Vim

## Introduction

Vim was published in november 2nd, 1991 as implementation of Vi (1976)

## How ViM is Ubiquitous

For starters, Vim is installed practically everywhere. I can’t remember ever logging into a system and not being able to use the Vim editor. Most Linux distributions install it by default. With other editors such as nano or Emacs, they may or may not be available. If you’re working on a system and you’re only editor choice is Vim, then you need to at least be able to use it to make simple edits.

Also, many programs rely on an external editor. By default that external editor is Vim in most cases. Typically this can be overridden by setting an environment variable such as EDITOR. However, if you execute one of these commands and find yourself looking at the Vim editor, you will be happy you at least know how to make simple edits and exit out of Vim. Just a couple of commands that rely on an external editor such as Vim include crontab, visudo, and git. **RISCRIVERE STESSO CONTENUTO CON PAROLE DIVERSE**

## How learning VIM improves your productivity

Typing speed is irrelevant up to a certain level. Your ability to navigate through code is much more important. This is where Vim along with its keybindings, layout and set up can help you speed up the process.

The skills that you build while learning Vim can be used in many different editors, such as VSCode (with the Vim plugin) or Sublime Text or so…

## Vim modes

### Normal Mode

By default, Vim starts in “normal” mode. Normal mode can be accessed from other modes by pressing Esc.

Normal mode is where one should spend most of their time while using Vim because in this mode you can navigate in the text and change blocks of code with few commands.

In normal mode, there are multiple ways to move around an open file. In addition to using the cursor keys to move around, you can use h (left), j (down), k (up), and l (right) to move as well.

You can also make changes to single characters in normal mode. You can delete single characters by moving your cursor over it and pressing x.

To perform an undo, press u in normal mode. This undoes changes up to the last time you were in normal mode. If you want to redo (i.e., undo your undo) press Ctrl+r in normal mode.

### Insert Mode

This is the second most used mode, and will be the most familiar behavior to most people. Once in insert mode, typing inserts characters just like a regular text editor. You can enter it by using an [insert command](#_k88aw1iew21k) from normal mode.

To leave insert mode and return to normal mode, press Esc.

### Visual Mode

Visual mode is used to make selections of text, similar to how clicking and dragging with a mouse behaves. Selecting text allows commands to apply only to the selection, such as copying, deleting, replacing, and so on.

To make a text selection:

Press v to enter visual mode, this will also mark a starting selection point

Move the cursor to the desired end selection point; vim will provide a visual highlight of the text selection

Visual mode also has the following variants:

V to enter visual line mode, this will make text selections by line

<C-V> to enter visual block mode, this will make text selections by blocks; moving the cursor will make rectangle selections of the text

To leave visual mode and return to normal mode, press Esc or <C-[>.

The visual mode actually has multiple subtypes: visual, block-visual and linewise-visual

visual: like described above. Enter by pressing v

block-visual: select any rectangular region. Enter by pressing <ctrl>+v

linewise-visual: always select full lines. Enter by pressing <shift>+v

### Command Mode

Command mode has a wide variety of commands and can do things that normal mode can’t do as easily. To enter command mode type ’:’ from normal mode and then type your command which should appear at the bottom of the window. For example, to do a global find and replace type :%s/foo/bar/g to replace all ‘foo’ with ‘bar’

: Enters command mode

% Means across all lines

s Means substitute

/foo is regex to find things to replace

/bar/ is regex to replace things with

/g means global, otherwise it would only execute once per line

Vim has a number of other methods that you can read about in the help documentation, :h or :help.

### Replace Mode

Replace mode allows you to replace existing text by directly typing over it. Before entering this mode, get into normal mode and put your cursor on top of the first character that you want to replace. Then press ‘R’ (capital R) to enter replace mode. Now whatever you type will replace the existing text. The cursor automatically moves to the next character just like in insert mode. The only difference is that every character you type will replace the existing one.

Movement Commands

j (move down);

k (move up);

l (move right);

h (move left);

*<N>*j;

You can also use the above command with ‘k’, ‘l’, ‘h’.

You can also move with the arrow keys.

gg (move to the first line of the file);

G (move to the last line of the file);

:*<N>*;

H (top line on screen);

M (middle line on screen);

L (bottom line on screen);

Utility Commands

Commands for opening, closing and saving files

Changing mode Commands

Commands for copy and pasting

How to create and use macros

## Insert Commands