

笔记本: Ubuntu 18.04  
创建时间: 2019/4/21 14:51  
标签: vim  
URL: <https://coolshell.cn/articles/5426.html>

---

# 简明 VIM 练级攻略

□ 2011年09月07日 □ 陈皓 □ 567 条评论 □  
925,104 人阅读

---

vim的学习曲线相当的大（参看各种文本编辑器的学习曲线），所以，如果你一开始看到的是一大堆VIM的命令分类，你一定会对这个编辑器失去兴趣的。下面的文章翻译自《[Learn Vim Progressively](#)》，我觉得这是给新手最好的VIM的升级教程了，没有列举所有的命令，只是列举了那些最有用的命令。非常不错。

——正文开始

---

你想以最快的速度学习人类史上最好的文本编辑器VIM吗？你先得懂得如何在VIM幸存下来，然后一点一点地学习各种戏法。

Vim the Six Billion Dollar editor

*Better, Stronger, Faster.*

学习 vim 并且其会成为你最后一个使用的文本编辑器。没有比这个更好的文本编辑器了，非常

地难学，但是却不可思议地好用。

我建议下面这四个步骤：

1. 存活
2. 感觉良好
3. 觉得更好，更强，更快
4. 使用VIM的超能力

当你走完这篇文章，你会成为一个vim的superstar。

在开始学习以前，我需要给你一些警告：

- 学习vim在开始时是痛苦的。
- 需要时间
- 需要不断地练习，就像你学习一个乐器一样。
- 不要期望你能在3天内把vim练得比别的编辑器更有效率。
- 事实上，你需要2周时间的苦练，而不是3天。

## 第一级 - 存活

1. 安装 vim
2. 启动 vim
3. **什么也别干！** 请先阅读

当你安装好一个编辑器后，你一定会想在其中输入点什么东西，然后看看这个编辑器是什么样子。但vim不是这样的，请按照下面的命令操作：

- 启动Vim后，vim在 *Normal* 模式下。
- 让我们进入 *Insert* 模式，请按下键 `i`。(陈皓注：你会看到vim左下角有一个`-insert-`字样，表示，你可以以插入的方式输入了)
- 此时，你可以输入文本了，就像你用“记事本”一样。
- 如果你想返回 *Normal* 模式，请按 `ESC` 键。

现在，你知道如何在 *Insert* 和 *Normal* 模式下切换了。下面是一些命令，可以让你在 *Normal* 模式下幸存下来：

- `i` → *Insert* 模式，按 `ESC` 回到 *Normal* 模式.
- `x` → 删当前光标所在的一个字符。
- `:wq` → 存盘 + 退出 (`:w` 存盘, `:q` 退出) (陈皓注：`:w`后可以跟文件名)
- `dd` → 删除当前行，并把删除的行存到剪贴板里
- `p` → 粘贴剪贴板

**推荐:**

- `hjkl` (强烈推荐使用其移动光标, 但不必需) → 你也可以使用光标键 (`←↑→`). 注: `j` 就像下箭头。
- `:help <command>` → 显示相关命令的帮助。你也可以就输入 `:help` 而不跟命令。(陈皓注: 退出帮助需要输入`:q`)

你能在vim幸存下来只需要上述的那5个命令, 你就可以编辑文本了, 你一定要把这些命令练成一种下意识的状态。于是你就可以开始进阶到第二级了。

当是, 在你进入第二级时, 需要再说一下 *Normal* 模式。在一般的编辑器下, 当你需要copy一段文字的时候, 你需要使用 `Ctrl` 键, 比如: `Ctrl-C`。也就是说, Ctrl键就好像功能键一样, 当你按下了功能键Ctrl后, C就不再是C了, 而且就是一个命令或是一个快捷键了, **在VIM的Normal模式下, 所有的键就是功能键了。**这个你需要知道。

标记:

- 下面的文字中, 如果是 `Ctrl-λ` 我会写成 `<C-λ>` .
- 以 `:` 开始的命令你需要输入 `<enter>` 回车, 例如 — 如果我写成 `:q` 也就是说你要输入 `:q<enter>` .

## 第二级 - 感觉良好

上面的那些命令只能让你存活下来，现在是时候学习一些更多的命令了，下面是我的建议：（陈皓注：所有的命令都需要在Normal模式下使用，如果你不知道现在在什么样的模式，你就狂按几次ESC键）

## 1. 各种插入模式

- *a* → 在光标后插入
- *o* → 在当前行后插入一个新行
- *O* → 在当前行前插入一个新行
- *cw* → 替换从光标所在位置后到一个单词结尾的字符

## 2. 简单的移动光标

- *0* → 数字零，到行头
- *^* → 到本行第一个不是blank字符的位置（所谓blank字符就是空格，tab，换行，回车等）
- *\$* → 到本行行尾
- *g\_* → 到本行最后一个不是blank字符的位置。
- */pattern* → 搜索 *pattern* 的字符串（陈皓注：如果搜索出多个匹配，可按n键到下一个）

## 3. 拷贝/粘贴（陈皓注：p/P都可以，p是表示在当前位置之后，P表示在当前位置之前）

- *P* → 粘贴
- *yy* → 拷贝当前行当行于 *ddP*

#### 4. Undo/Redo

- *u* → *undo*
- *<C-r>* → *redo*

#### 5. 打开/保存/退出/改变文件(Buffer)

- *:e <path/to/file>* → 打开一个文件
- *:w* → 存盘
- *:saveas <path/to/file>* → 另存为 *<path/to/file>*
- *:x*, *ZZ* 或 *:wq* → 保存并退出 (*:x* 表示仅在需要时保存, *ZZ*不需要输入冒号并回车)
- *:q!* → 退出不保存 *:qa!* 强行退出所有的正在编辑的文件, 就算别的文件有更改。
- *:bn* 和 *:bp* → 你可以同时打开很多文件, 使用这两个命令来切换下一个或上一个文件。  
(陈皓注: 我喜欢使用 *:n* 到下一个文件)

花点时间熟悉一下上面的命令, 一旦你掌握他们了, 你就几乎可以干其它编辑器都能干的事了。

但是到现在为止，你还是觉得使用vim还是有点笨拙，不过没关系，你可以进阶到第三级了。

### 第三级 - 更好，更强，更快

先恭喜你！你干的很不错。我们可以开始一些更为有趣的事了。在第三级，我们只谈那些和vi可以兼容的命令。

# 更好

下面，让我们看一下vim是怎么重复自己的：

1. `.` → (小数点) 可以重复上一次的命令
2. `N<command>` → 重复某个命令N次

下面是一个示例， 打开一个文件你可以试试下面的命令：

- [illegible]

*desu desu desu desu desu desu  
desu desu desu desu desu desu  
desu desu desu desu desu desu  
desu desu desu desu desu desu  
desu desu desu desu desu desu  
desu desu desu desu desu desu  
desu desu desu* "

- **.** → 重复上一个命令—— 100 "desu ".
- **3.** → 重复 3 次 "desu" (注意: 不是 300, 你看, VIM多聪明啊).

## 更强

你要让你的光标移动更有效率, 你一定要了解下面的这些命令, **千万别跳过**。

1. **N G** → 到第 N 行 (陈皓注: 注意命令中的G是大写的, 另我一般使用 :N 到第N行, 如 :137 到第137行)
2. **gg** → 到第一行。(陈皓注: 相当于1G, 或 :1)
3. **G** → 到最后一行。
4. 按单词移动:

1. **w** → 到下一个单词的开头。

2. **e** → 到下一个单词的结尾。

> 如果你认为单词是由默认方式, 那么就用小写的e和w。默认上来



说，一个单词由字母，数字和下划线组成（陈皓注：程序变量）

> 如果你认为单词是由blank字符分隔符，那么你需要使用大写的E和W。（陈皓注：程序语句）

```
      ew      E W
      ||      | |
x = (name_1,vision_3); # this is a comment.
```

下面，让我来说说最强的光标移动：

- **%**：匹配括号移动，包括 **(, {, [**.  
（陈皓注：你需要把光标先移到括号上）
- **\*** 和 **#**：匹配光标当前所在的单词，移动光标到下一个（或上一个）匹配单词（\*是下一个，#是上一个）

相信我，上面这三个命令对程序员来说是相当强大的。

## 更快

你一定要记住光标的移动，因为很多命令都可以和这些移动光标的命令连动。很多命令都可以如下来干：

```
<start position><command><end position>
```

例如 **0y\$** 命令意味着：

- `0` → 先到行头
- `y` → 从这里开始拷贝
- `$` → 拷贝到本行最后一个字符

你可以输入 `ye`，从当前位置拷贝到本单词的最后一个字符。

你也可以输入 `y2/foo` 来拷贝2个 “foo” 之间的字符串。

还有很多时间并不一定你就一定要按y才会拷贝，下面的命令也会被拷贝：

- `d` (删除)
- `v` (可视化的选择)
- `gU` (变大写)
- `gu` (变小写)
- 等等

(陈皓注：可视化选择是一个很有意思的命令，你可以先按v，然后移动光标，你就会看到文本被选择，然后，你可能d，也可y，也可以变大写等)

## 第四级 - Vim 超能力

你只需要掌握前面的命令，你就可以很舒服的使用VIM了。但是，现在，我们向你介绍的是VIM杀手级的功能。下面这些功能是我只用vim的原因。

在当前行上移动光标: `0` `^` `$` `f` `F` `t` `T` ,  
;

- `0` → 到行头
- `^` → 到本行的第一个非blank字符
- `$` → 到行尾
- `g_` → 到本行最后一个不是blank字符的位置。
- `fa` → 到下一个为a的字符处, 你也可以fs到下一个为s的字符。
- `t,` → 到逗号前的第一个字符。逗号可以变成其它字符。
- `3fa` → 在当前行查找第三个出现的a。
- `F` 和 `T` → 和 `f` 和 `t` 一样, 只不过是相反方向。

```
0  ^      fi      t)      4fi      g_  $  
|  |      |      |      |      |      |  
x = (name_1,vision_3); #this is a comment.
```

还有一个很有用的命令是 `dt"` → 删除所有的内容, 直到遇到双引号—— `"`。

**区 域 选 择** `<action>a<object>` **或** `<action>i<object>`

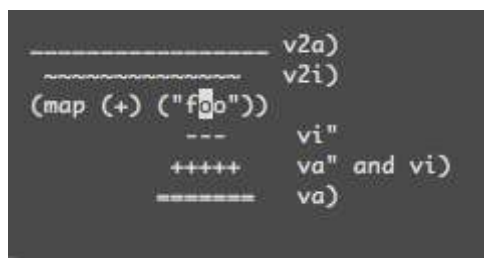
在visual 模式下, 这些命令很强大, 其命令格式为

`<action>a<object>` 和 `<action>i<object>`

- action可以是任何的命令，如 **d** (删除), **y** (拷贝), **v** (可以视模式选择)。
- object 可能是: **w** 一个单词, **W** 一个以空格为分隔的单词, **s** 一个句子, **p** 一个段落。也可以是一个特别的字符: **"**、**'**、**)**、**}**、**]**。

假设你有一个字符串 `(map (+) ("foo"))`. 而光标键在第一个 **o** 的位置。

- **vi** → 会选择 *foo* .
- **va** → 会选择 *"foo"* .
- **vi)** → 会选择 *"foo"* .
- **va)** → 会选择 *("foo")* .
- **v2i** → 会选择 *map (+) ("foo")*
- **v2a)** → 会 选 择 *(map (+) ("foo"))*



**块操作:** **<C-v>**

块操作，典型的操作: **0 <C-v> <C-d> I--**  
**[ESC]**

- **^** → 到行头

- **<C-v>** → 开始块操作
- **<C-d>** → 向下移动 (你也可以使用hjkl来移动光标, 或是使用%, 或是别的)
- **I-- [ESC]** → I是插入, 插入 "--", 按ESC键来为每一行生效。

```
-- Screen size
type Size = (Int,Int)
type Point = (Int,Int)
type TimePoint = (Int,Int,Int)
screenSize = (25,79)

type Drawer = Size -> Int -> [Float]

fabs x | x < 0 = -x
```

在Windows下的vim, 你需要使用 **<C-q>** 而不是 **<C-v>**, **<C-v>** 是拷贝剪贴板。

### 自动提示: **<C-n>** 和 **<C-p>**

在 Insert 模式下, 你可以输入一个词的开头, 然后按 **<C-p>**或是**<C-n>**, 自动补齐功能就出现了.....

```
- Lovecraft
- LyX
- LaTeX
- XeLaTeX

Now I'll type: X<C-p>, L<C-n><C-n>.
█

7,0-1 All
```

### 宏录制: **qa** 操作序列 **q**, **@a**, **@@**

- **qa** 把你的操作记录在寄存器 **a**。
- 于是 **@a** 会replay被录制的宏。

- @@ 是一个快捷键用来replay最新录制的宏。

### 示例

在一个只有一行且这一行只有“1”的文本中，键入如下命令：

- qaYp<C-a>q →
  - qa 开始录制
  - Yp 复制行.
  - <C-a> 增加1.
  - q 停止录制.
- @a → 在1下面写下 2
- @@ → 在2 正面写下3
- 现在做 100@@ 会创建新的100行，并把数据增加到 103.



可视化选择: v, V, <C-v>

前面，我们看到了 <C-v> 的示例（在Windows下应该是<C-q>），我们可以使用 v 和 V。一旦被选好了，你可以做下面的事：

- **J** → 把所有的行连接起来（变成一行）
- **<** 或 **>** → 左右缩进
- **=** → 自动给缩进（陈皓注：这个功能相当强大，我太喜欢了）

```
// AUTOINDENT with = example => S-V$%=
// S-V => Visual select lines
// $ => go to end of line
// % => go to end of function (corresponding { )
// = => Auto indent the selection
int allcases ( char prefix[], int n, char *list[] ) {
char *c;
char **next_list = list;
ylog(prefix);
19,1 26%
```

在所有被选择的行后加上点东西：

- **<C-v>**
- 选中相关的行（可使用 **j** 或 **<C-d>** 或是 **/pattern** 或是 **%** 等.....）
- **\$** 到行最后
- **A**, 输入字符串，按 **ESC**。

```
#!/usr/bin/env zsh
echo "Hello"
echo "I will add something"
echo "in the end of each line"
echo "Lets do it"
echo "C-vG$A >&2[ESC]"
2,0-1 All
```

**分屏: **:split** 和 **vsplit**.**

下面是主要的命令，你可以使用VIM的帮助 **:help split**. 你可以参考本站以前的一篇文章VIM分屏。

- `:split` → 创建分屏 (`:vsplit` 创建垂直分屏)
- `<C-w><dir>` : `dir`就是方向, 可以是 `hjkl` 或是 `←↓↑→` 中的一个, 其用来切换分屏。
- `<C-w>_` (或 `<C-w>|`) : 最大化尺寸 (`<C-w>|` 垂直分屏)
- `<C-w>+` (或 `<C-w>-`) : 增加尺寸

```
enddiv
[Vim] the Six Billion Dollar editor
> Better, Stronger, Faster.
[
Learn [vim] and it will be your last text editor.
There isn't any better text editor I know.
Hard to learn, but it will pay a billion times.
:q                                     23,0-1      7%
```

## 结束语

- 上面是作者最常用的90%的命令。
- 我建议你每天都学1到2个新的命令。
- 在两到三周后, 你会感到vim的强大的。
- 有时候, 学习VIM就像是在死背一些东西。
- 幸运的是, vim有很多很不错的工具和优秀的文档。
- 运行vimtutor直到你熟悉了那些基本命令。



- 其在线帮助文档中你应该要仔细阅读的是 `:help usr_02.txt`。
- 你会学习到诸如 `!`， 目录， 寄存器， 插件等很多其它的功能。

学习vim就像学弹钢琴一样，一旦学会，受益无穷。

---

正文结束

---

对于vi/vim只是点评一点：这是一个你不需要使用鼠标，不需使用小键盘，只需要使用大键盘就可以完成很多复杂功能文本编辑的编辑器。不然，Visual Studio也不就会有vim的插件了。

(全文完)

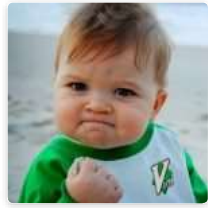


关注CoolShell微信公众账号和微信小程序

(转载本站文章请注明作者和出处 酷壳 – CoolShell , 请勿用于任何商业用途)

——=== 访问 [酷壳404页面](#) 寻找遗失儿童。 ===——

## 相关文章



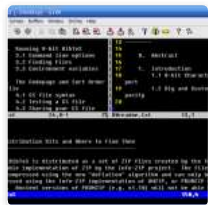
无插件Vim编程技巧



给程序员的VIM速查卡



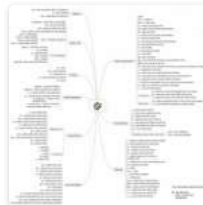
编译vim解决中文支持



Vim的分屏功能



《Vim Recipes》  
免费的Vim  
Cookbook



Vim命令速查卡

★★★★★ (131 人打了分, 平均分: 4.84)

□ 杂项资源, 编程工具

□ Vim