Vim & NeoVim

FCIS OSC

How to exit Vim

How to exit Vim

`[:q! <ENTER>]` or `[ZQ]` for discarding the changes.

`[:wq <ENTER>]` or `[ZZ]` for saving the changes.

Why use Vim

Why use Vim

- Very fast editor
- You have low system resources.
- You want a fast editing way (Very Fast).
- Extensible text editor.

Personally, I use it because I accidentally got used to its movement mechanism.

VIM Modes

VIM Modes

There multiple modes in vim for dealing withe text.

When enter any mode an indication will appear at the bottom left of the screen.

VIM Modes

Normal Mode

Normal Mode

This mode is entered by `[ESC]` key, and used in navigation and viewing the text.

In **Vim** and **NeoVim**, the cursor is denoted by a block shape by default.

VIM Modes

Insert Mode

Insert Mode

This mode is enterd by `[I]` key, and used for editing the text.

In **NeoVim**, the cursor is transformed to a vertical line (I-Beam).

VIM Modes

Command Mode

Command Mode

This mode is entered by `[:]` key, and used for passing vim commands to vim.

If you remember, we used this mode when quiting vim. the command `[w]` is a shortcut to *‘write’* and `[q]` is for *‘quit’*.

In **Vim** and **NeoVim**, a prompt with a colon `[:]` will appear at the bottom left indicating the command mode.

VIM Modes

Visual Mode

Visual Mode

This mode is entered by `[v]`, and used for selection the text, but you should be in **Normal** mode to enter it.

Navigation

Navigation

You can use the arrow keys, but if you will, you are not permitted to proceed with us -_-.

All the navigation keys should be performed in normal mode.

Navigation

Main Navigation

Main Navigation

The proper way for navigation is using `[h]`, `[j]`, `[k]`, `[l]`

- `[h]` –> Move left.
- `[j]` –> Move down.
- `[k]` –> Move up.
- `[l]` –> Move right.

Navigation

Motion Navigation

Motion Navigation

Keys

- [w] –> Move to the beginning of next word.
- [b] –> Move to the beginning of the previous word.
- [e] –> Move to the end of the next word.
- [gg] –> Move to the beginning of current buffer.
- [G] –> Move to the end of current buffer.

Text Manipulation

Text Manipulation

Simple Text Manipulation

Simple Text Manipulation

- `[x]` –> Remove a character.
- `[r <char>]` –> Change a character.

Text Manipulation

Motion Text Manipulation

Motion Text Manipulation

`<Manipulation Key> <Motion Key>`

Examples

- `[dw]` –> Delete to the beginning of the next word.
- `[db]` –> Delete to the beginning of the previous word.
- `[d$]` –> Delete to the end of the line (**$** is regex for end of the line).
- `[d{0 or ^}]` –> Delete to the beginning of the line

`[c]` is the same as `[d]`, but put vim in insert mode (for changing text).

The motion keys also word in visual mode `EX: [vw]` –> select to the beginning of the next word.

Text Manipulation

Text Object Manipulation

Text Object Manipulation

If the cursor is positioned at the middle of a word and you want to delete it, not just half of it to the beginning of the next word.

In the previous situation, **Text Objects** is the way to go.

Text object may be a *word*, a *sentince*, a *paragraph*, etc . . . .

Text Manipulation

Text Object Manipulation (Examples)

Text Object Manipulation (Examples)

To preform a **text object** manipulations [i] or [a] should be prefixed before the **text object** key.

Examples

- [diw] –> Delete a word **without** the around spaces.
- [daw] –> Delete a word **with** the around spaces.
- [dis] –> Delete a sentince without the around spaces.
- [dip] –> Delete a paragraph without the around spaces.

Same for [c] (change) and [v] (select/visualize).

Text Manipulation

Text Object Manipulation (More Examples)

Text Object Manipulation (More Examples)

!!! This is a good one !!!

Examples

- [ci"] –> Change what is between the double qoutes **leaving** the qoutes.
- [ca"] –> Change what is between the double qoutes **and** the qoutes.

Same for **single qoutes**, **braces**, **brackets**, and **parentheses**.

Text Manipulation

Undo & Redo

Undo & Redo

Just forget , it wont work and will put vim procces in backgroud , if you are using the terminal version of Vim.

If that happened, run the shell command `[$ fg]`.

keys

- `[u]` –> Undo the previous changes.
- `[R]` –> Redo the previous changes.

Copying (Yanking) & Pasting

Copying (Yanking) & Pasting

The verb **'yank'** is text in Vim as the coppied text goes to a place called vim **register** for later use.

- `[y]` –> Yanks the selected text.
- `[Y]` –> Yank the entire line.
- `[p]` –> For pasting the yanked text.

Searching

Searching

In order to find text inside the text, we use the forward slash `[/]`, which accepts regular expression.

Example

- `[/^[1-9]]` –> Will go to the line which starts with a number.
- `[/vim$]` –> Will go to the line which ends with **vim** key.

Press `[n]` key to go to the next occurrence of the regular expression.

Replacing

Replacing

Replacing is done with a substitution command `[:s]`, remember that `[:]` enters the command mode.

Usage: `[:s/regex/text/]`

Example

- `[:s/printf(/fprint(stderr, /g]` –> change printf function to fprintf.

[g] is used after the command to change all the occurrence in the line.

Vim & NeoVim Configuration

Vim & NeoVim Configuration

Vim and NeoVim look for a multiple config files to source.

- `"$HOME/.vimrc"` –> For Vim and NeoVim.
- `"$HOME/.config/nvim/init.vim"` –> For NeoVim.

These config files writting by a language called **Vim Script**.

Vim & NeoVim Configuration

Commonly used settings for Vim

Commonly used settings for Vim

Settings

```
set number " enable line numbering
set relativenumber " Enable relative line numbering
set autoindent " proceed with the current indentation when
set smartindent " sea [:h smartindent]
set scrolloff=8 " start scrolling if the cursor is 8 lines
set mouse=a " the ability to use the mouse in NeoVim
set cursorline " highlight the current line
set tabstop=8 " set the tab width
set shiftwidth=8 " set the tab width when shifting by < or
set clipboard+=unnamedplus " enable NeoVim to access the sy
```

Vim & NeoVim Configuration

Mapping

Mapping

When dealing with a common task/commands, maping a keyboard shortcut comes in handy.

Say you need to execute the current python file, or building the current project.

This is how it works.

map {key} {command}

Example

```
au Filetype python map <C-c> :w <CR> :! python3 % <CR>
au Filetype c,cpp map <C-c> :w <CR> :! make <CR>
```

Perfixing with [`au Filetype <lang>`], tell vim to map only for this file type

[`<CR>`] is as if you pressed [`<RETURN>`]

Vim & NeoVim Configuration

Plugins

Plugins

In order make it easy we should consider a **Plugin Manager**.

A good example is Plug, go to the github page and follow the instuction.

After installing it put these lines in your vim config.

Plug Config

```
call plug#begin('~/.vim/plugged')


" All the Plugin goes here.
" ex:
" Plugin 'Plugin-Maintainer/Plugin Name'
Plugin 'gruvbox-community/gruvbox' " Gruvbox Color Scheme
call plug#end()
```

after putting your plugins like above, do [`:PlugInstall`]. This will make **Plug** plugin manager to clone and install your plugins.

Vim & NeoVim Configuration

Color Scheme

Color Scheme

If you installed **Gruvbox** or any other color scheme, you can set it by

```
colorscheme gruvbox
```

some of the plugins you install will depened on a global variable to set it. In the case of **Gruvbox**.

```
let g:gruvbox_italic=1
let g:gruvbox_contrast_dark = 'hard'
let g:gruvbox_transparent_bg=1
```

these variable will make gruvbox transparent dark background with italic font.

Integration With other text editors and IDEs

Integration With other text editors and IDEs

May text editor and IDEs have a plugin for vim, which you can't use it to get use to vim movement mechanism.

- VS Code
- Atom
- JetBrains Products

The End

The End

- Author: Mina Saadallah
- Github: “https://github.com/MinaSaad47”