vi, The Text Editor

This is a UNIX text editor where files are edited within the terminal window. The advantage of *vi* is that since you are working in a terminal window, your hands do not have to leave the keyboard and reach for the mouse, making *vi* ideal for quick edits. *vi* has its roots from the UNIX program ex, a command line editor.

vim, or *vi, improved*, is very similar to *vi*, but with increased functionality.

Startup

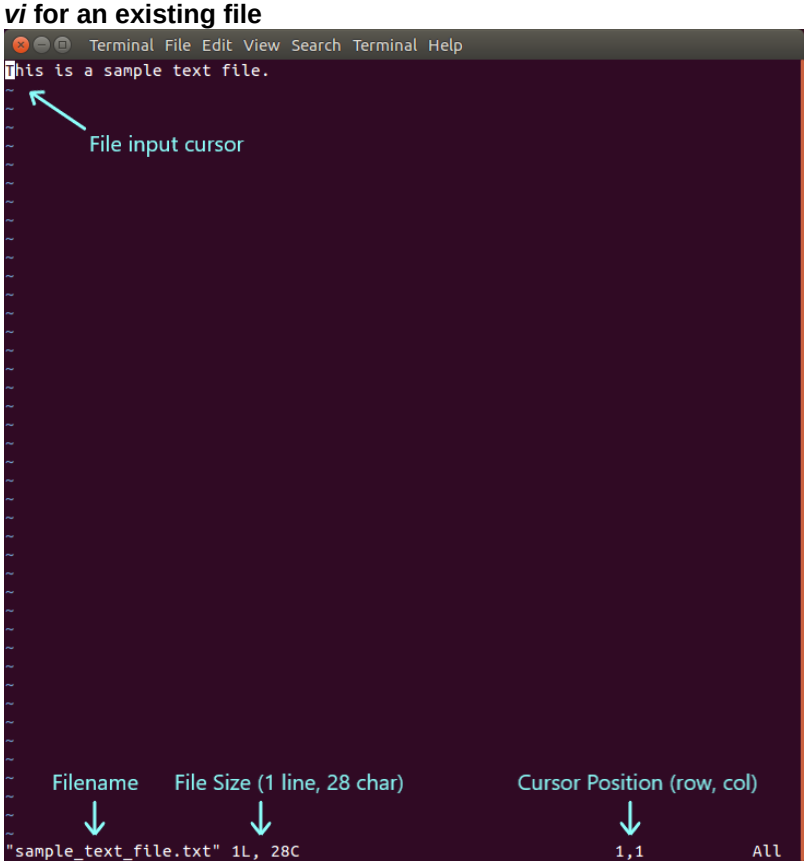
From the command line (Bash or Z shell terminal) prompt: `vi <filename>`

vi will start in *command mode*.

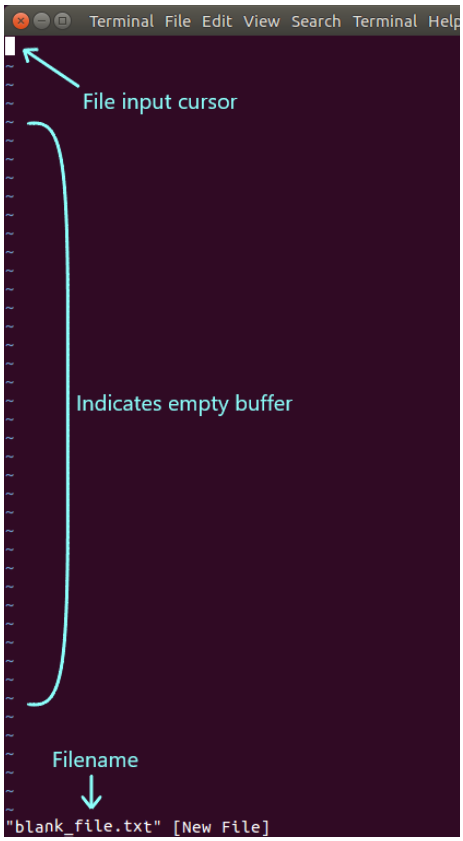
For an existing file, the contents will be displayed in your terminal, which is the *edit buffer*. Space after the end of the file to the bottom of the terminal window (with the exception of the last line) will have tildes (~) along the left side, indicating unused lines in the edit buffer.

If the file does not exist, this will create a blank file (the same as the `touch` command) and you will see a series of tildes (~) along the left side of your terminal, indicating the edit buffer is empty.

***vi* for an existing file**



***vi* for a new or blank file**

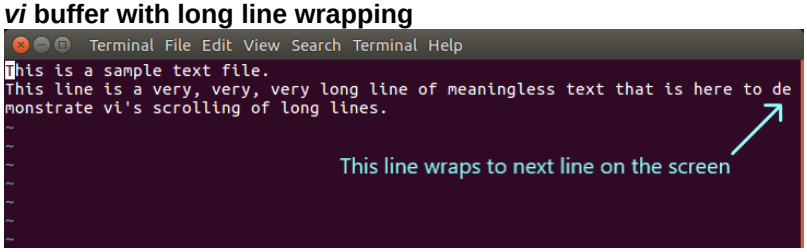


Basic Operation

The Edit Buffer

The workspace is called the *edit buffer*, which displays a copy of your file, not unlike other graphical user interface editors. But unlike most editors, there is no provision for seeing non-printing characters such as TAB or line feed. Lines that are longer than your terminal is wide will wrap, and files with more lines than the terminal will scroll off the bottom. Note that in this case, the last line of the terminal, similar to the examples above, will still be displayed.

***vi* buffer with long line wrapping**



It is important to note that you are working on a copy of your file and that your changes are not saved until you explicitly write them to the file.

Command Mode

This is the default mode at startup. While you see a cursor at the first position in the file, **anything typed while in command mode is interpreted as a command**, not text to be put into the file. In command mode, you can cut, copy and paste text, as well as move the cursor to the position in the file where you want to start editing.

Searching, including pattern matching, writing changes, i.e., saving the file, and exiting are done in command mode.

Although the cursor is at some location in the edit buffer, some commands are not displayed on the screen while others are displayed at the bottom of the terminal window.

Press `i` to *insert* text before the cursor, press a to *append* text after the cursor, press O to *open a new line above* the cursor and press o to *open a new line below* the cursor; this puts you into *edit mode*.

> ⚠ **Commands are case sensitive.**
>
> For example, lower-case u undoes your last change while upper-case U undoes the changes made to the current line (where the cursor is).

Pressing the ESC key while in edit mode will return you to command mode. The ESC key will also cancel commands that are not yet complete.

Edit Mode

Sometimes referred to as *insert mode*, this is the mode where you can make changes to, or edit, the text, and is reached by either the `i` (insert before), a (append after), O (open and input new line above), or o (open and input new line below) keys after placing the cursor at the position where you want to start editing. When in edit mode, keystrokes are interpreted as text to be entered into the file.

When in edit mode, you type text as if you would in any other editor, except you must press the `Enter` (or `Return`) key at the end of each line. *vi* is not a word processor that will automatically format for you.

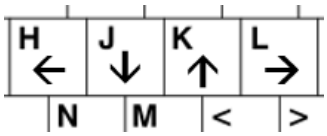
Return to command mode by pressing the ESC key.

Basic Editing

Moving the cursor and deleting text are done in command mode. Typing text is done in edit mode.

Moving the Cursor

In command mode, move the cursor to the desired position with the arrow keys or the `h` (), `j` (), `k` () or `l` () [all lower-case] keys.



Note that moving left from the beginning of a line or moving right from the end of a line will not move to the preceding or following line, respectively.

Keys	Move
<SPACE>	One space to the right (same as `l`)
<BACKSPACE>	One space to the left (same as `h`)
`w`	One word forward
`b`	One word back
`e`	End of the current word
`0`	Beginning of the current line
`$`	End of the current line
`H`	Top line on the screen
`M`	Middle line on the screen
`L`	Last line on the screen
`G`	End of the file
<CTRL>f	Forward one screen
<CTRL-b	Back one screen
<CTRL>e	Scroll down a single line
<CTRL>d	Scroll down half a screen
<CTRL>u	Scroll up half a screen

Deleting Text

Keys	Delete Operation
`x`	Delete character under the cursor (or to the right of a line cursor between characters)
`#x`	Delete # characters to the right of the cursor (current character included) Example: `3x` will delete the three characters under and to the right of the cursor

`dl` `#dl`	Delete the character under the cursor, same as `x` Delete # characters, same as `#x`
`dw` `#dw`	Delete the current word Delete # words (current word included)
`dd` `#dd`	Delete the current line Delete # lines (current line included)
`D`	Delete the rest of the current line

Copying and Pasting

Copying and pasting is done while in command mode.

Copying text is referred to as *yanking*, cutting text is referred to as *deleting*, and pasting text is referred to as *putting*. Text yanked or deleted are put into *buffers*. Unlike most system clipboards, *vi* has named and unnamed buffers allowing you to save up to 27 yanked or deleted series of text to be put multiple times. One buffer is unnamed (think of this as a traditional clipboard), and up to 26 can be named using the letters *a* through *z*. The unnamed buffer is the one accessed by default.

Using upper-case *A* through *Z* instead of lower-case in any yank or delete command will append the text to the end of that buffer's content.

The last deletion is stored in the unnamed buffer while the last nine deletions are stored in numbered buffers, numbered 1 (for the last deletion, the same one stored in the unnamed buffer until overwritten) through 9 (the ninth last deletion) [think of a *queue* with a length of 9, first in, first out].

While the last deletion can be overwritten in the unnamed buffer with a yank command, this deletion remains in the first numbered buffer until a subsequent deletion, where the new deletion takes over the first numbered buffer, the next eight are bumped to the following numbered buffers, and the ninth is bumped off the buffers.

To access a named or numbered buffer, precede the yank, delete or put command with double quotes (`"`) then the letter or number of the named buffer.

Keys	Buffer Operation
`v`	Select text mode. Use move commands to continue to select text from the point where you pressed `v` Then use yank or delete to copy or cut into the buffer
`y`	Yank the selected text into the unnamed buffer
`y+`	Yank to the end of the current line
`p`	Put the contents of the unnamed buffer after the cursor
`P`	Put the contents of the unnamed buffer before the cursor
`yy` or `Y` `#yy` or `#Y`	Yank the current line into the unnamed buffer Yank # lines (current line included) into the unnamed buffer
`"ay`	Yank the selected text into buffer *a* [valid values are *a* through *z*]
`"byy` or `"bY`	Yank the current line into buffer *b* [valid values are *a* through *z*]
`"ap`	Put the contents of buffer *a* after the cursor

`"bP`	Put the contents of buffer *b* before the cursor
`"#p`	Put the contents of the *#th* deletion before the cursor
`"Ay`	Append the selected text to the end of the content of buffer *a*

Saving and Exiting

You must be in command mode to save and / or exit *vi*. Press the ESC key to enter command mode, if in edit mode. All save and exit commands are preceded by the colon (:); after pressing the colon key, the colon and any subsequent commands will be displayed at the bottom of the screen.

Keys	Command
`:w`	Write (or save) the file, but do not exit *vi*
`:wq`	Write the file, then exit *vi*
`:q`	Quit (or exit) *vi*. This command will fail if changes are made to the file
`:q!`	Quit *vi*, discarding any changes, i.e., without saving changes