

Vim Workshop

By 汪睿@Hydraallen, 周赵嘉程@zzjc1234, 胡泓峥

2023/4/2 19:00

Part 0 -- Intro

Our aim

Learn how to use vim in your future project, and THOWR YOUR MOUSE.

What is vim

- An editor
 - vi -> visual interface
 - vim -> vi Improved
-

As you can see from the advertisement of this workshop, vim has many short-cuts and other fabulous functions that can aid our programming.

Why vim

- No switch between keyboard and mouse
- Easy to move
- No duplicate operation

How to learn vim

Practice Practice Practice.

From a small program or from an English essay

Part 1 -- Basic Operation

Menu

- [Vim Workshop](#)
 - [Part 0 -- Intro](#)
 - [Our aim](#)
 - [What is vim](#)
 - [Why vim](#)
 - [How to learn vim](#)

- [Part 1 -- Basic Operation](#)
 - [Menu](#)
 - [Before Start](#)
 - [Mode Intro](#)
 - [Normal Mode](#)
 - [Why key in vim is so uncomfortable?](#)
 - [Insert Mode](#)
 - [Life Hack](#)
 - [Visual Mode](#)
 - [Replace Mode*](#)
 - [Command Mode](#)
- [Part 2 -- vimrc](#)
 - [Without Plugin](#)
 - [Shortcut key&key mapping](#)
 - [With Plugin](#)
 - [Vim Macro](#)
 - [Suggestions](#)
 - [Other vim](#)
 - [Throw Away Your Mouse!](#)

Before Start

1. Use `:help`

You can use the command in the vim just like what you've done in the matlab

2. Vimtutor

You can type vimtuor in your terminal to start the vimtutor, a Chinese tutor for vim. It only takes you about 25~30 min. If you feel just give it a try if you want to go over the basic command of vim.

3. Official Website

Community is an important part of editor. There are a lot of good quality resource made by the developers and experienced vimers on the official website. You can join the community or search on the official website to get supported.

4. Vim Adventure

Vim adventure is a great game intended to help beginners to learn the vim with fun. Since it is non-free, you can pay for the rest of the game or find a reverse engineered version.

5. [Openvim](#)

6. [Cheatsheet](#)
-

Mode Intro

1. Normal Mode
2. Insert Mode
3. Visual Mode
4. Replace Mode*
5. Command Mode

Demo using empty

Normal Mode

1. Navigating

1. Basic: hjkl

You can use `h j k l` to replace the direction key on your key board.

`h` for `$\leftarrow`, `j` for `$\downarrow`, `k` for `$\uparrow`, `l` for `$\rightarrow`

2. Word: bwe

You can use the wbe to jump word by word.

Difference between `word` and `WORD`: The `word` is the combination of (a-zA-Z0-9), or the combination of punctuation and character like `@##$%` while the `WORD` is any non-empty string between the white-space.

NOTE THAT `as213df%##^>?^&` is two `word`, `as213df` and `%##^>?^&`.

- `w`: go to the beginning of next word
- `b`: go to the beginning of previous word
- `e`: go to the end of the word
- `W`: go to the beginning of next WORD
- `B`: go to the beginning of previous WORD
- `E`: end of the WORD

3. Line: 0, ^, \$

- `0`: Go to the beginning of line
- `^`: Go to the beginning of line (after whitespace)
- `$`: Go to the end of line

4. Char:

- `f{char}`: Go forward to character
- `F{char}`: Go backward to character

5. Document

- `gg`: First line
- `G`: Last line

- `{number}` : Go to line {number}
- `{number}G` : Go to line {number}
- `{number}j` : Go down {number} lines
- `{number}k` : Go up {number} lines

6. Window

- `zz` : Center this line
- `zt` : Top this line
- `zb` : Bottom this line
- `H` : Move to top of screen
- `M` : Move to middle of screen
- `L` : Move to bottom of screen

2. Clipboard:

1. `x` : Delete the character
2. `dd` : Cut the line
3. `yy` : Yank line
4. `p` : Paste
5. `P` : Paste before
6. `"*p / "+p:` Paste from system clipboard
7. `"*y / "+y:` Paste to system clipboard

Note: For users using wsl, you may encounter some problems when using the clipboard, this issue may be solved by part2

3. Editing

1. `u` : You can use `u` to undo the editing
2. `<C-R>` : You can use `<C-R>` to redo the editing

4. Opeartors

Operators let you operate in a range of text (defined by motion). These are performed in normal mode.

For Example:

Operator	Motion
d	w

- Text Objects

key	Object
p	Paragraph
w	Word
s	Sentence
[({ < A [], (), or {}	block
' " `	A quoted string
b	A block (
B	A block in {
t	A XML tag block

- Sentence and Paragraph in vim
- Command

d1	delete character (alias: "x")
dw	delete next word
dW	delete next WORD
diw	delete inner word
daw	delete a word
diW	delete inner WORD
daW	delete a WORD
db	delete to beginning of the word
dgn	delete the next search pattern match
dd	delete one line(actually cut one line)
dis	delete inner sentence
das	delete a sentence
dib	delete inner '(' ')' block
dab	delete a '(' ')' block
dip	delete inner paragraph
dap	delete a paragraph
diB	delete inner '{' }' block
daB	delete a '{' }' block

For more combinations, use `:help motion.txt`

You can repeat the command by assigning a number.

For example: 3dd means delete three lines. Times of repetition should be the multiplication of the numbers. Eg. 2y3y means yank 6 lines.

Demo with article.txt

Why key in vim is so uncomfortable?



Insert Mode

1. Open Insert Mode:

1. **a**: append
2. **i**: insert
3. **o**: insert in next line
4. **A**: append at the end of the line
5. **I**: insert text before the first non-blank in the line
6. **O**: insert in previous line
7. **s**: delete char and insert
8. **S**: delete line and insert
9. **c**: **c** is equal to "change". **c** is motion like **d** and **y**.
10. **C**: delete until the end of the line and insert

2. Exit Insert Mode

1. Press **<Esc>**: exit insert mode
2. **ctrl c**: exit insert mode and abort current command

Life Hack

- Use `.` to repeat previous edit
- Use `:!` to run command in terminal
- Use `:r` to insert the file below the cursor

Visual Mode

1. Visual Mode

Start Visual mode per character.

2. Visual Line Mode

Start Visual mode linewise.

3. Visual Block Mode

Start Visual mode blockwise.

4. Use `gu` and `gU`.

5. Use `>` and `<`

6. d, y, c, o in the visual mode

Exercise

Change `wallpaper.txt`

```
wallpaper_1.jpg  
wallpaper_1.jpg  
.....  
wallpaper_1.jpg
```

Into

```
Wallpaper_1.jpg  
Wallpaper_1.jpg  
.....  
Wallpaper_1.jpg
```

Replace Mode*

You can use `r` to replace a character.

You can use `R` to enter the replace mode.

Command Mode

1. Exiting Vim:

1. `:q`: quit the file
2. `:qa`: quit all the file
3. `:w`: Write(Save the file)
4. `:wq`: Write and quit the file
5. `:x`: Write and quit the file(only write when there is change).
6. `zz`: Save and quit (the same to :x)
7. `zQ`: Quit without checking changes
8. `!`: You can add a `!` at the end of the command to do the command by force. For example to quit the file by force, you can use `:q!`

2. Call the terminal

You can call the terminal by using the command `:term`. You can also use term to run command like compiling a c-file. For example:

```
:term gcc vim.c
:term ./a.out
```

You can also open a terminal window in the vim using the command; `:term`

You can use command to determine the position of the terminal. For example:

```
:belowright term ++rows=10
```

If you want to enter the terminal for a while and then go back to vim, you can use the following code

```
<C-z> "suspend to vim and go to the terminal
```

```
fg ##go back to vim
```

You can also determine which terminal you use with the command

```
:term {terminal you want to use}

" for example
:term python3
```

3. Run Vim command

Believe it or not, vim has its own language. You can use the command of vim to personalize your vim. This part will be included in the `Modify .vimrc(noplugin)`

- Search

You can use search the string by entering `/` and `?`

Difference between `/` and `?`:

`/` means search from top to bottom,

`?` means search from bottom to top.

If you want to search for exactly the `word (WORD?)`, you can search in this way: `^\<{word you want to search}\>` or `?^\<{word you want to search}\>`

For example, you can use `^\<hello\>` to search the word `hello` from top to bottom

- `n`: Search for the next pattern
- `N`: Previous match
- `*`: Next whole word under cursor
- `#`: Previous whole word under cursor

Vim will record your search history. You use command like `/$\uparrow` or `/$\downarrow` if you want to search the word you searched before.

You can also replace all the results fit the expression you search by command:

```
{range}s/{search}/{replace}/{flags} {count}
```

You can use `^\<{word}\>` to search for the "word"

- range is the searching range, You can define the range in following way:

```
{start},{end}
```

There are some characters have special meaning when defining the range:

`%` stands for all the lines in the file

`.` stands for the current line

`$` stand for the last line in the file

If range is not set, substitute will only happens on the current line.

- `s` stands for substitute
- Commonly used flags
 - `g`: Replace all occurrences in the line. Without this argument, replacement occurs only for the first occurrence in each line.
 - `i`: Ignore case for the pattern.
 - `I`: Don't ignore case for the pattern.
 - `n`: Report the number of matches, do not actually substitute.

Example:

```
"replace all int with double
```

```

:%s/int/double/g

"replace first int in each line with double
:%s/int/double/

"replace all int in line1-line3 with double
:1,3s/int/double/g

"replace all int from current line to the next 4 line with double
:.,.+4s/int/double/g

"do :%s/int/double/g for 4 times(from current line to current+3 line)
:%s/int/double/g4

```

For more information, you can type `:help substitute` in vim.

- Regex in Search

The behavior of regex in vim can be determined by setting the magic.

Mode	Behavior
magic \m	^, \$, ., *,[] has special meaning
nomagic \M	^,\$,\ has special meaning
verymagic \v	all character has special meaning
verynomagic \V	no character has special meaning except \

Example:

```
:%s/\vmagics?/abstract/g
```

Quiz using fix_article

Quiz using C_Code.c

Final Quiz

Part 2 -- vimrc

Without Plugin

See -> `Vim-Wksp/vimrc-noplugin/vimrc`

We strongly suggest you do this:

```
cd ~/
git clone git@github.com:zzjcl234/Vim-Wksp.git
mkdir .vim
cd .vim
cp ~/Vim-Wksp/vimrc-noplugin/vimrc ~/.vim/vimrc
vim
```

Now your vim should be look like this

[illegible]

Next

And `vim vimrc`

Read it line by line! (cover `split` btw)

Shortcut key&key mapping

1. `map` Equivalent key

`noremap` Change key

change `a` into `b`

```
noremap a b
```

Move faster!

```
noremap H 5h
noremap J 5j
noremap K 5k
noremap L 5l
```

Quick save and source

```
map s <nop>
"s为删掉当前字符并进入写入模式
map S :w<CR>
"<CR>表示回车，效果：s即保存写入
map Q :wq<CR>
"效果：Q即退出vim
map R :source R $MYVIMRC<CR>
"效果：R刷新vimrc
map ; :
```

2. `set` -> see detail in `vimrc-noplugin/vimrc`

3. split screen

`:spilt` split up and down

In `vimrc-noplugin`, `su` for split up, `sd` for split down

`:vspilt` split left and right

In `vimrc-noplugin`, `sl` for split left `sr` for split right

Move the cursor position manually `ctrl+w+h/j/k/l`

4. Change backgroud (Do you want to be manuel)

`:color` +tab

Go back to default `:color default`

With Plugin

See -> `Vim-Wksp/vimrc-withplugin/vimrc`

We strongly suggest you do this in wksp:

1. Install [Vundle](#)

```
cd ~/.vim
mkdir bundle
cd bundle
git clone https://github.com/VundleVim/Vundle.vim.git
```

Windows users may need to refer to this -> <https://github.com/VundleVim/Vundle.vim/wiki/Vundle-for-Windows>

2. Write `vimrc`

You don't need to write now because I have prepared it for you. All you need to do now is to understand!

```
set nocompatible          " be iMproved, required
filetype off

" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle##begin('~/.vim/bundle')
" alternatively, pass a path where Vundle should install plugins
"call vundle##begin('~/.vim/bundle')

" let Vundle manage Vundle, required

call vundle##end() " required

filetype on " required
filetype indent on " required
filetype plugin on " required
filetype plugin indent on " required
" To ignore plugin indent changes, instead use:
"filetype plugin on
"
" Brief help
" :PluginList       - lists configured plugins
" :PluginInstall    - installs plugins; append `!` to update or just :PluginUpdate
" :PluginSearch foo - searches for foo; append `!` to refresh local cache
" :PluginClean      - confirms removal of unused plugins; append `!` to auto-approve
removal
"
" see :h vundle for more details or wiki for FAQ
" Put your non-Plugin stuff after this line
```

3. Find and install plugin

- Search in Github
-



- write in vimrc `Plugin 'vim-airline/vim-airline'`
- plug install `:PluginInstall` 或 `:BundleInstall`
plug update `:PluginUpdate`
plug uninstall `:PluginClean`

4. Time to have a try!

We strongly suggest you do this in wksp:

```
cd ~/
git clone git@github.com:zzjc1234/Vim-Wksp.git
mkdir .vim
cd .vim
cp ~/Vim-Wksp/vimrc-withplugin/vimrc ~/.vim/vimrc
vim vimrc
```

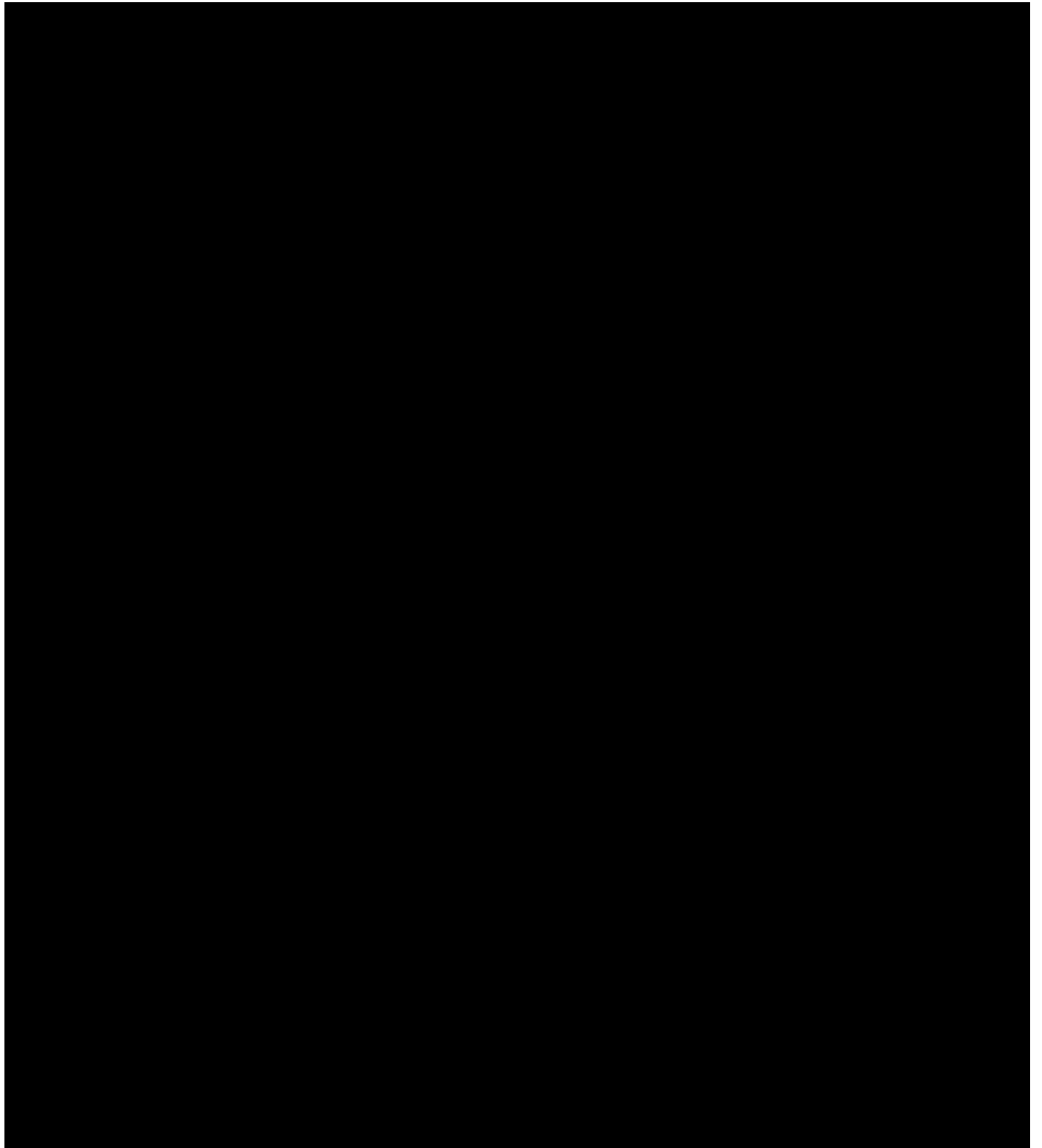
There will be some warnings or errors. But **DO NOT PANIC**, just press `enter`



Now install some plugin `:PluginInstall`

(In vimrc-withplugin, we included `airline, snazzy, ale, nerdtree`)

And now your vim should be look like this:



If you want use more plugin and don't want to spend too much time, you may refer to [Vim-Wksp/vimrc-sample/vimrc](#)

Vim Macro

You can use vim macro to complete the repetition work.

1. Enter ``q\<char\>`` to start recording the macro.
2. Edit one line
3. Enter ``q`` to stop recording
4. Move to the line you modify(You can include this step in recording the macro)
5. Enter ``@\<char\>``

Now exercise!

Write in `empty.txt`

```
wallpaper_1.jpg  
wallpaper_2.jpg  
.....  
wallpaper_100.jpg
```

Hint: `Ctrl+a` integer+1; `Ctrl+x` integer-1

Suggestions

1. If you want use vim more smoothly, you may need to change `esc` and `Capslock`

In macOS, you may use karabiner

In WIndows, you may use Powertoys

In Linux, I believe you can find one by yourself!

2. plugin

Here are some recommendations:

插件list (推荐)

- [airline](#) 状态栏
- [snazzy](#) 配色

```
colorscheme snazzy
```

- [nerdtree](#) 文件树
- [tagbar](#) 显示函数列表
- [ale](#) 错误检查
- [YouCompleteMe](#) 代码自动补全
- markdown-preview

3. Cheet Sheet

<https://devhints.io/vim>

4. b站 @ TheCW

<https://cloud.tencent.com/developer/column/4987>

Other vim

- Neovim

NeoVim 项目地址: [GitHub - neovim/neovim](#)

Vim 原项目地址: [GitHub - vim/vim](#)

- [supervim](#)
- [lunarvim](#)

有非常详细的官方文档，可以参考 `vim-wksp/vimrc-sample/lunarvim-config.lua`

- emacs



Throw Away Your Mouse!

- `vscode`

You may refer to -> `Vim-Wksp/vimrc-sample/vimrc-vscode`



- `help/man`
- [Ranger](#) -> available for macOS and Linux (Now, I'm using [joshuto](#))

Use vim key bindings to manage your files

You may refer to -> `Vim-Wksp/Usage-reference/ranger.md`

(If you use windows with powershell and posh, you can use [lf](#))

- Use vim key bindings when surfing the Internet
 - [vimium](#) -> Google Chrome
 - [vimari](#) -> Safari
 - [vimium c](#) -> edge

You may refer to -> `Vim-Wksp/Usage-reference/vimium.md`

- [sioyek](#) -> available for macOS, Windows and Linux

Use vim key bindings to view pdf files

You may refer to -> `Vim-Wksp/Usage-reference/sioyek.md`

- [lazygit](#)

- [tmux](#) -> vim mode in tmux