

An Emacs Tutorial for Vim User

w0mTea

December 3, 2014

Contents

1	前言	2
2	为什么离开 vim 改用 emacs	2
3	emacs 的安装	2
4	基础知识	3
4.1	快捷键的约定	3
4.2	常用快捷键	3
5	简单配置	3
6	过渡——evil-mode	4
6.1	安装	4
6.2	启用	6
7	emacs 中的包管理	6
7.1	el-get	6
7.1.1	安装	6
7.1.2	更新	7
7.1.3	删除	7
7.1.4	recipe 文件	7
7.1.5	本节参考资料	7
7.2	ELPA	7
7.2.1	简单配置	8
7.2.2	安装/删除/升级	8
7.2.3	本节参考资料	8
7.3	手动安装	8
8	保护你的手指	9
8.1	用手掌外缘按 ctrl	9
8.2	改键	9
8.3	踏板	10

9 编程语言配置	10
9.1 C/C++	10
9.1.1 缩进	10
9.1.2 各种扩展	10
9.2 Python	10
9.3 Lisp	10
9.4 Perl	10
9.5 Ruby	10
9.6 Java	10
10 重量级应用——org-mode	11
10.1 安装	11
10.2 文档结构	11
10.2.1 标题	11
10.2.2 折叠循环	11
10.2.3 标题的结构化操作	12
10.3 列表	13
10.4 表格	13
10.5 链接	13
10.6 To-Do	13
10.7 标签	13
10.8 时间和日期	13
10.9 日程	13
10.10 导出	13
10.11 参考资料	13
11 文档和资料	13
12 结尾	13

1 前言

写这篇教程的起因在于向许多 `vim` 推荐 `org-mode` 时，他们总是觉得虽然 `org-mode` 功能强大，可是使用 `emacs` 总是有着一些障碍。作为一个同样从 `vim` 转向 `emacs` 的人，我觉得或许分享我的经验可以让他们更快的接受 `emacs`，从而体验 `emacs` 的美妙。

本文默认读者所使用的是类 `unix` 操作系统，因此对于 `windows` 用户来说，如果某些操作在你的电脑上无法执行，请不必大惊小怪。

2 为什么离开 `vim` 改用 `emacs`

在绝大多数 `unix/linux` 教程都教我们用 `vi/vim` 的情况下，许多人从开始就习惯了 `vim` 的一切：它的简洁、高效，它的模式，它简单直接的按键绑定……于是就这样，一批批的 `vim` 用户诞生。然而对于一直和 `vim` 并称的 `emacs`，我们往往只是知道它很强，却不清楚它到底强在哪里。如果你想清楚的了解这个 `vim` 的老对手到底凭什么能和 `vim` 分庭抗礼，那么，你可以试着使用它。

如果你是一个 `lisp` 爱好者，那么你绝对不能错过 `emacs`。或许经过漫长的配置，`vim` 同样可以很好的支持 `lisp`，可是在 `emacs` 上，`lisp` 天然就可以被良好的支持。同时，`emacs` 也可以被 `lisp` 扩展。因此，你写的每一行 `lisp` 代码，或许都可以让你的 `emacs` 变的更好用。写 `lisp`，也是让我离开 `vim` 转向 `emacs` 的契机。

如果你厌倦了 `unix` 哲学，厌倦了做一件事只能用无数的小工具来组合；或者你喜欢一个大而全的东西，可以帮你做各种各样的事情，那么 `emacs` 无疑是一个更好的选择。事实上，往往 `emacs` 用的越久，每天对着它的时间也越长，它帮你做的事情也越超出简单的文本编辑。从开始的写文档写代码，到收邮件玩游戏写幻灯片，甚至到煮咖啡，`emacs` 能帮我们做越来越多的事情。

3 `emacs` 的安装

`emacs` 并不像 `vim/vi` 那样几乎被所有类 `unix` 系统内置，因此我们往往需要手动安装 `emacs`。

对于有包管理器的系统，使用包管理器通常都可以成功安装 `emacs`。需要注意的是，某些发行版的仓库默认不安装 `emacs` 的 `GUI` 版本，因此需要手动安装 `emacs-x11` 或类似名字的包。

需要注意的是，`emacs` 的 `GUI` 版本并不是 `xemacs`。我们使用的 `emacs` 实际上是 `gnu emacs` 这个分支，而 `xemacs` 则是另一个 `emacs` 的分支。虽然 `xemacs` 和 `gnu emacs` 有着相当的兼容性，可是在某些时候难免会碰到奇怪的问题。

对于没有包管理器或仓库没有 `emacs` 的系统，可以从 <http://ftp.gnu.org/gnu/emacs/> 下载。

4 基础知识

毫无疑问，对于一个从未是用过 vi/vim 的人来说，使用 **emacs** 并不是一个特别让人困惑的事，最起码开始不是，因为 **emacs** 和他之前用过的任何文本编辑器（比如 windows 下著名的 **notepad**）从表面上看并没有太大的不同。然而对于习惯了 vi/vim 模式操作的人来说，这样的操作模式真是让人无比烦恼：如果不使用一系列快捷键，效率就会变的无比低下；可是如果使用快捷键……那都是什么鬼东西！

为了让我们可以更快的接受 **emacs**，我们需要了解一些最最基础的东西。这些知识并不繁琐，但却总是有很大的帮助

4.1 快捷键的约定

由于没有 vim 那样的模式之分，**emacs** 的快捷键总是需要使用组合键。可是网上查到的 C-n, C-x C-s 都是啥意思呢？

emacs 中，快捷键的表示都遵循了一些固有的约定。C-x 表示同时按着 ctrl 和 x，C-x C-s 表示先按 ctrl+x，然后按 ctrl+s。当然，也可以按着 ctrl 不放，然后依次按 x 和 s 咯。

同样的，还有 M-系列的快捷键和 S-系列的快捷键。M-x 表示同时按 alt 和 x（alt 在不同键盘上可能不同，有可能也叫 meta 之类的）。S-x 表示同时按 shift 和 x。在这样的约定里，还有一些其他的特殊键，比如 ESC、RET（回车）等

4.2 常用快捷键

这里列举一些最简单同时也最最常用的快捷键。

快捷键	功能
C-x C-f	打开某文件
C-g	取消正在输入的命令
C-x C-c	关闭 emacs
C-x C-s	保存当前文件
M-x	运行命令

5 简单配置

emacs 采用 **emacs lisp** 作为配置语言，因此在配置里看到大堆的括号请不要惊讶。通常来说 **emacs** 的配置文件以及各种插件都以 .el 为后缀名。

emacs 的启动文件（类似于 vim 的 .vimrc）可以是下列三个中的一个：

- ~/.emacs
- ~/.emacs.el
- ~/.emacs.d/init.el

虽然说使用哪一个配置文件都可以，可是我还是建议使用最后一种。因为这种方案下，你可以把 **emacs** 相关的所有配置都放在 **.emacs.d** 这个文件夹下，而不是零散的东堆西散。尤其在你的配置文件变的很大的时候，你可以轻松的把启动文件中乱七八糟的配置代码拆分成单独的模块，每个模块单独占有一个文件，并且统一放在这个文件夹里。

另外，下文提到的包管理中，最好也把其相关文件放在 **.emacs.d** 文件夹下。

至于具体的配置，可以根据自己的需求来弄。后文会提供一些好的站点帮助大家完成自己的配置文档。而一些简单的配置，会在后面的内容里提到。

6 过渡——evil-mode

对于刚接触 **emacs** 的 **vimer** 来说，最难习惯的估计就是光标移动了。如果还能像 **vim** 那样操作无疑会愉快的多。而像 **vim** 一样操作 **emacs** 并不是你一个人的想法，因此早就有别的高手实现了这一功能，那就是 **evil-mode**。

注：在下文中，配置代码在 **pdf** 中有可能无法复制，如果不想手打可以参考 **emacsWiki**: <http://www.emacswiki.org/emacs/el-get>

6.1 安装

emacs 有着若干种安装扩展的方法，具体的会在下一节讲到。这里只讲一种我最常用到的也是感觉最方便的方法：**el-get** 安装。

在你的配置文件中加入下列部分：（需要注意的是，**el-get** 的默认位置也在 **.emacs.d** 文件夹内）

```
(add-to-list 'load-path "~/emacs.d/el-get/el-get")

(unless (require 'el-get nil t)
  (url-retrieve
   "https://github.com/dimitri/el-get/raw/master/el-get-
install.el"
   (lambda (s)
     (end-of-buffer)
     (eval-print-last-sexp))))

;; now either el-
get is `require'd already, or have been `load'ed by the
;; el-get installer.

;; now set our own packages
(setq
 my:el-get-packages
 '(el-get                                ; el-get is self-
 hosting
 switch-window                          ; takes over C-x o
```

```

    auto-
complete                ; complete as you type with overlays
zencoding-mode          ; http://
www.emacswiki.org/emacs/ZenCoding
color-theme             ; nice looking emacs
color-theme-tango))    ; check out color-
theme-solarized

;
;; Some recipes require extra tools to be installed
;;
;; Note: el-get-
install requires git, so we know we have at least that.
;;
(when (el-get-executable-find "cvs")
  (add-to-list 'my:el-get-packages 'emacs-goodies-
el)) ; the debian addons for emacs

(when (el-get-executable-find "svn")
  (loop for p in '(psvn                ; M-x svn-status
yasnipet                ; powerful snippet mode
)
do (add-to-list 'my:el-get-packages p)))

(setq my:el-get-packages
  (append my:el-get-packages
    (mapcar #'el-get-source-name el-get-sources)))

;; install new packages and init already installed packages
(el-get 'sync my:el-get-packages)

```

上述代码段会自动检查是否安装了 **el-get**，并自动在未安装的情况下安装。注意，这段代码需要系统中安装过 **git** 才能运行。同时为了在安装其他扩展时不会出问题，建议安装 **svn** 或 **cvs**。把上述代码段保存后，重新运行 **emacs**，就会自动安装 **el-get**

el-get 安好了，那么怎么安装 **evil-mode** 呢？回到上面那段代码，可以看到

```

;; now set our own packages
(setq
my:el-get-packages
'(el-get
switch-window
auto-complete
zencoding-mode
color-theme
color-theme-tango))

```

只要在这段代码内添加上我们想要的扩展，而且这个扩展恰好在 `el-get` 的仓库内，那么我们就可以自动的安装并启用对应扩展。大多数常见扩展都可以被 `el-get` 自动找到，`evil-mode` 也不例外。因此只要在这段代码中加上 `evil-mode` 就可以。搞定后和下面的差不多：

```
(setq
 my:el-get-packages
 '(el-get
   switch-window
   auto-complete
   evil-mode
   zencoding-mode
   color-theme
   color-theme-tango))
```

之后重启 `emacs`，就可以安装了。

6.2 启用

安装成功后，只需要在配置文件中加入

```
(require 'evil)
(evil-mode 1)
```

就可以全局启用 `evil-mode`。如果想手动启动 `evil-mode`，把上面的 `1` 改成 `0`，在需要启动的时候按 `M-x evil-mode RET` 即可。

现在，`vim` 熟悉的操作，不就回来了吗？

7 emacs 中的包管理

在上一章，我们已经使用了 `el-get` 来安装扩展。只需要在列表中加入你需要的包名就可以自动安装，岂不是爽的很？这一节会介绍一些 `el-get` 的其他用法。

不幸的是，并不是所有的软件包都可以通过 `el-get` 安装，因此还需要介绍一些其他方法来弥补这一小小的缺陷。

7.1 el-get

7.1.1 安装

除了之前提到的配置文件中加入包名的方法。除此之外还可以在 `emacs` 中实时安装扩展。

M-x el-get-install RET 并在出现的 **Package install** 中输入想要的包名即可安装。注意：打包名时要善用 `tab` 补全哦，不仅可以省事，还可以检查是否打错以及该包是否在 `el-get` 的仓库内。

el-get 安装的扩展包会被记录在一个文件中，无论通过哪一种方式安装扩展，所以是否加入包名到配置文件并不会影响使用。但是我仍建议仍加入到配置文件中的包列表中去，因为那样在其他环境需要安装时，你只需要复制你的配置并运行 **emacs** 即可安装所有之前安装过的插件。但若是实时安装的插件没有加入配置，在更换环境时会遗漏一些东西。

注：在更换环境时把整个 **.emacs.d** 文件夹拷贝过去也可以避免遗漏实时安装的插件。

7.1.2 更新

M-x el-get-self-update RET 即可更新 el-get

M-x el-get-update RET 再输入包名即可更新选定包。

M-x el-get-update-all RET 即可更新安装记录中所有已安装的包。注，网速慢慎用，可能会被更新信息刷屏好久

7.1.3 删除

删除包列表中要删除的包名，使用 **M-x el-get-remove RET** 再输入包名即可。

7.1.4 recipe 文件

el-get 使用一系列的 **recipe** 文件来处理安装包。每一个 **recipe** 文件都描述了安装包的名字、下载地址、版本、安装后的初始化动作等信息。这些 **recipe** 文件就相当于包管理器的软件源元数据，我们查询、安装等操作都需要用到它。

默认情况下，**recipe** 文件放在 **.emacs.d/el-get/el-get/recipes** 文件夹下。

对于 el-get 默认没有的扩展，一个安装方法便是自己写一个简单的 **recipe** 文件。具体的做法可以参考 **emacsWiki el-get** 页。

另外，对于发布在 **emacsWiki** 上的插件，可以使用 **M-x el-get-emacswiki-refresh** 来获取/刷新其 **recipe** 文件。因此如果要安装的包列在了 **emacsWiki** 上，那么就不用自己麻烦的去安装啦。

7.1.5 本节参考资料

本节仅列出了少数用法，更多用法请参考下列网站：

- EmacsWiki: <http://www.emacswiki.org/emacs/el-get>
- Github: <https://github.com/dimitri/el-get/>

7.2 ELPA

ELPA 也是一个 **emacs** 的包管理工具，在 **emacs24** 及以上版本已经默认集成 (**package.el**)。

7.2.1 简单配置

ELPA 需要添加一些仓库源，如下：

```
(require 'package)
(setq package-archives '(("gnu" . "http://elpa.gnu.org/
packages/")
  ("marmalade" . "http://marmalade-repo.org/packages/")
  ("melpa" . "http://melpa.milkbox.net/packages/")))
```

这样就会添加 **gnu** 官方源、**marmalade**、**melpa** 三个源。

在某些情况下，启动 **emacs** 时会显示 **package.el** 没有被初始化，可以通过加入下列代码在启动文件的非末尾位置解决：

```
(setq package-enable-at-startup nil)
(package-initialize)
```

7.2.2 安装/删除/升级

1. **M-x list-packages RET** 会列举所有包。通过 **C-s xxx** 可以快速找到你想安装的包
2. 光标移动到对应包名上，按：
 - **RET**：会显示包的介绍
 - **i**：标记该包为待安装
 - **u**：取消标记
 - **d**：标记为待删除
 - **U**：标记为待升级。只有可升级的包才可被标记
 - **x**：执行，会删除 **d** 标记的包，安装 **i** 标记的包
 - **r**：刷新列表
 - **q**：退出列表

7.2.3 本节参考资料

- EmacsWiki: <http://www.emacswiki.org/emacs/ELPA>
- ergoEmacs: http://ergoemacs.org/emacs/emacs_package_system.html

7.3 手动安装

有时候，会有一些冷门的包或者自己写的包无法在前面讲过的仓库里找到，而你也不想写 **el-get** 的 **recipe** 或者 **elpa** 的本地仓库，那么就会用到本节的知识。

首先，你需要让 **emacs** 可以找到你的扩展文件。假如你的文件在 **~/Documents/emacs-package** 目录下，那么在配置中加入：

```
(add-to-list 'load-path "~/Documents/emacs-package")  
(load "you-package") ;.el 后缀最好省略
```

这样就可以启用你的扩展了。

由于手动安装有着诸多不便，而且使用也较为少，因此这里仅列出最基本的用法。更复杂的用法，请见：

- ergoEmacs: http://ergoemacs.org/emacs/emacs_installing_packages.html

8 保护你的手指

emacs 需要大量使用 **ctrl** 和 **alt** 两个键，但在大部分 **qwerty** 键盘上，**ctrl** 的位置都在很难按到的角落里。据说，如果长期使用小指按角落的 **ctrl** 会很容易导致手部健康出现问题。因此，我专门加入了这一章来列举一些常用的方法来避免 **ctrl** 和 **alt** 位置不当带来的伤害。

8.1 用手掌外缘按 **ctrl**

由于许多键盘中 **ctrl** 处于左下角，所以可以把左手外翻向左下角压去，这样就可以按到 **ctrl**。

- 优点：简单，不需要特别的准备
- 缺点：笔记本键盘很难用，按着 **ctrl** 时左手几乎无法按其他任何键
- 推荐度：2/5

8.2 改键

仔细观察，不难发现我们的键盘上总有一些位置很黄金却很少用到的键，这之中典型的例子之一 **caps lock** 键。因此，我们不妨更改键位设置，把使用频度更高的键更换到这些位置上。

一些常用的改键方案包括：

1. 左 **ctrl** 和 **caps lock** 交换：似乎是网上流传最广的改键方法
2. 右 **alt** 和右 **ctrl** 交换：这种改法最适合空格两边都是 **alt** 的键盘，这样大拇指稍稍移动就可以按到 **ctrl** 和 **alt**

改键方法视具体环境不同而有所不同。在 **windows** 下，可以使用各种改键软件完成这一工作。在 ***nix** 下，对于使用 **xorg** 的用户来说也可以使用 **xmodmap**。如果使用 **DE**，那么很有可能在设置中心内也有调整键盘布局的选项。

- 优点：效果不错，可以根据自己的情况自由配置
- 缺点：需要自己进行一些准备；偶尔使用被改的冷门键可能会不方便
- 推荐度：4/5

8.3 踏板

有一类被称为踏板的神奇道具，可以定义踩下时发出的按键信号，这类踏板用于 **emacs** 那真真是极好的，可以极大的减轻手部负担。

- 优点：简单方便，效果超群，直接减少手的工作量
- 缺点：相比上面的方法来说最贵；不同系统驱动可能有潜在问题
- 推荐度：3/5

9 编程语言配置

几乎可以肯定，**emacs** 用户中程序员占了大多数，因此没有相应配置怎么能行？本章会列举一些常用的语言的用到的常用扩展以及简单配置，更具体的配置请参照 **emacsWiki** 上仔细进行。

9.1 C/C++

写 **linux** 内核那帮家伙有不少都是用 **emacs** 的，**emacs** 是 **c** 写的，**GNU** 的许多东西是 **c** 写的，所以 **emacs** 默认就可以不错的支持 **C**，而 **C/C++** 的相关插件堪称多如牛毛……这里仅列出一些常用插件，大家可以一一尝试，选择最复合自己习惯的。

9.1.1 缩进

写代码，缩进搞不好，心情绝对好不了，这里就写写 **emacs** 中的 **c** 的缩进。

9.1.2 各种扩展

9.2 Python

9.3 Lisp

9.4 Perl

9.5 Ruby

9.6 Java

事实证明，如果需要写 **java**，请出门左转 **eclipse** 或出门右转 **intellij**……**emacs** 写 **java** 并不是一个很好的选择，而且相关插件开发的人也是寥寥无几，毕竟把 **emacs** 弄到 **eclipse** 或 **intellij** 那样好用绝对是一个艰难的事，所以广大的 **emacs** 都机智的选择了上述两种 **IDE** 之一。

10 重量级应用——org-mode

org-mode 是啥呢？它是一个很强大的东西，可以快速高效的通过纯文本文件来完成做笔记、TO-DO list、项目计划等一系列事情。它有些类似 **vim** 下的 **vimwiki** 和 **markdown**，但比前两者强大的多。

用 **org-mode** 写文档类的东西，你需要关注的只有你的文档的结构和内容，而其他的，都有 **org-mode** 帮你搞定，毫无疑问，这是一种相当爽的感觉。而且 **org-mode** 写完后，可以轻松的导出成各种格式：**html**、**markdown**、**pdf**、**odt**（进而通过 **libreoffice** 等转换成 **MS Office** 格式）、**beamer** 等等。或许你已经猜到了，你正在看的这个 **pdf**，也是 **org-mode** 生成的。相信如果没有 **org-mode**，我是不会有勇气开始写这个文档的。

由于 **org-mode** 的内容十分多，多到即使我再写一年也未必写的全的程度，所以我只能写一些基本的用法并给出一些参考资料，大家若有需要，可以查阅 **org-mode** 资料。

10.1 安装

Emacs23 之后已经默认包含有 **org-mode** 了，但是默认包含的版本往往比较低，因此建议安装一个新的 **org-mode**。安装方法如包管理部分所讲那样，可以用 **el-get** 安装，也可以用 **ELPA** 安装，也可以手动安装，这里就不多说了。

安装成功后，使用 **emacs** 打开任意一个后缀名为 **.org** 的文件就会自动启动 **org-mode** 了。

10.2 文档结构

10.2.1 标题

文档结构的骨架就是一级级标题组成的树状结构，这里就来说说标题。

org-mode 里标题的表示十分简单，***** 表示标题，几个 ***** 就是几级标题。需要注意的是，星号后需加一个空格。每次手打星号其实是很烦人的事，因此 **org-mode** 给出了一些快捷键帮我们搞定：

- **C-RET**：插入一个同级标题。也就是说，上一个标题是二级标题的话，这个快捷键也会插入二级标题哦
- **M-RET**：插入一个同级同类标题。这个和 **C-RET** 基本一样，一些细微的区别会在列表部分给出

10.2.2 折叠循环

当文档写长以后，就会有无从下手的感觉，可是通过良好的折叠，可以让我们仅面对少部分内容，并且可以清楚的看到文档的结构，为此 **org-mode** 提供了一系列折叠功能。

在一个标题上按 **TAB**，这个标题之下的内容都折叠起来，只留下标题。在标题上按 **TAB**，所执行的操作是一个循环，如图：



除了在标题上按 **TAB** 外，还可以使用 **S-TAB** 或者 **C-u TAB** 来完成全局折叠。多次按此快捷键也是循环操作，依次在总览、各级标题和所有内容间切换，如图：



这里仅列举的最基础的用法，更多用法请见参考资料。

10.2.3 标题的结构化操作

文档规模上去以后，经常需要以子树为单位在结构上进行一些操作，因此 **org-mode** 提供了一些结构化操作来帮助我们。

1. 标题间移动

- **C-c C-n**: 下一个标题
- **C-c C-p**: 上一个标题
- **C-c C-f**: 下一个同级标题。只会在同子树的标题间移动
- **C-c C-b**: 上一个同级标题。只会在同子树的标题间移动
- **C-c C-u**: 上一级标题
- **C-c C-j**: 保持当前位置不变的跳转。使用该指令后，会打开搜索，输入内容即可找到对应位置。浏览完后，按 **C-g** 就可以退出浏览状态，回到原位置

2.

10.3 列表

10.4 表格

10.5 链接

10.6 To-Do

10.7 标签

10.8 时间和日期

10.9 日程

10.10 导出

10.11 参考资料

- org-mode manual: <http://orgmode.org/manual/index.html>

11 文档和资料

- emacs manual: http://www.gnu.org/software/emacs/manual/html_node/emacs/index.html
- emacs wiki: <http://www.emacswiki.org/emacs/>

12 结尾

本文仓促写成，错漏颇多，还望各位指出错误，让这份教程可以帮助更多的人。