



- □□□ -

# 提高 Vim 使用效率的 12 个技巧

分享到： 20

[带你入门Vue2.0及案例开发](#)  
[Laravel和 AngularJS开发全栈知乎](#)

[强力django+杀手级xadmin打造上线标准...](#)  
[Android专项测试-Python篇10年测试经...](#)

本文由 [伯乐在线](#) - [Alick](#) 翻译，[黄利民](#) 校稿。未经许可，禁止转载！  
英文出处：[sheerun](#)。欢迎加入[翻译组](#)。

## 1. 使用空格键做为Leader（热键）

Leader是个非常有创意的设计，通过不同按键的顺序操作可以执行各种命令，而不需要通过组合键的方式，自从使用Leader之后我就很少使用ctrl-xxx的这种组合键。

长久以来我都是使用，做为Leader，直到我意识到我可以使⽤键盘上更加好用的捷按键：空格键（<Space>）。

```
1 let mapleader = "\<Space>"
```

这个完全颠覆了我的 Vim 操作效率，我现在可以使用双手任何一个大拇指进行操作，同时其它手指可以保持在键盘的主键区。因为 Leader 相当容易使用，所以我就将各种常用的操作进行了 Leader 映射。

## 2. 将最常用的操作映射到 Leader 操作

我首先识别出最经常使用的操作，并将它们映射到 Leader 操作，我经常是这么使用的：

使用 <Space>o 创建一个新文件:

```
1 noremap <Leader>o :CtrlP<CR>
```

使用 `<Space>w` 保存文件（比 `:w<Enter>` 明显快得多）：

```
1 noremap <Leader>w :w<CR>
```

使用 `<Space>p` 与 `<Space>y` 进行剪切板拷贝、粘贴：

```
1 vmap <Leader>y "+y
2 vmap <Leader>d "+d
3 nmap <Leader>p "+p
4 nmap <Leader>P "+P
5 vmap <Leader>p "+p
6 vmap <Leader>P "+P
```

使用 `<Space><Space>` 进入 Vim 编辑模式：

```
1 nmap <Leader><Leader> V
```

我强烈建议你找到最常用的操作并将其映射到 Leader 。

### 3. 使用区域扩展功能

为插件 [terryma/vim-expand-region](#) 做按键映射：

```
1 vmap v <Plug>(expand_region_expand)
2 vmap <C-v> <Plug>(expand_region_shrink)
```

这样我就可以：

- 按一次 `v` 选择一个字符。
- 再按一次 `v` 自动扩展选择一个单词。
- 再按一次 `v` 自动扩展选择一段代码。
- ...
- 以此类推...
- 按 `<C-v>` 回退上一次的選擇操作。

虽然 `vvv` 貌似比 `vp` 的操作更慢，但使用这种方式时，我压根就不需要考虑当前要选择哪些以及应该使用哪个组合键进行操作。

这么一来 `v` 把诸如 `viw`, `vaw`, `vi"`, `va"`, `vi(`, `va(`, `vi[`, `va[`, `vi{`, `va{`, `vip`, `vap`, `vit`, `vat`, ... 这些操作都给替代了, 这下你懂了吧。

## 4. 找到文本查找操作工具

我一直都不喜欢 Vim 中的查找和替换操作, 直到我在 [Vim wiki](#) 找到了以下的配置:

```
1 vnoremap <silent> s //e<C-r>=&selection=='exclusive'?'+1':''<CR><CR>
2   :<C-u>call histdel('search',-1)<Bar>let @/=histget('search',-1)<CR>gv
3 omap s :normal vs<CR>
```

这个直接替代了我常用的操作序列:

- 使用 `/something` 查找
- 使用 `cs` 替换第一个, 然后按 `<Esc>` 键
- 使用 `n.n.n.n.n.` 查找以及替换余下匹配项。

PS: 也可以考虑使用 Vim 7.4 提供的 `cn` 命令。

## 5. 尝试更多更棒的键盘映射

我每天都在使用这几个快捷操作键, 估计这已经帮我省出了几个月时间。

### 自动跳转到粘贴文本的最后

使用 `ppppp` 进行多行多次粘贴操作

```
1 vnoremap <silent> y y`]
2 vnoremap <silent> p p`]
3 nnoremap <silent> p p`]`]
```

### 避免缓冲区的内容在粘贴后被覆盖

通过以下的配置可以避免缓冲区的内容被删除的文本内容所覆盖 (放到 `~/.vimrc` 文件的最后)

```
1 " vp doesn't replace paste buffer
2 function! RestoreRegister()
3   let @" = s:restore_reg
4   return ''
5 endfunction
6 function! s:Repl()
```

```
7   let s:restore_reg = @"
8   return "p@=RestoreRegister()<cr>"
9   endfunction
10  vmap <silent> <expr> p <sid>Repl()
```

**译注：这里没有Title，估计是作者漏了，加了个。**

## 在文件中快速跳转

- 通过 12<Enter> 跳转到第 12 行（12G 我觉得不称手）
- 按 <Enter> 跳到行文件末尾。
- 按 <Backspace> 回到文件开始。

```
1  nnoremap <CR> G
2  nnoremap <BS> gg
```

## 快速选择粘贴的文本

```
1  noremap gV `[v`]
```

## 关闭无聊的窗口提示

```
1  map q: :q
```

## 6. 提高单元测试执行效率

我用 [vim-vroom](#) 插件以及相应的 tmux 配置来进行我的测试。

vim-room 缺省使用 <Leader>r 执行测试，由于我已将 Leader 映射为 <Space>，因此我通过 <Space>r 来运行测试工具。

由于测试在 tmux 的单独窗口运行，因此我可以边看测试进展共修订我的代码。

## 7. 使用 Ctrl-Z 返回 Vim

我经常要在shell下执行一些命令，我通过 Ctrl-Z 挂起 Vim，在完成 shell 命令执行之后，通过 <Enter> 重新回到 Vim。

使用 fg 退回 Vim 让我觉得难受，我只想通过 Ctrl-Z 在 Vim 与 Shell 之间切换，不过我没有找到解决方法，因此我写了个在 ZSH 下完美运行的脚本：

```
1 fancy-ctrl-z () {
2     if [[ $#BUFFER -eq 0 ]]; then
3         BUFFER="fg"
4         zle accept-line
5     else
6         zle push-input
7         zle clear-screen
8     fi
9 }
10 zle -N fancy-ctrl-z
11 bindkey '^Z' fancy-ctrl-z
```

如果你将上述代码放到 ~/.zshrc 文件中，你就可以在 shell 与 Vim 之间快速切换，你真值得去试试看。

## 8. 正确地配置 Tmux

在 OS X 下使用 Tmux 和 Vim 工具非常不方便，因为：

- 系统的剪切板处理功能很弱
- Vim 与 Tmux 的窗口切换操作不同
- Tmux 下执行命令的热键不同（使用 C-b）
- 在 Tmux 中拷贝模式超难用

我花了非常多的时间去修正上述配置，具体可以见以下：

### 配置使用 <C-Space> 作为 tmux 的热键

有些人习惯使用 <C-a> 作为热键，不过我是用这个热键回到行首，所以我这里就不细说了。使用 <C-Space> 的方式会更好用，原因我一会再说：

```
1 unbind C-b
2 set -g prefix C-Space
3 bind Space send-prefix
```

### 使用 <Space> 进入拷贝模式

设想一下，使用 <C-Space> <Space> 就可以直接进入 Tmux 的拷贝模式有多方便。

```
1 bind Space copy-mode
2 bind C-Space copy-mode
```

### 使用 y 和 reattach-to-user-namespace (基于 OSX)

在使用系统的剪切板之前，你需要先执行 `brew install reattach-to-user-namespace`

```
1 bind-key -t vi-copy y
2 copy-pipe "reattach-to-user-namespace pbcopy"
```

## 使用 `vim-tmux-navigator`

你要使用 `<C-h>`，`<C-j>`，`<C-k>`，`<C-l>` 这几个快捷键在 vim 和 tmux 的各种窗口内快速切换。

同时我建议使用 `<C-Space>l` 和 `<C-Space>j` 的映射配置来进行 Tmux 窗口分割操作，这个绝对比用 `<C-Space>%` 和 `<C-Space>|` 来得快：

```
1 bind j split-window -v
2 bind C-j split-window -v
3
4 bind l split-window -h
5 bind C-l split-window -h
```

参看我的 [tmux.conf](#) 文件，这里有更多的干货。

## 9. 提高 Git 工程中 Ctrl-P 的执行效率

将下面的内容添加到你的 `.vimrc` 文件中（配置使用 `<Ctrl-P>` 来使用 git 或 silver 查找工具来自动补全）：

```
1 let g:ctrlp_use_caching = 0
2 if executable('ag')
3     set grepprg=ag --nogroup --nocolor
4
5     let g:ctrlp_user_command = 'ag %s -l --nocolor -g "'
6 else
7     let g:ctrlp_user_command = ['.git', 'cd %s && git ls-files . -co --exclude-standard',
8 'find %s -type f']
9     let g:ctrlp_prompt_mappings = {
10         'AcceptSelection("e")': ['<space>', '<cr>', '<2-LeftMouse>'],
11     }
12 endif
```

我建议使用 [vim-scripts/gitignore](#) 插件。

## 10. 使用包管理器

[neobundle.vim](#) 是一个强大的管理 Vim 插件的工具：

- 你不需要手工管理 git 的子库（submodules）
- 能够**并行地**安装以及更新插件

- 它支持 [YouCompleteMe](#) 这类需要 **build** 的插件

```
1 NeoBundle 'Valloric/YouCompleteMe', {
2     'build' : {
3         'mac' : './install.sh',
4     },
5 }
```

- 出同样支持 [pry](#) 这类支持打补丁的插件：

```
1 NeoBundle 'rking/pry-de', {'rtp': 'vim/'}
```

## 11. 充分使用 Vim 的插件的优点

- [YouCompleteMe](#)
- [ack.vim](#) ( [ag.vim](#) 也不错 )
- [tpope/vim-commentary](#)
- [tpope/vim-rsi](#)
- [tpope/vim-endwise](#)
- [tpope/vim-fugitive](#) 主要使用 `:Gblame`
- [tpope/vim-repeat](#)
- [tpope/vim-sleuth](#)
- [mmozuras/vim-github-comment](#)
- [vim-airline](#) 并加上以下配置：

```
1 NeoBundle 'bling/vim-airline'
2 let g:airline_theme='powerlineish'
3 let g:airline_left_sep=''
4 let g:airline_right_sep=''
5 let g:airline_section_z=''
```

我是个 Ruby 程序员，所以我还使用一些 Ruby 的插件：

- [tpope/vim-rails](#)
- [vim-textobj-rubyblock](#) (使用 `var` , `vir` 查找 ruby 代码块)
- [ruby\\_pry](#)
- [AndrewRadev/splitjoin.vim](#) 配置以下映射 `nmap sj :SplitjoinSplit<cr>` `nmap sk :SplitjoinJoin<cr>`

## 12. 在服务器上快速配置 Vim

我经常需要服务上使用 Vim 进行配置，不幸的是 Vim 的缺省配置相当不合理。

一种方案是使用 [vim-sensible](#) 插件来生成压缩包，不过这个对我来说不够好用。我写 [vimrc](#) 的插件来对 Vim 做合理的初始化（特别是对 Ruby 开发者），这个插件配置 Vim 只使用 `~/.vimrc` 作为配置文件，同时它还包含了优化的配色、包管理工具以及多种开发语言的语法着色。

这意味着我不需要手工配置服务器上的 `~/.vim` 目录，而是通过以下操作就可以方便地在服务器上配置 Vim 环境：

```
1 git clone --recursive https://github.com/sheerun/vimrc.git ~/.vim
```

我还写了个 [dotfiles](#)，用于快速配置我的开发环境。

思考

用好 Vim 的关键在于，你要在软件开发过程中不断发现你所遇到的 Vim 问题，并积极处理它们。

处理方法可以是在 `.vimrc` 中添加键盘映射，或是在 google 上查找解决方案，要不就在 IRC 上提问，或者其他方法。

你用什么方法提高了 Vim 的使用效率？






想了解更多？

哈，雇我做你的 Vim 顾问吧，给我[写个邮件](#)吧。

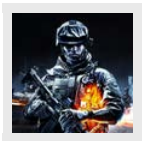
5 赞

15 收藏

8 评论



关于作者：Alick



这个码农很懒，啥也没有留下.

个人主页 · 我的文章 · 29 ·





## 相关文章

- 开发者的实用 Vim 插件 ( 3 ) · 2
- 开发者的实用 Vim 插件 ( 二 )
- 开发者的实用 Vim 插件 ( 一 )
- 编辑器之战: Vim 的复仇 · 3
- 作为 2016 年的开发者，你需要学习 Emacs 或 Vi · 5

## 可能感兴趣的话题

- 你们遇到过 因为机房温度高 天天下班要关服务器的吗 哪天加班要跟助理报... · 2
- Python GUI 多个对话框 · 2
- 网易2017春招笔试：堆砖块
- 朋友突然来问我，说是想转行来搞IT · 5
- 网易2017春招笔试：记单词 · 1
- 网易2017春招笔试：涂棋盘

登录后评论

直接登录     

## 最新评论



北海鲲鹏 ( 1 )

2015/06/12

使用空格键做为Leader ( 热键 ) 中漏了一个反斜杠`\'，应该是：  
let mapleader = "\'  
而不是  
let mapleader = ""

1 赞 回复



小编辑 ( 10 · )  
编辑

2015/06/12

@Alick @黄利民

1 赞 回复



Alick ( 29 · )  
码农-拉大东

2015/06/12

的确是有误，原文是：let mapleader = "\

赞 回复



周进林 ( 18 · )

2015/06/15

使用 w 文件（比 :w 明显快得多）  
漏翻了~~应该是“使用 w 保存文件”

2 赞 回复



Alick ( 29 · )  
码农-拉大东

2015/06/15

收到，已修订，非常感谢。

赞 回复



小编辑 ( 10 · )  
编辑

2015/06/15

不点赞的感谢是耍流氓，嘿嘿

1 赞 回复



不一和一一 ( )

2015/06/16

不错，精益求精的态度，赞

赞 回复



SCaffrey ( 10 )

2015/06/20

nnoremap o :CtrlP

CtrlP 是啥？

赞 回复



本周热门文章
本月热门文章
热门标签
<div>0 10 个常见的 Linux 终端仿真器</div> <div>1 我是小有成就，但我过不了白板面试</div> <div>2 趣文：程序员相亲指南</div> <div>3 没热情就不够格做程序员么？</div> <div>4 适用于开发者的最佳 Chrome 扩展...</div> <div>5 自动补全不算什么，一键直达目录才是...</div> <div>6 阿里巴巴 Java 开发手册评述</div> <div>7 聊聊分布式事务</div> <div>8 第 0 期技术微周刊，从经典的 Lin...</div> <div>9 Linux 命令行工具使用小贴士及技巧...</div>



业界热点资讯

更多 »



Ubuntu GNOME成为默认flavor 用户依然可安装Unity...

16 小时前 · 3



放弃 Unity 后 Canonical 宣布将裁员

20 小时前 · 3



### PostgreSQL 10 新增特性

20 小时前 · 2



### Nginx 创建认证的第三方模块程序

20 小时前 · 2



### Visual Studio Code 1.11 发布，包括重要更新

3 天前 · 14 · 2



## 精选工具资源

[更多资源 »](#)



### Caffe：一个深度学习框架

机器学习



### 静态代码分析工具清单：公司篇

静态代码分析



### HotswapAgent：支持无限次重定义运行时类与资源

开发流程增强工具



### 静态代码分析工具清单：开源篇（各语言）

静态代码分析 · 1



### Bugsnap：跨平台的错误监测

开发库

## 关于伯乐在线博客

在这个信息爆炸的时代，人们已然被大量、快速并且简短的信息所包围。然而，我们相信：过多“快餐”式的阅读只会令人“虚胖”，缺乏实质的内涵。伯乐在线内容团队正试图以我们微薄的力量，把优秀的原创文章和译文分享给读者，为“快餐”添加一些“营养”元素。

快速链接

- [网站使用指南](#) »
- [问题反馈与求助](#) »
- [加入我们](#) »
- [网站积分规则](#) »
- [网站声望规则](#) »

关注我们

新浪微博：[@伯乐在线官方微博](#)

RSS：[订阅地址](#)

推荐微信号



合作联系

Email：[bd@jobbole.com](mailto:bd@jobbole.com)  
QQ：2302462408（加好友请注明来意）

更多频道

- [小组](#) – 好的话题、有启发的回复、值得信赖的圈子
- [头条](#) – 分享和发现有价值的内容与观点
- [相亲](#) – 为IT单身男女服务的征婚传播平台
- [资源](#) – 优秀的工具资源导航
- [翻译](#) – 翻译传播优秀的外文文章
- [文章](#) – 国内外的精选文章
- [设计](#) – UI,网页，交互和用户体验
- [iOS](#) – 专注iOS技术分享
- [安卓](#) – 专注Android技术分享
- [前端](#) – JavaScript, HTML5, CSS
- [Java](#) – 专注Java技术分享
- [Python](#) – 专注Python技术分享