

Vim

Vim

Vim介绍

Vim学习方法、学习曲线和训练方法

Vimtutor

基本使用

光标移动

插入(insert)

删除(delete)

替换和更改

置入类命令

Vim 键位

Vim介绍

Vim的介绍网上一大堆，弱小如我既不能如Vim之父一样阐述的系统全面，也不能像其它大佬把Vim玩的出神入化，只能略述：**Vim是提高你编程效率以及更好适应命令行环境的利器**

Vim学习方法、学习曲线和训练方法

1.学习方法：边用边学，不要强记，和Markdown语法和LaTeX语法一样，用到哪不记得了查一下，然后不要埋怨自己怎么记不住，大家都一样，为“手熟尔”。晚上资料很多也很杂，挑选出适合新手看完能直接快速上手的不多，个人建议在 `Git bash` 命令行中直接输入 `vimtutor` 然后进入学习对初期效果最佳。之后可以找Github上一些其它资料进行提升，再最后配置自己的 `vim`。

2.学习曲线：Vim的学习曲线是那种对新手初学不友好的，就如同windows对比linux的学习一样。效率的大幅提升只有在时间砸入较多后才会明显体现。但是不要慌张，这就好比闭关，坚持练习就好。

3.看完教程后，每天一般时间用 `vim` 的模式来打码，剩下时间用 `IDE` 的正常模式。比如刷LeeCode就正常，用Clion或Pycharm打的时候就开启 `vim` 模式，Jetbrain全家桶的软件都提供 `vim` 的插件，File->plugins进入后手动搜索，选最高下载量的就行。

Vimtutor

基本使用

- 左 `h` 下 `j` 上 `k` 右 `l` 来移动阅读程序段或文本
- `:wq` 保存并退出 `:w` 保存当前文件 `:q` 退出 `:q!` 强制退出 `:qall` 有多个窗口时退出全部
- `< Esc >` 回到normal模式

光标移动

- 左 `h` 下 `j` 上 `k` 右 `l`
- `[n]w` `[n]`里填具体的数字，跳到几个字母以后
 - 输入 `2w` 使光标向前移动两个单词
- `e` 移动到第`n`个单词的末尾
 - 输入 `3e` 使光标向前移动到第三个单词的末尾
- `0` 移动到光标所在行的头部

插入(insert)

- `i` 在光标所在位置进行插入
- `A` 在光标所在行的最后进行插入文本
 - 看下面这个例子，一开始你会发现好像无法移动光标到 `.` 后面，怎么按 `l` 也没用

```
this line of words is cleaned up|
```

然后按下 `shift+a` 后你就可以发下如下的情形。

```
---> this line of words is cleaned up.|  
-- 插入 --
```

这时你就可以开心地继续在行末键入内容了。

- `a` 在光标所在位置后一个进行插入字符
 - 注意看以下的区别:

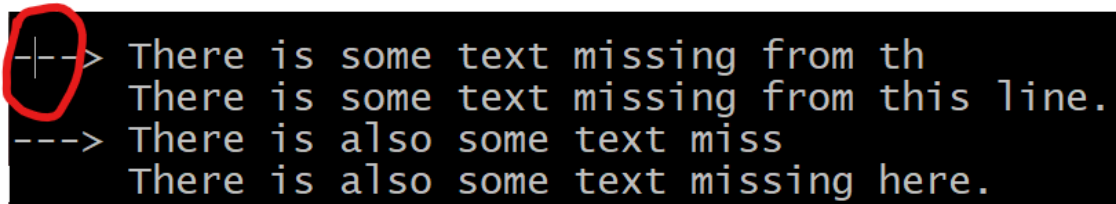
- 初始状态，光标位于最左侧：

```
---> There is some text missing from th  
      There is some text missing from this line.  
---> There is also some text miss  
      There is also some text missing here.
```

- 按 `A` 后看：

```
---> There is some text missing from th|  
      There is some text missing from this line.  
---> There is also some text miss  
      There is also some text missing here.
```

- 按 `a` 后看：



```
-|--> There is some text missing from th
      There is some text missing from this line.
---> There is also some text miss
      There is also some text missing here.
```

删除(delete)

- `dw` 删除从当前光标当前位置直到下一个单词起始处，不包括它的第一个字符。
 - 同时 `d2w` 就是删除光标后两个单词，中间的数字就是要删除的字的个数
 - 此处有一个组合操作公式关于Vim的一些操作 `d` `number` (数字) `motion` 拓展出来是如下的形式：
 - 在正常模式下修改命令的格式是：
`operator` `[number]` `motion`
其中：
`operator` - 操作符，代表要做的事情，比如 `d` 代表删除
`[number]` - 可以附加的数字，代表动作重复的次数
`motion` - 动作，代表在所操作的文本上的移动，例如 `w` 代表单词(word)，
`$` 代表行末等等。
- `de` 删除从当前光标当前位置直到单词末尾，包括最后一个字符。，`e`我没有查到确切的代表哪个单词，我猜测是 `end of word` 的`e`，这样好记一点。
- `d$` 删除从当前光标当前位置直到当前行末

替换和更改

- `r` replace
 - 请移动光标到第一个出错的位置，接着输入 `r` 和要替换成的字符，这样就能将错误替换掉
- `c`
 - `cw` 更改光标后一个错误单词
 - `ce` 更改光标后整个错误句子

此处 `s` 可以记忆为 `string` 的意思， `s/ole/new/g`

置入类命令

- `dd` 后输入 `p` 将最后一次删除的内容置入光标后
- `v` 进入visualisation可视化模式后，移动光标编程高亮的部分就是被选中的部分，对选中的部分按 `y`，再让光标移动到你需粘贴到的地方，按 `p` 就把先前那部分复制到新的位置了。

Vim 键位

vi / vim 键盘图

Esc

命令
模式

~ 转换大小写	! 外部过滤器	@ 运行宏	# prev ident	\$ 行尾	% 括号匹配	^ "软" 行首	& 重复:s	* next ident	(句首) 下一句首	"soft" bol down	+ 后一行行首
. 跳转到标注	1	2	3	4	5	6	7	8	9	0 "硬" 行首	- 前一行行首	= 自动格式 ³
Q 切换至ex模式	W 下一单词	E 词尾	R 替换模式	T back 'till	Y 拷贝行	U 撤消行内命令	I 到行首插入	O 分段(前)	P 粘贴(前)	{ 段首	}	段尾
q 录制宏	w 下一单词	e 词尾	r 替换字符	t 'till	y 拷贝 ^{1,3}	u 撤消命令	i 插入模式	o 分段(后)	p 粘贴(后)	[杂项]	杂项
A 在行尾附加	S 删除行并插入	D 删除至行尾	F 行内字符反向查找	G 文尾/行号	H 屏幕顶行	J 合并两行	K 帮助	L 屏幕底行	:	ex 命令	! 寄存器标识	行首/列
a 附加	s 删除字符并插入	d 删除 ^{1,3}	f 行内字符查找	g 附加命令 ⁶	h ←	j ↓	k ↑	l →	:	重复 u/T/f/F	! 跳转到标注的行首	\ 未用!
Z 退出 ⁴	X 退格	C 修改至行末 ^{1,3}	V 可视行模式	B 前一单词	N 查找上一处	M 屏幕中间行	< 反缩进 ³	> 缩进 ³	?	向前搜索		
z 附加命令 ⁵	x 删除(字符)	c 修改	v 可视模式	b 前一单词	n 查找下一处	m 设置标注	, 反向	.	重复命令	/	向后搜索	

动作	移动光标, 或者定义操作的范围
命令	直接执行的命令, 红色命令 进入编辑模式
操作	后面跟随表示操作范围的指令
extra	特殊功能, 需要额外的输入
q.	后跟字符参数

w,e,b 命令

小写(b): quux(foo, bar, baz);
大写(B): QUUX(foo, BAR, BAZ);

主要ex命令:

:w (保存), :q (退出), :q! (不保存退出)
:ef (打开文件 f),
:%s/x/y/g ('y' 全局替换 'x'),
:h (帮助 in vim), :new (新建文件 in vim),

其它重要命令:

CTRL-R: 重复 (vim),
CTRL-F/-B: 上翻/下翻,
CTRL-E/-Y: 上滚/下滚,
CTRL-V: 块可视模式 (vim only)

可视模式:

漫游后对选中的区域执行操作 (vim only)

备注:

- (1) 在 拷贝/粘贴/删除 命令前使用 "x (x=a..z,*)" 使用命令的寄存器('剪贴板') (如: "ay\$ 拷贝剩余的行内容至寄存器 'a'")
- (2) 命令前添加数字 多遍重复操作 (e.g.: 2p, d2w, 5i, d4j)
- (3) 重复本字符在光标所在行执行操作 (dd = 删除本行, >> = 行首缩进)
- (4) ZZ 保存退出, ZQ 不保存退出
- (5) zt: 移动光标所在行至屏幕顶端, zb: 底端, zz: 中间
- (6) gg: 文首 (vim only), gf: 打开光标处的文件名 (vim only)

原图: www.viemu.com

翻译: fdl (linuxsir)

● 第一讲小结

1. 光标在屏幕文本中的移动既可以用箭头键, 也可以使用 hjkl 字母键。
h (左移) j (下行) k (上行) l (右移)
2. 欲进入 Vim 编辑器(从命令行提示符), 请输入: vim 文件名 <回车>
3. 欲退出 Vim 编辑器, 请输入 :q! <回车> 放弃所有改动。
或者输入 :wq <回车> 保存改动。
4. 在正常模式下删除光标所在位置的字符, 请按: x
5. 欲插入或添加文本, 请输入:

i 输入欲插入文本 在光标前插入文本
A 输入欲添加文本 在一行后添加文本

特别提示: 按下 键会带您回到正常模式或者撤消一个不想输入或部分完整的命令。

好了, 第一讲到此结束。下面接下来继续第二讲的内容。

● 第二讲小结

1. 欲从当前光标删除至下一个单词，请输入：dw
2. 欲从当前光标删除至当前行末尾，请输入：d\$
3. 欲删除整行，请输入：dd
4. 欲重复一个动作，请在它前面加上一个数字：2w
5. 在正常模式下修改命令的格式是：

operator [number] motion

其中：

operator - 操作符，代表要做的事情，比如 d 代表删除

[number] - 可以附加的数字，代表动作重复的次数

motion - 动作，代表在所操作的文本上的移动，例如 w 代表单词(word)，
\$ 代表行末等等。

6. 欲移动光标到行首，请按数字0键：0
7. 欲撤消以前的操作，请输入：u (小写的u)
欲撤消在一行中所做的改动，请输入：U (大写的U)
欲撤消以前的撤消命令，恢复以前的操作结果，请输入：CTRL-R

● 第三讲小结

1. 要重新置入已经删除的文本内容，请按小写字母 p 键。该操作可以将已删除的文本内容置于光标之后。如果最后一次删除的是一个整行，那么该行将置于当前光标所在行的下一行。
2. 要替换光标所在位置的字符，请输入小写的 r 和要替换掉原位置字符的新字符即可。
3. 更改类命令允许您改变从当前光标所在位置直到动作指示的位置中间的文本。比如输入 ce 可以替换当前光标到单词的末尾的内容；输入 c\$ 可以替换当前光标到行末的内容。
4. 更改类命令的格式是：

c [number] motion

● 第四讲小结

1. CTRL-G 用于显示当前光标所在位置和文件状态信息。
G 用于将光标跳转至文件最后一行。
先敲入一个行号然后输入大写 G 则是将光标移动至该行号代表的行。
gg 用于将光标跳转至文件第一行。
2. 输入 / 然后紧随一个字符串是在当前所编辑的文档中正向查找该字符串。
输入 ? 然后紧随一个字符串则是在当前所编辑的文档中反向查找该字符串。
完成一次查找之后按 n 键是重复上一次的命令，可在同一方向上查找下一个匹配字符串所在；或者按大写 N 向相反方向查找下一匹配字符串所在。
CTRL-O 带您跳转回较旧的位置，CTRL-I 则带您到较新的位置。
3. 如果光标当前位置是括号(、)、[、]、{、}，按 % 会将光标移动到配对的括号上。
4. 在一行内替换头一个字符串 old 为新的字符串 new，请输入 :s/old/new
在一行内替换所有的字符串 old 为新的字符串 new，请输入 :s/old/new/g
在两行内替换所有的字符串 old 为新的字符串 new，请输入 :#,#s/old/new/g
在文件内替换所有的字符串 old 为新的字符串 new，请输入 :%s/old/new/g
进行全文替换时询问用户确认每个替换需添加 c 标志 :%s/old/new/gc

- 第五讲小结

1. `!command` 用于执行一个外部命令 `command`。

请看一些实际例子：

(MS-DOS) (Unix)

`!dir` `!ls` - 用于显示当前目录的内容。

`!del FILENAME` `!rm FILENAME` - 用于删除名为 `FILENAME` 的文件。

2. `:w FILENAME` 可将当前 VIM 中正在编辑的文件保存到名为 `FILENAME` 的文件中。
3. `v motion :w FILENAME` 可将当前编辑文件中可视模式下选中的内容保存到文件 `FILENAME` 中。
4. `:r FILENAME` 可提取磁盘文件 `FILENAME` 并将其插入到当前文件的光标位置后面。
5. `:r !dir` 可以读取 `dir` 命令的输出并将其放置到当前文件的光标位置后面。

- 第六讲小结

1. 输入小写的 `o` 可以在光标下方打开新的一行并进入插入模式。
输入大写的 `O` 可以在光标上方打开新的一行。
2. 输入小写的 `a` 可以在光标所在位置之后插入文本。
输入大写的 `A` 可以在光标所在行的行末之后插入文本。
3. `e` 命令可以使光标移动到单词末尾。
4. 操作符 `y` 复制文本，`p` 粘贴先前复制的文本。
5. 输入大写的 `R` 将进入替换模式，直至按 `Esc` 键回到正常模式。
6. 输入 `:set xxx` 可以设置 `xxx` 选项。一些有用的选项如下：
`'ic' 'ignorecase'` 查找时忽略字母大小写
`'is' 'incsearch'` 查找短语时显示部分匹配
`'hls' 'hlsearch'` 高亮显示所有的匹配短语
选项名可以用完整版本，也可以用缩略版本。
7. 在选项前加上 `no` 可以关闭选项： `:set noic`

- 第七讲小结

1. 输入 `:help` 或者按 `F1` 键或 `Ctrl-H` 键可以打开帮助窗口。
2. 输入 `:help cmd` 可以找到关于 `cmd` 命令的帮助。
3. 输入 `CTRL-W CTRL-W` 可以使您在窗口之间跳转。
4. 输入 `:q` 以关闭帮助窗口
5. 您可以创建一个 `vimrc` 启动脚本文件用来保存您偏好的设置。
6. 当输入 `:` 命令时，按 `CTRL-D` 可以查看可能的补全结果。
按 `Tab` 可以使用一个补全。