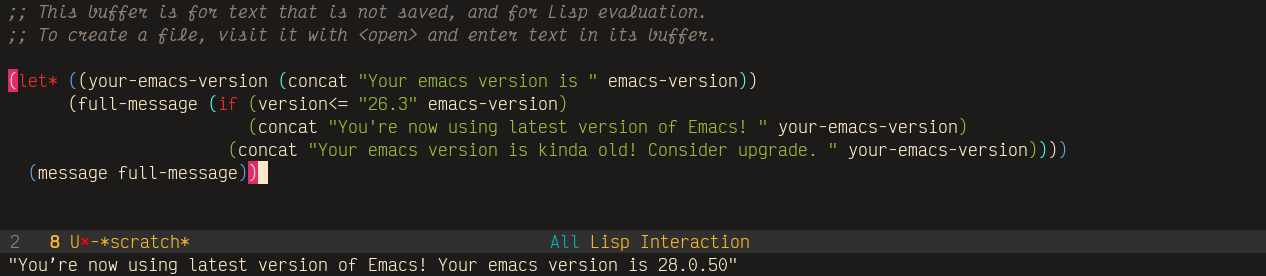
Emacs自力求生指南 – 前言

Chen

# Eamcs是什么？

 就我理解， Emacs 是一个长得像文本编辑器的 REPL ：

* 它是 Emacs lisp ──一个 Lisp 方言──的运行时。
* 每次你打开 Emacs ，它都会使用一个干净的[[1]](#footnote-23)运行时跑一遍 ~/.emacs.d/init.el 脚本。[[2]](#footnote-24)
* 你所看到的内容（窗口、文字、状态栏、光标）以及与它的互动（键盘、鼠标）所造成的内容改变，均为上述脚本执行函数所产生的 副作用 。

# 缺点？

先说缺点吧，客观一些。

## 学习成本高

想要随心所欲地使用 Emacs 的话，不可避免地需要用 Elisp 深度定制。

推荐参考书是 [ANSI Common Lisp](https://acl.readthedocs.io/en/latest/index.html)指南 。

把 Emacs 当作 CL 的 REPL 基本没啥问题，只要 (require 'cl) 就可以用 Common Lisp 的关键字了。

这么想，你在用一个编辑器的时候顺便还能把 LISP 大家族 Racket 、 Clojure ( ClojureScript )、 Scheme 都入了个门，何乐而不为啊。

## 有些操作会阻塞编辑器

虽然 Emacs 26+ 的 async 已经实现得很好了，日常操作基本不会被阻塞。但你依然还是可能会被什么套件里的同步调用卡一下。比如：

* 打开一个 parse 特别费劲的文件，比如一个超大的单行 JSON
  + 等 tree-sitter 完善后这将不会成为一个问题，应该……
* 有些网络 IO，比如 gnus 下载新闻
  + 现代的前后端分离型软件（比如 telega ）不会有这个问题
* 开一个大图片预览 Buffer 或打开一个 PDF
  + 惊人的事实： Emacs 通过把 PDF 转成图片来预览。所以如果你对一个上百 M 的大 PDF 做缩放操作会十分酸爽……

虽然上述都能被 C-g 打断，不过嘛，没有一个工具是万能的。如果你觉得有个功能 Emacs 干得不够好，那就立刻换一个工具吧。时间宝贵。

## 分量不够轻，导致默认装机量不够

这个是真的没办法了… vi （不是 vim ）几乎是每个服务器 Linux 的标配，但 Emacs 的基础包实在太大[[3]](#footnote-29)，甚至不少桌面版 Linux 都不会预装它。不过 macOS 居然预装了它，难道帮主爱用？

## 前后端互动麻烦

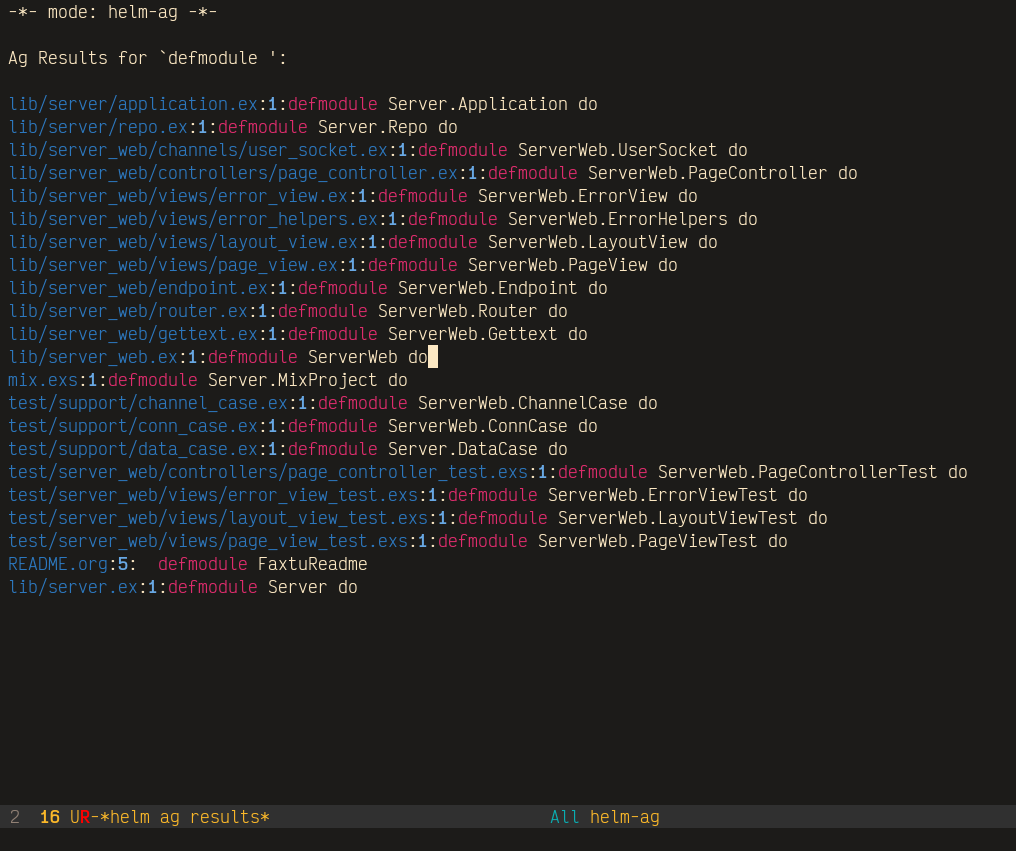
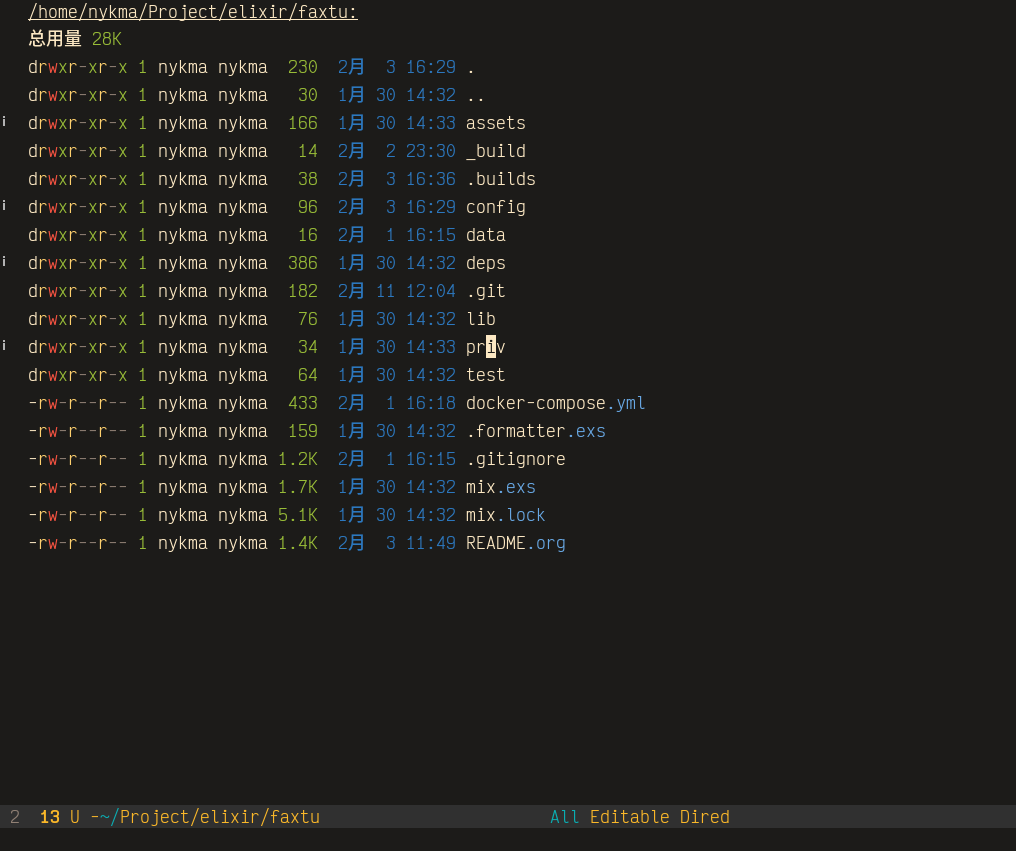
Emacs 作为前端与后端进程通信的场景，在极端情况下，性能不理想：后端 burst 大量数据时可能会卡死 Emacs ，比如 LSP 后端一次性给了太多的补全建议之类的。

如果写 Emacs 的外部动态库，弹性就不太够。

# 长处

## 万物皆文本

除了状态栏（mode line）外[[4]](#footnote-33)，所有你看到的文字都能使用统一的逻辑、统一的环境来互动：

* 你在写文档 (org-mode) ，文档里需要插入一段 C 。org-mode 提供一个函数，把这段嵌入代码映射到一个新的子窗口。你可以在这个子窗口里享受所有编辑 C 项目时你所使用的工具和环境，比如代码补全(LSP 和 company-mode)、预定义的代码片段（ yasnippet ）、语法查错（ flycheck ）等。
* 用 [helm grep](https://github.com/emacs-helm/helm/wiki/Grep) 可以搜索整个 Project ，搜索结果呈现在一个 Buffer 里。你可以 直接修改[这个 Buffer](https://github.com/mhayashi1120/Emacs-wgrep) 并「保存」 。同样，修改过程中你可以使用所有你早已熟悉的文本处理工具和流程，比如可视化正则文本处理器 anzu 、多光标 multiple-cursors.el ，甚至临时写个 elisp 函数并当场执行也可以。
* Emacs 自带的 dired 是一个文件浏览器。同样，你可以在它的 buffer 里「 直接修改并保存 」。从此批量更名再也不用找额外的软件或者记额外的命令。注意图中右下角的 Editable 。
* eval-expression （默认 M-: ） 可以把最底下一行（minibuffer）变成一个临时的 Elisp REPL，这里你可以执行任何 Elisp 函数，结果也会回显在里面。哪怕这个输入框只有一行高度，你会发现编辑体验和编辑一个 .el 文件是一致的：都有括号配平、都有函数名补完、一样能使用片段展开，甚至还能继续用 C-x C-e 来「临时执行表达式的一部分」。
* 这一行的可订制性和大 buffer 是一模一样的，很多软件，诸如 ivy 或者 smex ，都把这一行玩出了花。

## 文本皆结构，编辑文本实为操作结构

首先我强烈建议你花三分钟[看看这个视频](https://www.youtube.com/watch?v=D6h5dFyyUX0) 。

然后来重温一下这个经典的小故事：

转载自译言网 | <http://article.yeeyan.org/view/legendsland/209584> 在 ILC 2002 大会上，前Lisp大神，当今的Python倡导者 Peter Norvig，由于某些原因，做一个类似于马丁路德在梵蒂冈宣扬新教的主题演讲，因为他在演讲中大胆地声称Python就是一种Lisp。

讲完后进入提问环节，出乎我意料的是，Peter点了我过道另一侧，靠上面几排座位的一个老头，他衣着皱褶，在演讲刚开始的时候踱步进来，然后就靠在了那个座位上面。

这老头满头凌乱的白发，邋遢的白胡须，像是从旅行团中落下的游客，已经完全迷路了，闲逛到这里来歇歇脚，随便看看我们都在这里干什么。我的第一个念头是，他会因为我们的奇怪的话题感到相当失望；接着，我意识到这位老头的年纪，想到斯坦福就在附近，而且我想那人也在斯坦福 —— 难道他是……

“嗨，John，有什么问题？” Peter说。

虽然这只是10个字左右的问题，我不会假装自己记住了Lisp之父约翰麦卡锡说的每一个字。他在问Python程序能不能像处理数据一样，优雅地处理Python代码。

“不行。John, Python做不到。”

Peter就回答了这一句，然后静静地等待，准备接受教授的质疑，但老人没有再说什么了。此时，无语已胜千言。

什么叫「像处理数据一样处理代码」？我们知道整个 Emacs 都是用 Elisp 构建起来的，而 Lisp 的迷人之处就在于「代码即数据，数据即代码」： List 在没被求值之前是数据，被求值时就成了代码。

视频里使用的那些快捷键和函数，与其说是文本操作，不如说是操作了语法树后，又重新渲染回 buffer 文本。所以在 Emacs 里写 Lisp 、写 Clojure 、写 Elm 是非常非常享受的事情，心智负担和操作负担都比其它抽象语言好得多。

不要去玩那些括号玩笑了，差远了，用 paredit 写 lisp 根本不需要数括号，哪怕把括号全隐藏都能写出 valid 的程序。

代码是什么？是文本。数据是什么？是结构。「文本即结构」的血脉流淌在 Emacs 的各个角落，除了写 Lisp 之外：

* paredit-everywhere 可以把「编辑语法树」的思想扩展到几乎所有程序语言上
* # 举一个例子， || 表示光标所在位置  
  defmodule Test do  
  ||def abc do  
   "Hello World"  
   end  
  end  
  # C-k 为「删除到行尾」。  
  # 在上述光标所处位置，一般版本的 C-k 会立刻打破 do...end 平衡  
  # 如果使用 paredit-everywhere 提供的 paredit-kill 的话：  
  defmodule Test do  
   ||  
  end  
    
  # 如果光标在引号里呢？  
  defmodule Test do  
   def abc do  
   "Hello ||World"  
   end  
  end  
  # paredit-kill 后：  
  defmodule Test do  
   def abc do  
   "Hello ||"  
   end  
  end  
    
  # 其它括号也是一样  
  defmodule Test do  
   def abc do  
   some\_array = ||[  
   "1",  
   "2",  
   "3",  
   ]  
   end  
  end  
  # paredit-kill 后：  
  defmodule Test do  
   def abc do  
   some\_array = ||  
   end  
  end  
    
  # 我直接把 paredit-kill 绑定在 C-k 了。没这功能我写不了程序。
* org-mode 里的 org 指的是 organized plain text ，就是「文本即结构」的最直接体现。比如：
  + org-refile 会把整个标题及其所属内容移动到另一个标题之下，期间所有的级别变化、缩进都会自动完成
  + 调整标题或列表的上下顺序使用 M-up 和 M-down ，同样是以整个结构为单位的移动
  + 每个元素都分配有自己的 UUID。创建链接使用 UUID ，哪怕目标元素事后改变了位置或内容也不怕
  + 表格明明是纯文本写的，操作起来却和 Excel 差不多，甚至还能写自动计算公式
  + org-capture 可以快速往表格里 append 一行数据（而不用操心这个表格的边框有没有被打断之类的）
  + 甚至还有个 类似 SQL 的软件 可以以复杂的条件组合来查询你的文档库。

## GUI友好、鼠标友好、不反直觉

奇怪的是没几个 Emacs 介绍文提到这个的：Emacs 鼓励你使用它的 GUI 模式。

* 无参数启动 emacs 就是 GUI[[5]](#footnote-49)

# footnote

1. 其实带了一些 Emacs 预设的默认值。比如没有 ~/.emacs.d/init.el 文件时 Emacs 也依然能生成一个窗口来。 [↑](#footnote-ref-23)
2. Emacs 28 之后变成 $XDGCONFIGHOME/emacs/init.el 了，一般是 ~/.config/emacs/init.el [↑](#footnote-ref-24)
3. 如果精简太多就会让 Emacs 失去太多功能，最后变成跟 nano 差不多的存在…… [↑](#footnote-ref-29)
4. 但是这并不意味着状态栏无法定制了。事实上，不仅状态栏软件包 多如牛毛 ，而且甚至还有能 把状态栏和 minibuffer 二合一的软件 ，只能说定制 Emacs 是没有极限的。 [↑](#footnote-ref-33)
5. 也有 CLI 模式： emacs -nw [↑](#footnote-ref-49)