



Linux Ubuntu 26.04 Manual Pages on command 'view.1'

\$ man view.1

VIM(1) General Commands Manual VIM(1)

NAME

vim - Vi IMproved, a programmer's text editor

SYNOPSIS

vim [options] [file ..]

vim [options] -

vim [options] -t tag

vim [options] -q [errorfile]

ex

view

gvim gview evim eview

rvim rview rgvim rgview

DESCRIPTION

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between Vim and Vi.

While running Vim a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.

Most often Vim is started to edit a single file with the command

vim file

More generally Vim is started with:

vim [options] [filelist]

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

file .. A list of filenames. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. You can get to the other files with the ":next" command. To edit a file that starts with a dash, precede the filelist with "--".

- The file to edit is read from stdin. Commands are read from stderr, which should be a tty.

-t {tag} The file to edit and the initial cursor position depends on a "tag", a sort of goto label. {tag} is looked up in the tags file, the associated file becomes the current file and the associated command is executed. Mostly this is used for C programs, in which case {tag} could be a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function. See ":help tag-commands".

-q [errorfile]

Start in quickFix mode. The file [errorfile] is read and the first error is displayed. If [errorfile] is omitted, the filename is obtained from the 'error? file' option (defaults to "AztecC.Err" for the Amiga, "errors.err" on other systems). Further errors can be jumped to with the ":cn" command. See ":help quickfix".

Vim behaves differently, depending on the name of the command (the executable may still be the same file).

vim The "normal" way, everything is default.

ex Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "-e" argument.

view Start in read-only mode. You will be protected from writing the files. Can also be done with the "-R" argument.

gvim gview

The GUI version. Starts a new window. Can also be done with the "-g" argument.

evim eview

The GUI version in easy mode. Starts a new window. Can also be done with the "-y" argument.

rvim rview rgvim rgview

Like the above, but with restrictions. It will not be possible to start shell commands, or suspend Vim. Can also be done with the "-Z" argument.

OPTIONS

The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash.

+{num} For the first file the cursor will be positioned on line "num". If "num" is missing, the cursor will be positioned on the last line.

+/{pat} For the first file the cursor will be positioned in the line with the first occurrence of {pat}. See ":help search-pattern" for the available search patterns.

+{command}

-c {command}

{command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be enclosed in double quotes (this depends on the shell that is used). Example: vim "+set si" main.c

Note: You can use up to 10 "+" or "-c" commands.

-A If Vim has been compiled with ARABIC support for editing right-to-left oriented files and Arabic keyboard mapping, this option starts Vim in Arabic mode, i.e. 'arabic' is set. Otherwise an error message is given and Vim aborts.

-b Binary mode. A few options will be set that makes it possible to edit a binary or executable file.

-C Compatible. Set the 'compatible' option. This will make Vim behave mostly like Vi, even though a .vimrc file exists.

-d Start in diff mode. There should be between two to eight file name arguments. Vim will open all the files and show differences between them. Works like vimdiff(1).

-d {device}, -dev {device}

Open {device} for use as a terminal. Only on the Amiga. Example: "-d con:20/30/600/150".

-D Debugging. Go to debugging mode when executing the first command from a script.

-e Start Vim in Ex mode, just like the executable was called "ex".

- E Start Vim in improved Ex mode, just like the executable was called "exim".
- f Foreground. For the GUI version, Vim will not fork and detach from the shell it was started in. On the Amiga, Vim is not restarted to open a new window. This option should be used when Vim is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and "!" commands will not work.
- F If Vim has been compiled with FKMAP support for editing right-to-left oriented files and Farsi keyboard mapping, this option starts Vim in Farsi mode, i.e. 'fkmap' and 'rightleft' are set. Otherwise an error message is given and Vim aborts.

Note: Farsi support has been removed in patch 8.1.0932.
- g If Vim has been compiled with GUI support, this option enables the GUI. If no GUI support was compiled in, an error message is given and Vim aborts.
- H If Vim has been compiled with RIGHTLEFT support for editing right-to-left oriented files and Hebrew keyboard mapping, this option starts Vim in Hebrew mode, i.e. 'hkmap' and 'rightleft' are set. Otherwise an error message is given and Vim aborts.
- i {viminfo}

Specifies the filename to use when reading or writing the viminfo file, instead of the default "~/.viminfo". This can also be used to skip the use of the .viminfo file, by giving the name "NONE".
- I Lisp mode. Sets the 'lisp' and 'showmatch' options on.
- L Same as -r.
- m Modifying files is disabled. Resets the 'write' option. You can still modify the buffer, but writing a file is not possible.
- M Modifications not allowed. The 'modifiable' and 'write' options will be unset, so that changes are not allowed and files can not be written. Note that these options can be set to enable making modifications.
- n No swap file will be used. Recovery after a crash will be impossible. Handy if you want to edit a file on a very slow medium (e.g. floppy). Can also be done with ":set uc=0". Can be undone with ":set uc=200".
- N No-compatible mode. Resets the 'compatible' option. This will make Vim behave a bit better, but less Vi compatible, even though a .vimrc file does not exist.

- nb Become an editor server for NetBeans. See the docs for details.
- o[N] Open N windows stacked. When N is omitted, open one window for each file.
- O[N] Open N windows side by side. When N is omitted, open one window for each file.
- p[N] Open N tab pages. When N is omitted, open one tab page for each file.

-P {parent-title}

Win32 GUI only: Specify the title of the parent application. When possible, Vim will run in an MDI window inside the application. {parent-title} must appear in the window title of the parent application. Make sure that it is specific enough. Note that the implementation is still primitive. It won't work with all applications and the menu doesn't work.

-r List swap files, with information about using them for recovery.

-r {file} Recovery mode. The swap file is used to recover a crashed editing session. The swap file is a file with the same filename as the text file with ".swp" appended. See ":help recovery".

-R Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from accidentally overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The -R option also implies the -n option (see above). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'".

-s Silent mode. Only when started as "Ex" or when the "-e" option was given before the "-s" option.

-s {scriptin}

The script file {scriptin} is read. The characters in the file are interpreted as if you had typed them. The same can be done with the command ":source! {scriptin}". If the end of the file is reached before the editor exits, further characters are read from the keyboard.

-S {file} {file} will be sourced after the first file has been read. This is equivalent to -c "source {file}". {file} cannot start with '-'. If {file} is omitted "Session.vim" is used (only works when -S is the last argument).

-T {terminal}

Tells Vim the name of the terminal you are using. Only required when the automatic way doesn't work. Should be a terminal known to Vim (builtin) or defined in the termcap or terminfo file.

- u {vimrc} Use the commands in the file {vimrc} for initializations. All the other initializations are skipped. Use this to edit a special kind of files. It can also be used to skip all initializations by giving the name "NONE". See ":help initialization" within vim for more details.
- U {gvimrc} Use the commands in the file {gvimrc} for GUI initializations. All the other GUI initializations are skipped. It can also be used to skip all GUI initializations by giving the name "NONE". See ":help gui-init" within vim for more details.
- v Start Vim in Vi mode, just like the executable was called "vi". This only has effect when the executable is called "ex".
- V[N] Verbose. Give messages about which files are sourced and for reading and writing a viminfo file. The optional number N is the value for 'verbose'. Default is 10.
- V[N]{filename} Like -V and set 'verbosefile' to {filename}. The result is that messages are not displayed but written to the file {filename}. {filename} must not start with a digit.
- w{number} Set the 'window' option to {number}.
- w {scriptout} All the characters that you type are recorded in the file {scriptout}, until you exit Vim. This is useful if you want to create a script file to be used with "vim -s" or ":source!". If the {scriptout} file exists, characters are appended.
- W {scriptout} Like -w, but an existing file is overwritten.
- x If Vim has been compiled with encryption support, use encryption when writing files. Will prompt for a crypt key.
- X Don't connect to the X server. Shortens startup time in a terminal, but the window title and clipboard will not be used.
- Y Don't connect to the wayland compositor
- y Start Vim in easy mode, just like the executable was called "evim" or "eview". Makes Vim behave like a click-and-type editor.
- Z Restricted mode. Works like the executable starts with "r".

-- Denotes the end of the options. Arguments after this will be handled as a file name. This can be used to edit a filename that starts with a '-'.
 --clean Do not use any personal configuration (vimrc, plugins, etc.). Useful to see if a problem reproduces with a clean Vim setup.
 --cmd {command}
 Like using "-c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "-c" commands.
 --echo-wid GTK GUI only: Echo the Window ID on stdout.
 --gui-dialog-file {name}
 When using the GUI, instead of showing a dialog, write the title and message of the dialog to file {name}. The file is created or appended to. Only useful for testing, to avoid that the test gets stuck on a dialog that can't be seen.
 Without the GUI the argument is ignored.
 --help, -h, -?
 Give a bit of help about the command line arguments and options. After this Vim exits.
 --literal Take file name arguments literally, do not expand wildcards. This has no effect on Unix where the shell expands wildcards.
 --log {filename}
 If Vim has been compiled with eval and channel feature, start logging and write entries to {filename}. This works like calling `ch_logfile({filename}, 'ao')` very early during startup.
 --nofork Foreground. For the GUI version, Vim will not fork and detach from the shell it was started in.
 --nopugin Skip loading plugins. Implied by -u NONE.
 --not-a-term
 Tells Vim that the user knows that the input and/or output is not connected to a terminal. This will avoid the warning and the two second delay that would happen.
 --remote Connect to a Vim server and make it edit the files given in the rest of the arguments. If no server is found a warning is given and the files are edited in the current Vim.
 --remote-expr {expr}

Connect to a Vim server, evaluate {expr} in it and print the result on stdout.

--remote-send {keys}

Connect to a Vim server and send {keys} to it.

--remote-silent

As --remote, but without the warning when no server is found.

--remote-wait

As --remote, but Vim does not exit until the files have been edited.

--remote-wait-silent

As --remote-wait, but without the warning when no server is found.

--serverlist

List the names of all Vim servers that can be found.

--servername {name}

Use {name} as the server name. Used for the current Vim, unless used with a --remote argument, then it's the name of the server to connect to. If the socketserver backend is being used, if the name starts with "/", "./", or "../", it is taken as either an absolute, relative or relative path to the socket.

--clientserver {backend}

Use {backend} as the backend for clientserver functionality, either "socket" or "x11" respectively. Only available when compiled with both socketserver and X11 features present

--socketid {id}

GTK GUI only: Use the GtkPlug mechanism to run gVim in another window.

--startuptime {file}

During startup write timing messages to the file {fname}.

--ttyfail When stdin or stdout is not a terminal (tty) then exit right away.

--version Print version information and exit.

--windowid {id}

Win32 GUI only: Make gVim try to use the window {id} as a parent, so that it runs inside that window.

ON-LINE HELP

Type ":help" in Vim to get started. Type ":help subject" to get help on a specific subject.

For example: ":help ZZ" to get help for the "ZZ" command. Use <Tab> and CTRL-D to complete subjects (":help cmdline-completion"). Tags are present to jump from one place to another

(sort of hypertext links, see `":help"`). All documentation files can be viewed in this way, for example `":help syntax.txt"`.

FILES

`/usr/local/share/vim/vim??/doc/*.txt`

The Vim documentation files. Use `":help doc-file-list"` to get the complete list.

`vim??` is short version number, like `vim91` for Vim 9.1

`/usr/local/share/vim/vim??/doc/tags`

The tags file used for finding information in the documentation files.

`/usr/local/share/vim/vim??/syntax/syntax.vim`

System wide syntax initializations.

`/usr/local/share/vim/vim??/syntax/*.vim`

Syntax files for various languages.

`/usr/local/share/vim/vimrc`

System wide Vim initializations.

`~/.vimrc`, `~/.vim/vimrc`, `$XDG_CONFIG_HOME/vim/vimrc`

Your personal Vim initializations (first one found is used).

`/usr/local/share/vim/gvimrc`

System wide gvim initializations.

`~/.gvimrc`, `~/.vim/gvimrc`, `$XDG_CONFIG_HOME/vim/gvimrc`

Your personal gVim initializations (first one found is used).

`/usr/local/share/vim/vim??/optwin.vim`

Script used for the `":options"` command, a nice way to view and set options.

`/usr/local/share/vim/vim??/menu.vim`

System wide menu initializations for gVim.

`/usr/local/share/vim/vim??/bugreport.vim`

Script to generate a bug report. See `":help bugs"`.

`/usr/local/share/vim/vim??/filetype.vim`

Script to detect the type of a file by its name. See `":help 'filetype'"`.

`/usr/local/share/vim/vim??/scripts.vim`

Script to detect the type of a file by its contents. See `":help 'filetype'"`.

`/usr/local/share/vim/vim??/print/*.ps`

Files used for PostScript printing.

For recent info read the VIM home page:

<URL:<http://www.vim.org/>>

SEE ALSO

vimtutor(1)

AUTHOR

Most of Vim was made by Bram Moolenaar, with a lot of help from others. See ":help credits" in Vim.

Vim is based on Stevie, worked on by: Tim Thompson, Tony Andrews and G.R. (Fred) Walter.

Although hardly any of the original code remains.

BUGS

Probably. See ":help todo" for a list of known problems.

Note that a number of things that may be regarded as bugs by some, are in fact caused by a too-faithful reproduction of Vi's behaviour. And if you think other things are bugs "because Vi does it differently", you should take a closer look at the `vi_diff.txt` file (or type `:help vi_diff.txt` when in Vim). Also have a look at the 'compatible' and 'coptions' options.

2025 Jun 27

VIM(1)