left-hand margin, or past the end of the file. If this option is set, an error message is displayed for all errors.

`w300 [no default]`
: **Vi** only. Set the window size if the baud rate is less than 1200 baud. See the **window** option for more information.

`w1200 [no default]`
: **Vi** only. Set the window size if the baud rate is equal to 1200 baud. See the **window** option for more information.

`w9600 [no default]`
: **Vi** only. Set the window size if the baud rate is greater than 1200 baud. See the **window** option for more information.

`warn [on]`
: **Ex** only. This option causes a warning message to the terminal if the file has been modified, since it was last written, before a **!** command.

`window, w, wi [environment variable LINES]`
: This option determines the default number of lines in a screenful, as written by the **z** command. It also determines the number of lines scrolled by the **vi** commands **^F** and **^B**. The value of window can be unrelated to the real screen size, although it starts out as the number of lines on the screen (see the SCREEN SIZING section). Setting the value of the **window** option is the same as using the **−w** command line option.

  If the value of **window** (as set by the **window**, **w300**, **w1200** or **w9600** options) is smaller than the actual size of the screen, large screen movements will result in displaying only that smaller number of lines on the screen. (Further movements in that same area will result in the screen being filled.) This can provide a performance improvement when viewing different places in one or more files over a slow link.

`wrapmargin, wm [0]`
: **Vi** only. If the value of wrapmargin is non-zero, **vi** will break lines, that are more than that number

of characters long, into two lines at the blank character closest to the value. If wrapmargin is 0, or if there is no blank character upon which to break the line, the line will not be broken.

`wrapscan, ws [on]`

This option causes searches to wrap around the end or the beginning of the file, and back to the starting point. Otherwise, the end or beginning of the file terminates the search.

`writeany, wa [off]`

If this option is set, file-overwriting checks that would usually be made before the **write** and **xit** commands, or before an automatic write (see the **autowrite** option), are not made. This allows a write to any file, provided the file permissions allow it.