

A naïve vim tutorial

"Which editors are popular today? See this [Stack Overflow survey](#) (there may be some bias because Stack Overflow users may not be representative of programmers as a whole). [Visual Studio Code](#) is the most popular editor. [Vim](#) is the most popular command-line-based editor." [1]

从一个例子开始

How to exit the vim editor

- 如何产生一个随机字符串
 - 让新手退出 vim

How do I exit the Vim editor?

Asked 9 years, 2 months ago Active 2 months ago Viewed 2.5m times

▲ I'm stuck and cannot escape. It says:

4466 "type :quit<Enter> to quit VIM"

▼

★ 1029


vim vi

Share Improve this question Follow

edited May 14 '19 at 22:49

 Peter Mortensen
29k ● 21 ● 97 ● 124

asked Aug 6 '12 at 12:25

 jclancy
42.6k ● 5 ● 26 ● 32

正经的退出命令

- 保存并退出
 - `:wq`
- 强制退出
 - `:q!`

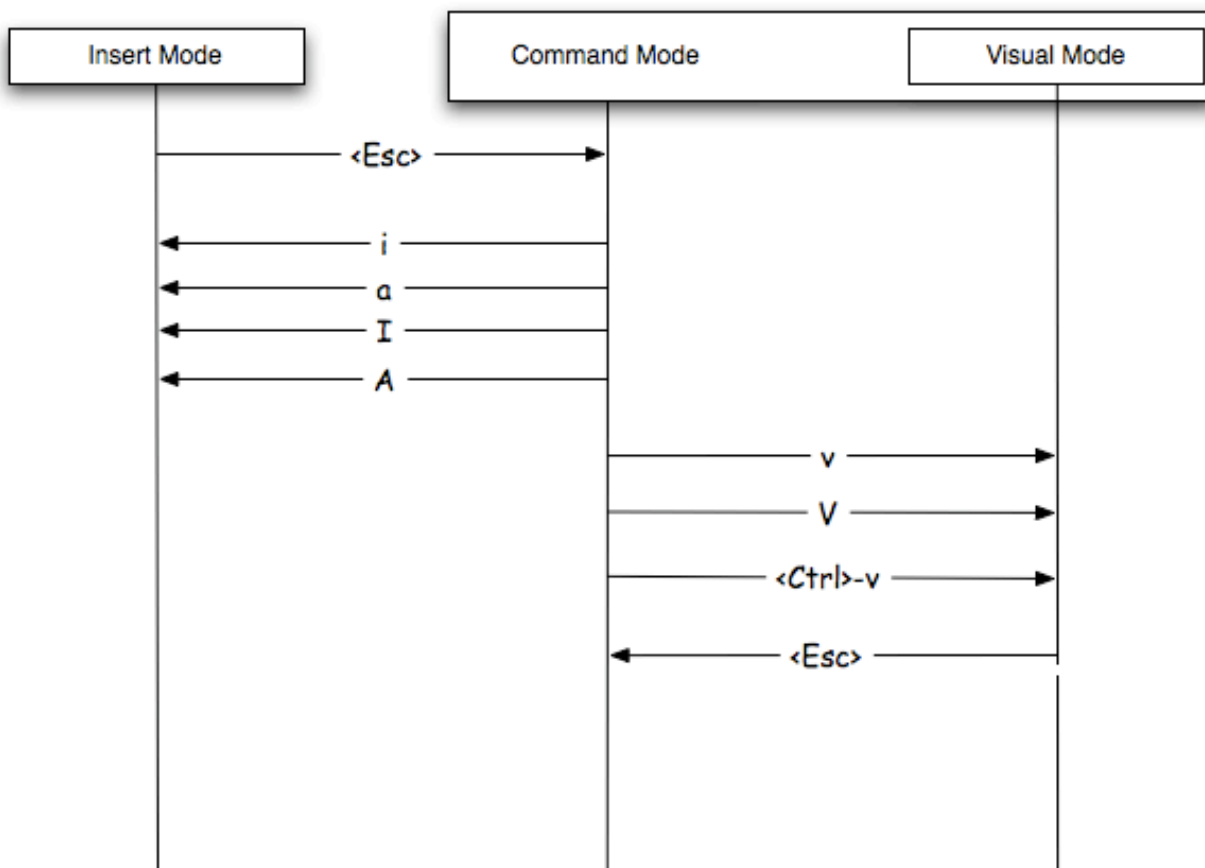
Vim 哲学 [1]

- Vim is programmable (with Vimscript and also other languages like Python), and Vim's interface itself is a programming language: keystrokes (with mnemonic names) are commands, and these commands are **composable**.
- Vim avoids the use of the mouse, because it's too slow.
- Vim even avoids using the arrow keys because it requires too much movement.

三个主要模式

默认左下角会显示当前所处的模式（无显示即为 Normal Mode）

- Normal Mode (Command Mode)
- Insert Mode: 插入文本
- Visual Mode: 选择文本



Normal mode => Insert mode

- 在下方插入一行: `o`
- 在上方插入一行: `O`
- 在当前光标后插入: `a`
- 在当前光标前插入: `i`
- 在行尾插入: `A`

- 在行首插入: `I`

光标移动

- 基础移动
 - 上下左右: `hjkl`
 - 移动到第一行: `gg`
 - 移动到最后一行/指定行: `G`
- 进阶移动
 - 移动到下一个单词的开头: `w`
 - 移动到下一个单词的结尾: `e`
 - 移动到上一个单词的开头: `b`
 - 移动到行首: `0`
 - 移动到第一个非空字符: `^`
 - 移动到行尾: `$`
 - 移动到匹配的括号处: `%`
 - 移动到变量定义处: `gd`
 - 移动到前一个没有匹配的左大括号处: `[{`

复制粘贴

- 复制: `y`
- 粘贴: `p`

不要害怕尝试

- Undo: `u`
- Redo: `C-r`

删除

- 删除整行: `dd`
- 删除字符: `x`
- 删除到行尾: `D`

替换

- 替换字符: `r`
- 替换到行尾: `C / Da`
- 修改大小写: `~`

Composable

- 删除2个单词 `d2w`
- 删除单词, 做两遍 `2dw`
- `2d2w`
- `d5j` = 在 visual mode 下选中后5行, 再按 `d`

搜索替换

- 搜索下一个: `/`
- 搜索上一个: `?`
- 快速搜索当前单词: `#/*`
- 将 range 范围内的 from 替换为 to: `:[range]s/from/to/[flags]`

range 列表 [2]

Range	Description	Example
<code>21</code>	line 21	<code>:21s/old/new/g</code>
<code>1</code>	first line	<code>:1s/old/new/g</code>
<code>\$</code>	last line	<code>:\$s/old/new/g</code>
<code>%</code>	all lines (same as <code>1, \$</code>)	<code>:%s/old/new/g</code>
<code>21,25</code>	lines 21 to 25 inclusive	<code>:21,25s/old/new/g</code>
<code>21,\$</code>	lines 21 to end	<code>:21,\$s/old/new/g</code>
<code>.,\$</code>	current line to end	<code>:.,\$s/old/new/g</code>
<code>.,+1,\$</code>	line <i>after</i> current line to end	<code>:.+1,\$s/old/new/g</code>
<code>.,.+5</code>	six lines (current to current+5 inclusive)	<code>:. ,.+5s/old/new/g</code>
<code>.,.5</code>	same (<code>.5</code> is interpreted as <code>.,+5</code>)	<code>:. ,.5s/old/new/g</code>

宏

生成从 1 到 1000 的序列

- 录制宏: `q`
- 运行宏: `@`

修饰词

- `i` inner
- `a` around
- `t` till
- `f` find

Split window

- `sp`
- `vsp`
- 移动: `C-w [hjkl]`

Mark

- 生成标签 `m`
- 跳转 ```

注释不需要的大段代码

- `visual block`
- 使用注释插件 `vim-commentary` / `nerdcommenter`
 - 安装插件
 - `mkdir -p ~/.vim/pack/vendor/start`
 - 将插件 clone 到该目录下即可

man in vim

- `:help`

在 `ext4` 的基础上开发 `ext5`

下载 linux 源码

生成 tags 方便跳转

- `ctags -R .`
- 查找 `inode_operations` 的定义
 - No tags file
 - 当前目录不存在 tags 文件
 - 在父目录中寻找 tags 文件: 在 `.vimrc` 中添加 `set tags=./tags;,tags`
- 直接打开定义 `inode_operations` 的文件
 - `vim -t inode_opeartions`

将 `ext4` 替换为 `ext5`

以 `fast_commit.c` 为例

将 range 范围内的 from 替换为 to: `:[range]s/from/to/[flags]`

- `:%s/ext4/ext5`
 - 仍有 `ext4` 未被替换
- 替换同一行中的多个 `ext4`
 - 使用 flag `g` 替换: `%s/ext4/ext5/g`
- 存在 `Ext4` 未被替换
 - `%s/Ext4/Ext5/g`
- 存在 `EXT4` 未被替换
 - `%s/EXT4/EXT5/g`

使用了3条命令将 `ext4` 全部替换为 `ext5` , 如何简化

- 正则表达式
 - `:%s/\([Ee][Xx][Tt]\)4/\15/g`
- 我们只需要大小写不敏感即可
 - `:%s/\(ext\)4/\15/ig`
- 其他办法
 - `\zs` 与 `\ze`: `ext` 只用于匹配, 不需要替换
 - `:%s/ext\zs4/5/ig`

flags [3]

flag	作用
&	复用上次替换命令的flags
g	替换每行的所有匹配值（默认只替换每行的第一个匹配值）
c	替换前需确认
e	替换失败时不报错
i	大小写不敏感
I	大小写敏感

批量替换多个文件内的 `ext4`

```
:args *.c *.h
:argdo %s/ext4/ext5/g
:argdo update
:argdo exit
```

意外收获

`sed -i "[range]s/from/to/[flags]" filename` 将文件 `filename` 内的 `from` 替换为 `to`

- `-i` edit files in place

查看 `mkdir syscall`

- 搜索 `syscall_define\d(mkdir`
- `vim filename +linenumber`

其他插件（可以问问其他同学经常使用哪些插件）

- `gitgutter`
- `easymotion`
- `fzf`
- `vim-linux-coding-style`
- `cscope`

vimrc [1]: <https://missing-semester-cn.github.io/2020/files/vimrc>

Vim 键位使用广泛

- `shell`
 - `bash: set -o vi`
 - `zsh: bindkey -v`
 - `fish: fish_vi_key_bindings`
- `ranger`
- `vimium` Chrome 插件
- [Vscode 插件](#) 已被安装 3.15M 次（截至 2021.11.17）

Tutorial

- `vimtutor`
- <https://vimsnake.com/>
- <http://www.vimgenius.com/>
- <https://vim-adventures.com/>
- MIT missing semester: <https://missing.csail.mit.edu/2020/editors/> <https://missing-semester-cn.github.io/2020/editors/>
- 键位图: <http://www.viemu.com/vi-vim-cheat-sheet.gif>

Reference:

[1] Editors (Vim), <https://missing.csail.mit.edu/2020/editors/>

[2] Ranges, <https://vim.fandom.com/wiki/Ranges>

[3] Search and Substitute, [https://github.com/iggrcredible/Learn-Vim/blob/master/ch12_search_and_substitut
e.md](https://github.com/iggrcredible/Learn-Vim/blob/master/ch12_search_and_substitute.md)