简介

成为高手很容易.当初我在 Vi 阵营,偶尔用 Emacs 还忘记"退出"的快捷键,一年后我跨入高手行列.

很多文章强调 Emacs 有多牛,但关于"如何做"则语焉不详.即使涉及到"如何做",谈细节多而方法论少,所以本文就 **侧重方法论**.

全文结构如下:

- 为什么 Emacs 值得学习,如对开源文化熟悉可跳过这一章
- 态度很重要
- 本文最核心观点,要充分利用高手成果,不要重新发明轮子
- 尽快掌握 Emacs 的步骤
- 进一步提高的提示(社区,阅读,知识管理)
- 跳出具体 Emacs 技巧,重要的是人
- 答疑和小结

为什么用 Emacs(可选)

简单谈谈,因重点是"怎么做",不是"为什么".

真正精通后 Emacs,其他编辑器自然精通

一旦尝过最好的,你自然了解好的编辑器应该有哪些功能.

比如一个内嵌的插件管理器是最基本的.

如下载了第三方插件,如果发觉其有问题,可以在不碰该插件原始代码的情况下修复.

插件服务器关闭了,应可以在自带的 U 盘上快速建立镜像.

目前流行的编辑器如 Sublime Text 还做不到以上几点.

Emacs 的特点决定了其社区水准不低

Emacs 用 Lisp 开发,Lisp 不同寻常的语法决定了其开发者都是资深开发者,掌握了多门语言.

Lisp 并不能给你带来任何好处,所以其社区成员都是纯粹的技术爱好者,投机取巧的功利主义者对其没有兴趣.

和 IDE 比较,Emacs 做的更多更快

IDE 针对特定语言或框架优化,而 Emacs 完成通用任务更有效.

例如,我碰到难题,需要上 IRC 请教国外高手(工作流是,粘贴代码到 http://gist.github.com, 在 irc 提问,看网页,将解决方案粘贴回来),Emacs 集成了 IRC 工具和浏览器 (例如),操作就很方便.

我使用 Visual Studio 多年,Firefox 也是高手.相信我,大多数情况下 Emacs 更快.

口说无凭,请看高手操作的 youtube 视频, Emacs Power: Can your editor do THIS!

顺便说一下,很多初学者关心的代码自动完成,主流语言支持都不错(包括 **Java**).

Emacs 会永存

个人开发者会丧失兴趣,公司会倒闭.但自由软件基金会将一直存在下去.

Emacs 作为其招牌软件也会维护下去,我的投资永不会贬值.

可以立刻开始工作.

软件开源,配置是纯文本,且资源消耗小,安装包很小 (命令行版本 30M 左右),任何环境下都可用.

这在大项目中特别有益,例如,某项目需同时编辑 Perl, Java,C, Bash, SQL, 要编辑远程服务器上的代码,网速不快. Emacs 的优势就体现出来了.

一年指的是一年中的空闲时间

我没说一年内须什么事都不干专学 Emacs, 我最反对没有短期回报的悬梁刺股.

我利用一年中通勤时间就取得了很大进步,自信到可以写下本文.

态度决定一切

理解软件自由

何为软件自由没有比自由软件基金会更权威了.我建议把 https://www.gnu.org/philosophy/free-sw.zh-cn.html 反复读,理解何为四大自由.

一旦真正理解了软件自由,Emacs 就变得非常简单了.

例如,很多用户习惯让 Emacs 启动时自动从其官方插件仓库 https://elpa.gnu.org 下载安装插件.当改网站偶尔下线或者公司的防火墙拦截了对外网站访问时,Emacs 就会启动失败.

这也就是一分钟可以解决的小事,如果你理解软件自由, **有勇气** 到 `~/.emacs.d/elpa/` 目录下看一看的话。

我不明白为什么年年会有那么多人对此长篇大论的讨论.

一个插件仓库(repository)本质上就是一个文件夹,它有一个含有插件列表名为 `archive-contents` 的文本文件,以及一系列插件包.你完全可以把这些文件下载下来,在本地硬盘里建立 ELPA 的镜像.

对个人来说,安装我写的插件 elpa-mirror 每年备份一下所有插件就足够了.

避免门户之见

所谓门户之见就是贴标签."我们的"对应"他们的"."熟悉的"对应"陌生的","正统的"对应"异端"的.

"我们的","熟悉的","正统的",就是"好的";"他们的","陌生的","异端的"就是"坏的".

比如用了 Emacs 就排斥 Vim 的快捷键,或者反之.

避免门户之见的关键就是意识到标签只存在于你的主观想像中.真实世界不会因为你的想像而扭曲.

以 Emacs 和 Vim 的快捷键为例,两种快捷键完全可以无缝接合.

当然思想的问题不是我空谈能解决的,关键是要实干.一个很好的治愈方法就是把 http://planet.emacsen.org/ 上约 4000 篇文章通读一遍.大约需要 8 个小时左右.泛读就可以了.目的就是了解世界有多大.

以科学理性做指导

之前有读者反映我的方法类似于大学里写论文做研究,事实上这正是我的灵感来源.

Emacs 只是一种特定领域得科学技术,其学习方法和其他学科是通用得.

打好基础,让自己的知识面有 **足够的** 广度和 **适当的** 深度,对新手是最重要的.否则会在一些琐碎问题上浪费时间.

比如新手的错误就是花大量的时间记快捷键,事实上网上教程列出的初学者"必知"快捷键都不是必需的.

具体步骤

开始前,解释一下后文用到的命名惯例,

- `C` 表示按下 Ctrl 键, `M` 表示按下 Alt 键
- `M-x my-command` 表示同时按下 Alt 和 X, 输入"my-command",然后回车

无 Linux/Unix 经验新手的快速指南(可选)

建议,

- 安装 Emacs 24
- 不安装任何第三方插件
- 掌握基本知识,什么是环境变量(比如 PATH, HOME 之类的变量),什么是 stdin, stdout, pipe
- 读官方教程,学会基本的文本操作(大概十几个快捷键)
- 使用 Emacs 24 自带的 org-mode 作个人管理
- org-mode 关键是用起来,只要记住按 TAB 键是展开内容就可以了,其他都不用学

这一步的目的是知道 Emacs 如何和其他软件交互,是必需的.

例如用 Emacs 开发 C++最简单成熟的方案是使用 GNU Global.配置 Global 必需要知道设置环境变量 `GTAGSLIBPATH` .如果你连环境变量是什么都不知道,那么用 Emacs 开发 C++也无从谈起.

尽可能多的掌握其他 **Linux** 知识是很有用的,即使你只在 Windows 下使用 Emacs.

读官方教程

按以下步骤阅读教程:

- 不安装任何插件打开 Emacs, 比如在 Shell 中运行命令 `emacs -nw -Q`

- 同时按下 Alt 和 X 健,输入 help-with-tutorial(类似快捷键后文简写为`M-x help-with-tutorial`代替),回车.

仅需半小时.关于 Emacs 多难学的谬论可以休矣.半小时的代价微不足道.想想你去练了多少个半小时吧.

即使你不打算使用 Emacs 默认的快捷键,这步也是必须的,不要跳过!

最起码要知道以下命令,

- `M-x describe-variable`, 快捷键 `C-h v`, 查看变量的文档
- `M-x describe-function`, 快捷键 `C-h f`, 查看命令的文档
- `M-x describe-key`, 快捷键 `C-h k`, 查看快捷键的文档

以实际问题作为切入点

微小的努力如能得到巨大回报,你会越学越有乐趣,进入一个感情上的正反馈.

在任何领域要成为高手,兴趣是最重要的.

以我为例,我急需 GTD 的工具,而 Emacs 的 Org-mode 是同类软件中最好的(没有之一). 用 Org-mode 大大节省了时间后,我对 Emacs 爱屋及乌,兴趣高涨了 100 倍.

反面例子是很多人以啃 Lisp 教程开始他们的 Emacs 之旅,坚持下来的人寥寥无几.

待解决的问题设定优先度

关键在于理性地考虑你最迫切需要解决的一个问题.

以这个问题作为出发点,除此之外都可以妥协.

虽然 Emacs 无所不能,但是饭也要一口一口吃.有时候退一步进两步.

例如,我一直以为 Emacs 的中文显示很完美,所以搞不懂为什么有人会在字体配置上花那么多时间.在陆续接到反馈后,我才明白原来是因为我一直在终端下使用 Emacs,终端软件可以完美显示中文字体,所以就没 Emacs 什么事了.需要配置字体的人用的是图形界面 Emacs.

当初只在终端下使用 Emacs 是因为需连接到远程服务器.我认为这是重点.甚至为此放弃了漂亮的配色主题(后来发觉此牺牲毫无必要).

塞翁失马,由此也避免了图形界面版本的所有问题.

站在巨人的肩膀上

这方面我是个负面榜样.刚开始抱着玩的心态,到处找有趣的配置粘贴到我的配置中去.

这是浪费时间!

我应一开始就照抄世界级大师 Steve Purcell 的 Emacs 配置.

警告,Purcell 总爱试用最新的 Web 开发的新技术,对他而言稳定性不是第一位的,如果你有热情和能力,愿意一起折腾,那么水平会提高很快.

这个如果是很重要的前提,当我上了 Purcell 的船时,我已有 10 年开发经验,精通多种语言.

如你不愿折腾,那至少不要重复我的错误,不要质疑,不要创新,跟着高手做.直说了把,你是初学者,开始阶段应以模仿为主.这点怎么强调也不过分!

为了加深印象,让我再举一例.有人向我反映,Emacs 快捷键太多,背起来压力很大.我的建议是,拿高手配置来用,而不是强加给自己背快捷键这样无聊的任务.你会发觉高手已安装了名为 smex 的插件,使直接输入命令比快捷键还快.

如果你还未信服,请再考虑一下我的理由:

- 文章标题是 **一年成为高手**,不是一年入门.
- 高手是世界级别的高手,不是关起门来一个小圈子内的高手
- 我就是这么做的,你可以看看一年内我给他报了多少 bug
- 说到底是态度问题,如果你真下定决心,考虑到 Purcell 的天赋和勤奋,追赶他的最好办法只有加入他
- 要超越高手就必须了解其标杆在哪,你需要一年时间去模仿去学习
- 基于 Purcell 的配置给他报 bug(甚至是提交补丁),你就是考虑到了他未考虑到的问题,至少在这点就超过他了,日积月累就很可观了.

好吧,你现在信服了.但是你是否 **真正理解** 了?

比如你是否马上推论到:即使不用高手的配置,也可在 github 上订阅(watch)高手配置,其更新通知等价于免费的维护服务.

报 bug

像武侠小说那样拜高手为师是白日做梦.唯一能让高手指点的办法是先付出.最可靠的付出就是报 bug.

我就是这样学到一些高级 Lisp 技巧的.

不要有报 bug 低级的想法.很多高手都是乐于且善于报 bug.倒是菜鸟喜欢重新发明轮子.

帮助高手,你的起点就高,还有得到指点的好处.

持续改进

前提是起点高,要在高手已有工作上改善.即使是微小的改善,如果坚持一段时间,就是巨大的进步了,你就可以在这一点上笑傲江湖.

再找出另一高手需要改善的地方,使用同样的方法.

例如,默认在 Emacs 中移动子窗口焦点不是很方便.需按"C-x O"多次.我找到了 emacs 插件 switch-window,只要按"C-x O"一次,会有提示子窗口编号,接下来只要输入编号就可以了.但还有改善空间,我又找到了 window-number.el,只要按"M-NUM"就可以了.

window-number.el 已完美,但 Alt 键还是有点慢,我结合 evil 和 evil-leader,可以按逗号和数字飞速切换子窗口了.

加入社区更上一层楼

最重要的是专一.

例如,Quora.com 上有很多有趣的话题.请克制兴趣,不去定阅和 Emacs 无关的话题.

Reddit

Reddit 是最好的.优点是一直能访问.

Google Plus

Google Plus 贴子质量高.例如,我加入了 Linkedin 和 Facebook 的 Emacs 论坛,目前都退出了.不是它们不专业,只是 Google Plus 讨论技术层次较高.

目前人气不如 reddit,原因在于 Google.

GitHub 是 geek 云集的地方

GitHub 的版本控制服务很好.现在它的社区化倾向越来越强了,我喜欢.

例如,可以看一下 https://github.com/search?p=1&q=stars:>20+extension:el+language:elisp&ref=searchresults&type=Repositories 上最酷的 Emacs 插件.

Emacs 牛人的博客

最好的是 Planet Emacsen,多个 Emacs 博客的集合.

Quora.com

我偏爱的是"列举最有用的命令"之类的具体问题.很多回答大开眼界.即使我已精通 Emacs.

那种"如何入门"的问题,人人都能插上一脚.即使有高水平的回答,也淹没在众多平庸回答中.

如果你的问题就是比较泛泛而谈的,从一个能测量水准的具体问题入手找到高手,然后看高手是如何回答那些比较泛的问题的.

在 twitter 上以 "emacs :en" 定期搜索

twitter 人多,更新结果快.

之所以加上":en"是因为要排除日文内容,因我不懂日文.

如果你懂日文,则应充份利用日文资源,其质量相当高.

在 stackoverflow 上搜索相关讨论

google "emacs-related-keywords site:stackoverflow.com"

我会定期搜索,同一帖子反复精读.因为讨论质量很高.

http://emacs.stackexchange.com 是 Stackverflow 旗下专门的 Emacs 问答社区.

到 Youtube 上看 emacs 相关的视频

我就是看了 Google Tech Talks 上这个 Org-mode 作者的介绍 而爱上 org-mode.

不过 Youtube 搜索结果是最佳匹配的.由于相关视频并不多,如按照默认算法,每次总是那几个.所以如果关注最新进展,搜索应以时间排序.

读书最有效

EmacsWiki

EmacsWiki 是社区维护的文档,是最酷插件和最佳实践的集合点.

有人抱怨文档太乱,质量参差不齐.前者我有同感.后者不赞同.EmacsWiki 文档质量相当高,因其是 **唯一的** 半官方文档.忍受其乱中有序的现状吧.

最佳阅读方法是,选定一特定主题,从头读到尾.这样对最新进展都了解了.是否要采用其建议另当别论.

Emacs Lisp 书籍推荐(可选)

Bob Glickstein 的 Writing GNU Emacs Extensions 是最好的.

生动,例子丰富.作者用心安排了书的结构.例如,很早就介绍了 defadvice 的用法.defadvice 是 Emacs Lisp 的精华.

Xah Lee 提供付费 Lisp 教程也相当不错.

Steve Yegge 的 Emacs Lisp 教程

他的 Emergency Elisp 很简洁.我特别喜欢"Statements"一章.

知识管理

不要低估长期管理的累积效应.

正面例子参考 Steve Purcell 的配置. 2000 年开始维护!其声誉和质量不用我多费口舌.

知识积累的越多,这些知识之间的联系就会越多.联系增长的速度是以指数的方式增长的.如从头来过,意味着积累的知识的书面记录丢失了.损失是很大的.基数已归零,增长的量又能有多少.

所以决不要重置配置!

这也是后文谈到为什么要用工具保存配置和知识的原因.

配置纳入 github 的版本控制

我的配置见 https://github.com/redguardtoo/emacs.d.

版本控制可以认为是一个集中式的知识管理,任何时刻任何地点对配置的修改都要及时上传合并 (merge). 这是积累能力的关键.

共享实际也是一种利己行为,有很多人使用我的配置,等于帮我测试.

将相关资料 (如电子图书,博客文章) 备份

我将所有资讯都放在 dropbox 的服务器上,这样资料就同步到我的智能手机和我的平板电脑上,我可利用空闲时间学习.

请点击这里注册 dropbox 帐号.注意,dropbox 客户端完全可以在国内使用,虽然访问其首页可能有点问题.

我还写了许多博客文章.这些文章都存在 org 格式的文件中.最后发布的静态博客也纳入版本控制,参见 http://github.com/redguardtoo/redguardtoo.github.io.

第三方插件推荐

初学者的问题是装了太多插件,管理成了问题.

我建议的原则是少而精,被少数最优秀的插件培养出品味后,可自由挑选适合的.

标准如下:

- 高品质
- 常更新
- 很强大

所有插件都可通过包管理器下载.

以下是清单:

名称	说明	同类插件
Evil	将 Emacs 变为 Vim	没有
Org	org-mode,全能的笔记工具	没有
company-mode	自动完成输入,支持各种语言和后端	auto-complete
expand-region	快捷键选中文本,可将选择区域伸缩	没有
smex	让输入命令变得飞快	没有
yasnippet	强大的文本模板输入工具	没有
flymake	对不同语言做语法检查	flycheck
ivy or helm	自动完成,在其上有插件完成具体功能	ido
ido	和 helm 类似,helm 和 ido 可同时用	helm
js2-mode	javascript 的主模式,自带语法解释器	js-mode
w3m	网络浏览器(需安装命令行工具 w3m)	Eww
simple-httpd	Lisp 写的 Web 服务器	elnode
window-numbering.el	跳转到不同的子窗口	switch-window.el
web-mode	支持各种 HTML 文件	nxml-mode
magit	玩转 git	没有

名称	说明	同类插件
git-gutter.el	标记版本控制的 diff(支持 subversion)	没有

Emacs 是一种生活方式

牛人其他方面也很牛.举一反三你收获会很多.

Sacha Chua 就是这样一个有牛人气质的女孩,这是她的 Youtube 录像. 她学习的方式是 让 Emacs 自动将手册语音合成,这样她在房间里走来走去的时候也可以听文档了.

我现在有意识地整理高手名单,观察他们 **除了 Emacs 外** 用什么工具.

例如, js2-mode 的维护者 Masafumi Oyamada(网名 mooz)也开发了 keysnail 和 percol. 特别是 percol, 使我命令行效率提高了 10 倍.

这个阶段可称之为 **心中有剑,手中无剑**.

是否用 Emacs 不重要了,重要的是随心所欲.例如,很多人争论哪个编辑器自带的文件管理较好.我从 mooz 那学到大招后,就跳出五行外,不在三界中了.

付之于行动

如何行动因人而异.

关键是真正理解本文要点.

例如，你是否意识到之前的章节意味着以下行动:

- 找出所有插件的作者
- 在 Quora/Twitter/GitHub/Reddit/Google+上跟随他们
- 通读他们已发表的贴子

使用 Evil

Evil 是 EvilVim 模拟器.

如果你不熟悉 Vim,在命令行里运行 `vimtutor` 或者安装 Emacs 插件 evil-tutor 学习 Vim 基本命令.

该教程大概需要半小时.关于 Vim 的基本操作的讨论就到此为止了.网上关于 Vim 教程汗牛充栋,你可以自行阅读.

本文的重点是探讨如何结合 Emacs 和 Vim 获得完美文本编辑器,达到*神用编辑器之神*的境界.

Text Object

了解 Vim Text Object 的概念.

Evil 的强大之处就是你可以用 Emacs Lisp 来自定义"Text Object".自由的 Lisp 使得你完全不用理睬 Vim 中的"约定俗成".

比如在操作自定义的 Text Object 时,当前焦点完全可以在 Text Object 之外.这是 Lisp 写的寻找附近的文件路径或者 URL. 用 Vim Script 写个类似的脚本难很多.即使你用了 vim-textobj-user 之类的插件辅助开发也没用的.

而且 Lisp 代码完全可以调用*任何*的第三方插件或者 Emacs 的不计其数的 API.比如 Evil 中操作 Text Object 的过程中我完全可以显示对话框问用户问题,访问几个网站等等.

这些额外功能对 Vim 来说就是不可能完成的任务了.

Leader 键

Vim 自带 Leader 键的功能,你先按了 Leader 键(很多人定义为空格键)后,再按其他键(比如"kk")会触发你自定义的命令.本质就是给你更多的快捷键.

在 Emacs 中我们需要使用第三方插件如 evil-leader 来实现类似功能.

某些 Vim 用户不能迁移到 Evil 的原因就是自定义了太多使用 Ctrl 键的快捷键,和 Emacs 默认的快捷键有冲突.

这些用户没有意识到的是借鉴 Emacs 的思想,他们在 Vim 和 Emacs 的效率可以有巨大的提升. 我只提三点供参考:

第一,典型 Vim 的用户的问题是没有充份利用 Leader 快捷键.我看过大多数 Vim 高手在 GitHub 上的设置,他们一般定义*10 到 20 个左右*Leader 相关的快捷键.

我定义了*300 个*Leader 相关的快捷键.

典型 Evil 用户(如 spacemacs 用户)大概有 3000 到 10000 个 Leader 相关快捷键可用.

第二,Vim 用户的另一个问题是快捷键没有优化.最常用的快捷键应该最容易按.何为最常用快捷键必须来自*真实数据*.

这是我用 Emacs 的插件 keyfreq 测试六个月后得到的部份数据 (我的 Leader 键定义为逗号):

Times	Percentage	Command	Key
4967	12.00%	evilmi-jump-items	%
2892	6.99%	compile	, o o
2178	5.26%	find-file-in-project-by-selected	, k k
1953	4.72%	copy-to-x-clipboard	, a a
1566	3.78%	paste-from-x-clipboard	, z z
1227	2.96%	er/expand-region	, x x
897	2.17%	evil-repeat	.
866	2.09%	ido-find-file	, x f, C-x C-f
819	1.98%	toggle-full-window	, f f
815	1.97%	etags-select-find-tag-at-point	C-], , h t

Times	Percentage	Command	Key
721	1.74%	back-to-previous-buffer	, b b
682	1.65%	split-window-vertically	, x 2
539	1.30%	find-function	, h f, C-h C-f
494	1.19%	counsel-recentf-goto	, r r
397	0.96%	counsel-git-grep	, g g
376	0.91%	delete-other-windows	, x 1, C-x 1
372	0.90%	evilnc-comment-or-uncomment-lines	, c i
351	0.85%	eval-expression	, e e, M-:
326	0.79%	evilmi-select-items	, s i
320	0.77%	paredit-doublequote	
307	0.74%	evil-filepath-outer-text-object	
300	0.72%	steve-ido-choose-from-recentf	
295	0.71%	split-window-horizontally	, x 3
283	0.68%	git-add-current-file	, x v a
279	0.67%	winner-undo	, x u, , s u, C-x 4 u
278	0.67%	describe-function	, h d, C-h f
278	0.67%	evil-goto-mark-line	'
269	0.65%	ido-kill-buffer	, x k, C-x k
254	0.61%	evil-goto-definition	g d
253	0.61%	pop-tag-mark	M-*
251	0.61%	git-messenger:popup-message	, x v b, C-x v p
246	0.59%	my-goto-next-hunk	, n n
237	0.57%	evilnc-comment-operator	, ,
235	0.57%	flyspell-goto-next-error	, f e, C-,
214	0.52%	evil-exit-emacs-state	
212	0.51%	browse-kill-ring-forward	
210	0.51%	flyspell-buffer	, f b

第三, 由于 Lisp 的强大 Leader 键的使用在 Emacs 中有无限可能如果你使用 general.el 代替 evil-leader, 你可以以同时定义多个 Leader 键.可以在切换文件的时候切换 Leader 键等等.

Evil 和 Emacs 原生插件的兼容性

如果你真正理解了我前面的章节,这就根本不是问题.

之前我提到了要保持头脑开放,要尽可能抄高手的代码,积极的报 bug 等观点.现在让我演示一下如何应用.

我知道有很多人宣称,Evil 和 Emacs 的许多插件有快捷键冲突,重新配置很麻烦.

一开始我也相信了这些一派胡言,所以每装一个新的插件,都要辛辛苦苦再设置 evil 的快捷键.

有一天我问自己,Lisp 那么强大,Evil 那么优秀,也许有更方便的简洁方案?许多人说不行不一定是真理,只有实际调查过的人才有发言权.

我也没有自己钻研 Evil 的代码,取而代之的是给 Evil 的开发者 Frank Fischer 报了个 bug,他给我了一个完美的方案,根本不需要重设快捷键.

这是这个方案在 git-timemachine 中的完美应用.

Evil 专用的插件介绍

我就选择 MELPA 上最流行的 5 个插件简单介绍一下,类似的优秀插件还有很多.

要点不在于你装了多少插件,而在于理解由于 Lisp 的强大和 Emacs 的自由,这些插件比 Vim 对应的插件功能更多,更容易拓展.

evil-surround

对应 vim-surround.

我通常用 expand-region 选中一段文本,然后按 `S` 或者 `M-x evil-surround-region`,再按任意字符(比如双引号)就可以在文本首尾两端附加该字符.

当然它也支持修改删除操作.

之前提到的 text object 也完美支持.

懂 Lisp 的话可以修改 `evil-surround-operator-alist` 自己定制操作.

evil-nerd-commenter

对应 vim-nerd-commenter,这是我写的,功能更强大.

你可以 `M-x 5 evilnc-comment-or-uncomment-lines` 快速注释当前 5 行或者取消注释当前 5 行.

你也可以选中一个区域 `M-x evilnc-comment-or-uncomment-lines`

由于 Emacs 的强大,默认就支持所有世界上已知的语言,而核心代码也就是 1 行而已.Vim 插件对应的功能代码要有 400 行.

如果你在 org-mode 格式的单一文件中混杂多种语言的话,它也能智能识别.这个功能在 Vim 中基本不可能实现.

evil-matchit

对应 vim-matchit.又是我写的.自然功能更强大.

本质就是你当前焦点在文件的某个位置 A,你按 % 或者 `M-x evilmi-jump-items`,焦点移到位置 B,你再按同样的键,又回到了位置 A.

比如在一个 HTML 文件中,你就可以在 `<body>` 和 `</body>` 间跳来跳去.其他各种编程语言都支持.

Vim 对应的代码我读过,限制比较多,比如你一定要先定义一对正则表达式来匹配 A 和 B 的位置.这种限制在某些语言如 **Python** 中就会比较麻烦.

Emacs 的实现就完全体现了 Emacs 的自由精神,我建立了一个动态查询的矩阵,矩阵的元素就是函数对象而已.用户可以在运行时替换这些函数对象,所以怎么跳转,跳到哪都是完全自由的.

所以 python 的支持就毫无问题.想支持更多的语言或者对我的实现不满意,在 `.emacs` 中写几行 Lisp 代码就可以了.

evil-escape
按自定义快捷键退出当前的各种状态,相当于 Vim 中的 `ESC` 或者 Emacs 中的 `C-g`.

我定义自定义快捷键为 `kj`.如果你效率高的话,取消的默认快捷键就太慢了.

让我给你举个例子说明什么叫效率高.我移动手指去按 ESC 键需要 0.5 秒.

Sublime Text 默认的文本搜索要比我的 Emacs 设置慢 40 倍.如果 Sublime Text 搜索需要我等待 40 秒,那么节省按取消键的 0.5 秒就毫无意义.

我只需要 1 秒完成搜索,所以把取消操操作从 0.5 秒减少到 0.1 秒的感觉就完全不一样.

evil-visualstar
对应 vim-visual-star-search.

选择一段文本,按 # 或者 * 搜索.

在 Shell 和 Interactive Interpreter 中使用 Evil

可以 `M-x shell` 或者 `M-x term` 进入 Shell.

传统上大家都在 Shell 中用 Emacs 的默认快捷键.

不过你仔细计算过的话,会发现 Vim 的快捷键更有效率.

Shell 的作用无非就是运行命令或者脚本代码,然后输出结果.

当我们在 Emacs 中运行 Shell 的时候,命令和代码往往是从别的地方拷贝过来的.

粘贴命令和代码到 Shell 中,分析/过滤/搜索输出的结果,都是 Vim 的快捷键更方便.

我之前提到的所有关于 Evil 的技巧和插件都适用于此.

Interactive Interpreter 和 Shell 没有本质区别,无非就是解释器支持的语言不一样罢了.比如 inf-ruby 支持 Ruby.

你可以按 `C-z` 切换纯 Emacs 快捷键.我从不切换,因为我对这种杂交的快捷键非常满意.

Evil 的小结

对 Vim 用户来说,Evil 不仅提供了 Vim 的完美模拟,还开辟了用 Lisp 拓展 Vim 的新世界.

对 Emacs 用户来说,Evil 也不仅仅是提供了新的快捷键,而是提供了更多的可编程的数据结构和范式(如 text object).

关键是发挥你的创造力,自由地接合 Emacs 和 Vim 的长处,发明新技术和新技巧.这种机会目前是很多的,赶快行动起来吧.

答疑

菜鸟怎么开始

到 https://github.com/redguardtoo/emacs.d 参考"Install stable version in easiest way"一节.

只要点击下载两个 zip 文件就可以了,不需 Git 的任何知识.

Steve Purcell 的配置是否有文档可以参考?

除了 README 外没有,我主要是通过看 EmacsWiki 和源代码来了解.窍门是源代码文件的头部有使用指南和作者的联系方式.

高手的配置是否太重量级?

高手的配置都是轻量级的,因为他们知道如何优化.

比如有种叫 Autoload 的技术. 只有用到模块的某一功能时那个模块才会被载入内存. 我推荐的高手都知道这类技巧.

除了 Purcell 的配置,还有其他高手的设置吗?

我搜了下 github:

- Bozhidar Batsov's emacs.d
- Sylvain Benner's spacemacs (Spacemacs 是针对 Vim 用户优化的，所以非 Vim 用户不用试了)
- Eric Schulte's Emacs Starter Kit.

有没有更简单的配置?

可用 我的配置:

- 去掉了 Git 依赖.
- 网络不是必须的
- 安装了拼音输入法
- C++支持强大

注意,Purcell 作为顶尖 Web 开发者,会试用最新的 Web 技术,而我的配置 Web 类插件更新会滞后一段时间.另外我的工具链和 Purcell 不完全一致.你自己权衡了.

该使用 Emacs 的哪个版本

目前稳定版是 Emacs 24.3 或 24.4,建议不要用高于此版本的 Emacs.

通常不用担心版本问题.主流的 Linux 发行版会处理.

Vi 高手要转阵营吗?

嘿嘿,我也是 Vi 精通后转到 Emacs 的.就是因为 Emacs 的强大(例如和 gdb 的完美结合)以及其脚本语言是 Lisp.

当然 Vi 的多模式编辑和快捷键比 Emacs 要高效得多,所以最佳方案是 Vi+Emacs.

目前我用 Evil, 在 Emacs 下模拟 Vim,结合两者优点.

现在我是 **神用编辑器之神**!

警告,我默认启用了 Vim 的快捷键,不习惯可打开~/.emacs.d/init.el,将其中一行代码注释掉,细节参考 README.

为什么很多 Vim 高手不能接受 Evil?

因为他们对 Vim 快捷键做了深度配置.Emacs 默认要经常按 Ctrl 键,如自定义的 Vim 快捷键也用 Ctrl 键,难免有冲突.

解决办法是大家都使 Leader(Vim 直接支持,Emacs 需第三方插件).

还有一个办法是呆在 Vim 的舒适区里.如能忍受没有 org-mode 和 lisp 的生活,那么不会有问题.

如犹豫不决,请重读"态度决定一切"一节.

我一旦认识到 Evil 和 Evil-leader 的潜力,立刻把我 Vim 的设置按 Emacs 的重设了一遍。

更光辉灿烂的例子就是 spacemacs 的作者了,无数的 github 星星代表了他的成功.

不习惯默认快捷键,怎么办?

忍!

默认快捷键经过几十年考验相当高效,未成为高手前还是要忍.

如一定要在用 Windows 快捷键的,可考虑 ergoemacs.

快捷键太多记不住怎么办?

没必要记,我也只记常用的十几个快捷键.顺其自然,多用记住,不用就忘,很正常.

目前很多高手在用 Smex,可飞快输入命令,快捷键实际上不需要了.

使用牛人配置后,界面有些奇怪的 bug,怎么改?

不要改! 参考上文站在巨人的肩膀上一章,你觉得奇怪是因为缺乏经验,把某些特性误认为是 bug.请坚持至少一年.

例如,有人反映右边第 80 列处总有一竖线,希望能去掉.

实际上这是一特性,提醒用户一行宽度不要超过第 80 列. 这是 每行不要超过 80 列的原因.

我建议第一年应 **尽量理解而不妄加判断**.

已更新软件包,但是没有任何作用,也没有任何错误信息

删除 HOME 目录下的".emacs", "~/.emacs.d/init.el"就是取代原来的".emacs".

如有任何关于如何配置的问题

- 读官方教程
- 善用 google 和我提供的信息

例如,问：在 .emacs.d 中的 init.el 文件起什么作用？答：google "emacswiki init.el".

使用牛人配置后启动报错,如何解决?

先确认已装上了 **你需要的** 第三方命令行工具,这些工具是可选的,清单见我的 README.

如排除了以上原因,带上"–debug-init"参数重新启动,然后将错误信息及环境报告到对应的开发者.

报告时应给出细节.例如很多读者给我的 bug 都是由于第三方插件版本较新引起的,我拿到版本号后,才能下载特定版本已重现 bug.否则只能靠猜,来回邮件浪费很多时间.

牛人的配置太复杂,还是从一简单的配置改起好控制

那你就是走我后悔莫及的老路,一个人在黑暗中摸索.开头兴致很高,但现实是残酷的,碰到复杂问题解决不了.只能逃避,借口 Emacs 太复杂而放弃了.

我最终醒悟过来走上光明大道,很多走上岐路的人恐怕就没这个觉悟和毅力了.

希望自己掌控坦率地说是一个非技术问题,因为没有自信心,所以有补偿心态. 希望通过一种错误的方式来证明自己.结局无非是恶性循环.

正确地方法是放下身段至少一年 (我已反复强调这一点),打好基本功,读书,虚心向高手学习.

为什么我用了牛人配置后自己额外添加的插件无效

Emacs 是个开放平台,其众多插件发布前并不一定有严格的测试.所以插件之间可能有冲突.

这也是我为什么建议初学者直接使用牛人配置的原因,因为牛人已经解决了众多兼容性的问题,你只要直接享受他的服务就行了.

即使你发觉了牛人尚未来得及处理的 bug,最有效方法是提交报告给牛人,而不是自己去钻研 Lisp.

我想用 Windows 版本的 Emacs 而不是 Cygwin 版本,怎么做?

需对命令行操作熟悉.关键知识点有两个：

1. 设置 HOME 环境变量,因为 .emacs.d 中的某些 elisp 脚本假定 .emacs.d 在 HOME 所指定的路径中.

2. Emacs 的某些功能需要使用第三方的命令行工具,这些工具的路径应该添加至环境变量 PATH 中(可选,原因见后面).

如你不知道如何在 Windows 下添加修改环境变量,不知道如何安装第三方工具,建议还是先用 Cygwin 中的 Emacs,因它已自带工具,没有的话安装也方便.且在 Cygwin 下环境变量 HOME 默认已设.

第三方命令行工具清单请参考我的.emacs.d 中的 README.

Emacs 在代码跳转和自动完成上和商业 IDE 有差距,怎么办?

这个差距说到底是后端语法解析引擎的问题.通常这个问题都是以微软的 Visual Studio 和 IBM 的 Eclipse 作参照.

就 C++来说目前有用苹果公司的 clang 的方案,效果不错.具体用什么插件来调用这些引擎有很多选择,不展开了.

实战中,我通常就用 ctags 作为后端引擎,因其通吃所有语言. 虽然解析效果差一点,但是恰当的命名规范(尽量少重名)可以弥补.

如 ctags 不满意,可考虑用 Gnu Global (gtags).

以上讨论的都是后端引擎.

就前端界面来说,做的比较好的是 company-mode,维护很活跃,你可就特定语言如何配置咨询其开发者.

Java 和 C#语言的主力开发工具最好用 IDE 而不是 Emacs.C#又比 Java 更难在 Emacs 中使用.原因你懂的.

网页浏览

强烈建议用 Keysnail.

这是最佳的,我已试过 **所有** 可选项.

邮件

我用 Gnus. 但有很多其他方案.

如你必须访问 Microsoft Exchange Servers, 还要用 Davmail.

用了 Davmail 后, 还可以用 Popfile 来分捡邮件. Davmail+Popfile 让我生活在天堂.

为什么 Emacs 启动时从服务器 (elpa) 安装第三方软件包 (package) 会失败?

请启动 Emacs 后,运行 `M-x package-refresh-contents` 以从服务器更新软件索引,然后重启 Emacs 即可.

如果你没有用 Emacs24,没有完全拷贝高手的配置 (这是本文的中心思想),那么你需要安装 package.el,细节参考这里.

Emacs 下载软件包 (package) 是通过 http 方式,所以如果网络出问题的话你需要用 http 代理服务器,具体操作见后文.

有些网站 Emacs 访问不了

在命令行中启动 Emacs 时加上 "http_proxy=your-proxy-server-ip:port" 前缀.

例如,

```
http_proxy=http://127.0.0.1:8000 emacs -nw
```

有些软件包下载不下来,也不会用代理

那么就用我的 Emacs 配置.

和我的配置配套的是我建立的独立的第三方包服务,请参考其主页上的 README.

早点学习 Emacs Lisp 是否有助于成为 Emacs 高手?

否,只会起阻碍作用!

Lisp 语法和通常的语言不同,除非有相当编程经验(至少 10 年),一般人都会对其有一点负面情绪(当然是毫无道理的偏见!).学习任何新东西,长期来说兴趣最重要.一开始应避免任何负面情绪.

Emacs Lisp 又是只用于 Emacs 的语言,有大量术语需要掌握.如"Buffer","Yank","Font face",只有资深用户才能理解.

所以在软件使用没有相当基础前学习其拓展语言是浪费时间.

参考前文关于找到切入点的一节,我推荐的顺序是,先用优秀的配置享受到好处,有了兴趣后学习 Lisp 就水到渠成了.

有世界级高手(名字不点了)对我的建议不以为然,他说 Lisp 很强大很有趣,应该先学.

但是他的盲点在于,忘记了自己转向 Emacs 前在其他编辑器上已一览众山小了.他用 Python 拓展 Sublime 已熟到厌烦,Lisp 的奇特语法反而刺激了兴趣.编辑器的常用术语也不在话下. 而本文针对的是大多数的凡人.

选择适合自己的路,一年以后天才也好,凡人也好, **达到的高度都是一样的**.

掌握 Emacs Lisp 是否是成为高手的必要条件?

否.但 Lisp 是很强大的语言,特点是一切皆可改.当我说"一切"的时候,我就是指字面意义上的"一切",不是修辞上的夸张.

我用过许多编辑器,除了 Emacs 没有一个能做到"一切可改"这点 .vim 也不行.

所以学点 Lisp 对提高 Emacs 水平没坏处.另外 Lisp 语法不错,值得程序员一学.

顺便说一下,Lisp 很简单,比 VB 容易多了,一旦你适应其语法,就会发觉它其实蛮友好的,至少少打很多字.

有必要学习键盘宏(Keyboard Macros)吗?

没必要,Lisp 足够了.

但是键盘宏生成的 Lisp 代码有时候比较有趣,建议你精通 Lisp 后再来玩玩键盘宏.

基本操作我会了,下一步学什么迷茫中

关键是你打算用这把瑞士军刀做什么.

前文已强调过以兴趣和解决实际问题作为切入点.

举一些我自己的例子说明:

- 我有写博客需要,懒得用 Wordpress 那个破界面,所以用 org2blog
- 开发 Ruby on Rails 程序需要 IDE,装了 rinari
- 做跨平台 C++桌面开发,装了 cmake-mode
- 需在多个子窗口间跳来跳去,所以装了 window-numbering.el
- 大项目需同时调试多种语言,所以装了 evil-nerd-commenter,这样不用记特定语言的语法就可注释掉代码.

如何学习 org-mode?

Org-mode 简明手册 是不错的中文教程.

最好的英文教程是 Carsten Dominik(Org-mode 发明者)在 google tech talks 上的演讲.其要点为 org-mode 本质是一个文本文件,只要记住按 TAB 展开或者缩进条目就可以了.其他特性可慢慢学.

对于"一切都用 Emacs 来完成"的观点你怎么看?

不要走火入魔.Emacs 本质是个平台,提供了无限可能性.

从实用角度讲,Emacs 和其他工具结合有时能更快完成工作(不过在没有一年的修炼之前 **千万不要猜 Emacs 不能做什么**).

以下是 Emacs 不一定能吃独食的地方:

- 剪贴簿: 应结合命令行工具 xsel(Linux)/pbpaste(OSX)/putclip(Cygwin)
- Web 浏览: 用 Firefox 结合插件如 keysnail
- 远程登录管理: 用 screen/tmux
- FTP: 用专门的 FTP 软件
- 文件管理: 用专用软件
- Lisp 速度比较慢，如有大计算量的工作,交给第三方工具来作.

重点是头脑灵活,既坚信 Emacs 无所不能,也适当变通.

联系我

这是我的 Twitter 和 Google Plus 以及 微博,也可通过我 email<chenbin DOT sh AT GMAIL DOT COM>联系我.我在新浪 weibo.com 上开通账号 emacsguru.

主力博客为 http://blog.binchen.org.

我不回答具体配置的问题.如你通读本文,应知道哪里找答案更好.

结语

再强调一下本文最重要的观点:

- 以 **基于解决实际问题产生的兴趣引导**
- **完全照抄世界顶尖高手如 Steve Purcell 的配置**,尽量避免自己写 Lisp
- 给高手报 bug 就是最好的学习,
- 学习 Emacs 和 **学任何专业技能(拉小提琴,解数学题)的方法论都是一样的**,请参考一万小时天才理论.

关键是你以严肃的态度把其当作专业技能学习.

很多人之所以不赞同我的核心观点,是因为内心深处还有把 Emacs 当玩具来炫耀"我有多酷"的意识.

Emacs 强大到可以作为另类娱乐来博眼球.但本质是专业人士使用的神器.

打个比方,职业杀手对于刀只关心两件事:

1. 高效地杀人
2. 任何环境下都可靠

刀的装饰是否漂亮或技巧是否自己原创对他并不重要.

Emacs 就是那把刀.

如何报 bug

本文官方网址为 https://github.com/redguardtoo/mastering-emacs-in-one-year-guide.

有任何疑问,请在以上网址报 bug.这比 Email 快.因 GitHub 会以邮件通知我,GitHub 邮件永远归类至我的最优先文件夹下.

如给我发 Email,会淹没于垃圾邮件中.

不要复制粘帖本文

Emacs 本质上是一个社区和平台,不断有新的有趣的人和技术出现.我会定期更新本文.

如果你拷贝粘帖全文,会使自己和他人错过更新.

我建议分享本文的链接,

- 中英文纯文字版会发布在 GitHub 上 (https://github.com/redguardtoo/mastering-emacs-in-one-year-guide)
- 中文 HTML 版会发布到我的官方博客 (http://blog.binchen.org/?p=268)
- 考虑到中国大陆的网络情况,还有另一中文 HTML 版镜像 (http://blog.csdn.net/redguardtoo/article/details/7222501)