6.2. Using the Vim editor

6.2.1. Two modes

The **vi** editor is a very powerful tool and has a very extensive built-in manual, which you can activate using the **:help** command when the program is started (instead of using **man** or **info**, which don't contain nearly as much information). We will only discuss the very basics here to get you started.

What makes **vi** confusing to the beginner is that it can operate in two modes: command mode and insert mode. The editor always starts in command mode. Commands move you through the text, search, replace, mark blocks and perform other editing tasks, and some of them switch the editor to insert mode.

This means that each key has not one, but likely two meanings: it can either represent a command for the editor when in command mode, or a character that you want in a text when in insert mode.

Pronunciation

It's pronounced "vee-eye".

6.2.2. Basic commands

6.2.2.1. Moving through the text

Moving through the text is usually possible with the arrow keys. If not, try:

- **h** to move the cursor to the left
- **l** to move it to the right
- **k** to move up
- **j** to move down

SHIFT-G will put the prompt at the end of the document.

6.2.2.2. Basic operations

These are some popular **vi** commands:

- **n dd** will delete n lines starting from the current cursor position.
- **n dw** will delete n words at the right side of the cursor.

- **x** will delete the character on which the cursor is positioned
- **:n** moves to line n of the file.
- **:w** will save (write) the file
- **:q** will exit the editor.
- **:q!** forces the exit when you want to quit a file containing unsaved changes.
- **:wq** will save and exit
- **:w newfile** will save the text to `newfile`.
- **:wq!** overrides read-only permission (if you have the permission to override permissions, for instance when you are using the *root* account.
- **/astring** will search the string in the file and position the cursor on the first match below its position.
- / will perform the same search again, moving the cursor to the next match.
- **:1, $s/word/anotherword/g** will replace `word` with `anotherword` throughout the file.
- **yy** will copy a block of text.
- **n p** will paste it n times.
- **:recover** will recover a file after an unexpected interruption.

6.2.2.3. Commands that switch the editor to insert mode

- **a** will append: it moves the cursor one position to the right before switching to insert mode
- **i** will insert
- **o** will insert a blank line under the current cursor position and move the cursor to that line.

Pressing the **Esc** key switches back to command mode. If you're not sure what mode you're in because you use a really old version of **vi** that doesn't display an "INSERT" message, type **Esc** and you'll be sure to return to command mode. It is possible that the system gives a little alert when you are already in command mode when hitting **Esc**, by beeping or giving a visual bell (a flash on the screen). This is normal behavior.

6.2.3. The easy way

Instead of reading the text, which is quite boring, you can use the vimtutor to learn you first Vim commands. This is a thirty minute tutorial that teaches the most basic Vim functionality in eight easy exercises. While you can't learn everything about **vim** in just half an hour, the tutor is designed to describe enough of the commands that you will be able to easily use Vim as an all-purpose editor.

In UNIX and MS Windows, if Vim has been properly installed, you can start this program from the shell or command line, entering the **vimtutor** command. This will make a copy of the tutor file, so that you can edit it without the risk of damaging the original. There are a few translated versions of the tutor. To find out if yours is available, use the two-letter language code. For French this would be **vimtutor** ***fr*** (if installed on the system).

Prev Home Next
Text editors Up Linux in the office