Emacs/elisp 笔记

Yu Yang

November 7, 2013

Contents

1 emacs/elisp 笔记

这是我的 emacs 配置：`https://github.com/yuyang0/emacs.d` , 我在 purcell 的配置的基础上添加了一些我自己的设置

1.1 emacs 大杂烩

1.1.1 emacs 各种设置与注意事项

1. emacs 在英文系统的输入法问题

```
mv /usr/bin/emacs /usr/bin/emacs.raw
gedit /usr/bin/emacs

#!/bin/sh
export LC_CTYPE=zh_CN.utf-8
/usr/bin/emacs.raw "$@"
```

2. 交换 Control_L 与 Capslocks

   通过 xev | grep 'keycode' 得到键码，然后输入以下文件:
   remove Lock = Caps_Lock
   remove Control = Control_R
   keycode 66 = Control_R NoSymbol Control_R
   keycode 105 = Caps_Lock NoSymbol Caps_Lock
   add Lock = Caps_Lock
   add Control = Control_R
   保存为.xmodmap，然后运行 xmodmap .xmodmap

1.1.2 grep

如果是想搜索当前 buffer，那么你可以是 occur，如果你想搜索多个文件，那么可以使用 grep，推荐两个命令

- lgrep：只会搜索当前目录 (不搜索子目录)
- rgrep：它会递归搜索子目录

wgrep 包可以是 emacs 在 grep buffer 中直接修改文件内容

- C-c C-p: 在 grep buffer 中只要按该快捷键，就可以在 grep buffer 直接编辑文件
- C-c C-k: 放弃所有修改
- M-x wgrep-save-all-buffers

1.1.3 isearch 的相关技巧

C-x C-x: 可以在 isearch 之后回到原来的位置

1.1.4 ido-mode 的快捷键

C-s C-r：在匹配的文件左右切换
C-p ：会启动部分匹配，而前面是严格的前缀匹配
C-t ：正则表达式匹配 eg:*.py$ 匹配所有 python 源文件
M-up M-down：切换以前访问的目录

1.1.5 dired 的快捷键

dired 的标记方式有很多种，如标记删除（D 标记，用于删除），如标记为 *（m 所做的标记就是 * 标记），默认如果有被 * 标记的文件，那么所有的文件操作就是在这些标记的文件上进行的，否则就在当前行代表的文件上进行操作，当然还可以有其他类型的标记如 t 标记，k 标记，只是这些标记没有快捷键，一般 D 标记和 * 标记用的多，其他标记一般用于临时的转换一下。*c 命令可以修改标记，eg：*c D t 将 D 标记修改为 t 标记，*c t * 将 t 标记修改为 * 标记

"#": 标记所有自动保存的文件（D 标记）
～：标记所有的备份文件（D 标记）
&：标记所有的垃圾文件 (D 标记)
d: mark D 标记
x：执行删除

m: mark * 标记
t: 将所有 * 标记的文件变为未标记，所有未标记的变为 * 标记
**：标记所有可执行文件
*@：标记所有符号链接
*s: 标记所有文件
*/：标记所有目录（不包括. ..)
%m or *%: 正则匹配文件名
%g：正则匹配所有文件的内容
u: unmark
常用的文件操作：
U: unmark all
D: 立即删除
R：rename
C：copy
A: 正则搜索
Q：正则替换
Z：解压文件
L：把 lisp 文件加载进 emacs
B：byte compile
O：chown
G：chgrp
M：chmod
+：create a directory
!: 可以显示打开文件的命令（对 pdf 等文件不要 RET）

我设置的快捷键：
/ m n: 标记所有文件名匹配正则表达式的文件（mark name）
/ m e: 标记所有可执行文件 (mark executable files)
/ m d: 标记所有的目录 (mark directory)
/ m c：标记所有内容匹配正则表达式的文件 (mark content)
/ m l: 标记所有符号链接 (mark symlink)
/ m u: unmark all marks
/ /：unmark all marks
/ u: 将所有标记的文件的文件重命名为大写
/ l：将所有标记的文件的文件重命名为小写
/ c：change marks，和 *c 等价

1.1.6 ibuffer 的快捷键

p：prev
n: next
m：mark
u: unmark
t: mark all
**: unmark all
x: kill the marked buffer or current buffer if no buffer marked
S: save the marked buffer or current buffer if no buffer marked
D: same as 'x'
O: 在所有 mark 的 buffer 中正则搜索
U: 在所有 mark 的 buffer 中替换
Q: 在所有 mark 的 buffer 中 query，replace
/m: 按主模式过滤
n: 按名字过滤，可以只输入一部分
/: 撤销过滤，全部显示
/p: 多重过滤时撤销上一层过滤
/f: 过滤 filename，与/n 类似

1.1.7 org-mode

org-mode 是一个神器, 有许多强大的功能

1. org-mode 的快捷键

   org-w3m-copy-for-org-mode: 在 w3m 中可以带链接，图片拷贝到 org 中（先选中 region）
   tab，S-tab
   C-c C-n: next tiltle
   C-c C-p: prev tilte
   C-c C-f: 平级的上一个标题
   C-c C-b：平级的下一个标题
   C-c C-u：上一级标题：next
   M-RET : 插入同级标题
   M-S RET：插入一个同级的 TODO 标题
   M-LEFT/RIGHT: 标题升/降级
   M-S-LEFT/RIGHT：子树升/降级
   C-c*：将本行设置为标题
   C-c C-l: insert a link
   C-c C-o: open a link

链接: ”【【link】【descripter】】”（用英文的方括号）
org-mode 能自动识别链接，如 http，file 等
有序列表；1. 2. 3.
无序列表：*，+，-开头都可以
注解的格式：方括号 + 数字，或者方括号 +fn+ 数字
C-c C-c 可以在注解与正文之间跳转
(下面的格式，开头结尾必须都是空格或标点)
a.*…*: 粗体
b./…/: 斜体
c.+….+: ~~删除线~~
d._…_: 下划线
下标：H_2 O(会将 _ 后的字符串加上下标，空格为截止符)
上标：E=mc^2 (同上)

内容元数据及其快捷键：
<v tab: begin_verse 区域内换行
<s tab: begin_src 区域内为代码按 *C-c C-’* 进入主模式编辑，在按退出
<e tab: begin_example 例子，每行以：开始
<q tab: begin_quote 区域左右都会留出缩进，用于引用
<c tab: begin_center 居中区域
<l tab: begin_latex
<h tab: begin_html 嵌入 html
<a tab: begin_ascii

表格：
在某一行的顶格输入 ‘|’，然后输入第一行第一列，在输入 ‘|’，接着第二行第二
列，依次类推，完成后 tab，会将当前行对齐并为你创造表格的下一行，tab 和
#S-tab 可以在表格中正向或者反向的移动，在表格的空白项中输入数字 eg:<6> 则
限定为 6 个字符长，多余的部分会隐藏，用 C-c C-c 可以展开

TODO list：
在一个标题上按 shift+left/right，C-c C-t 可以设置 todo list 的状态

2. Babel 通过 Babel, 你可以直接在 org-mode 中运行各种编程语言的代码, 每一个代码片段都组织为一个 block, 可以向这个代码片段传递参数, 同时每一个代码片段都可以产生输出, 而这个输出又可以作为输入传递给另一个代码片段, 关于 Babel 的详细介绍可以看这篇官方的Instruction 与这篇pdf , 特别是后者, 建议认真看看, 只有 26 页,Babel 的基本结构分为数据块和代码块数据块:

```
#+name: <name>
<data block body>
```

使用该数据时, 直接引用 <name> 就好代码块:

```
#+name: <name>
#+headers: <header arguments>
#+begin_src <language> <header arguments>
 <body>
#+end_src
```

- name: 如果指定, 那么最后得到结果就会赋给该变量, 可以通过该变量将代码块的执行结果作为参数传递给其他代码块

```
#+name: ret1
#+BEGIN_SRC python :results output
print 'hello world'
#+END_SRC

#+RESULTS: ret1
: hello world

#+BEGIN_SRC sh :var arg=ret1
echo $arg
#+END_SRC

#+RESULTS:
: hello world
```

- language: 代码的类型,eg: C, cpp, python, ruby
- header-arguments: 头部参数, 头部参数可以放在两个地方: 代码块的上方以及 <language> 后, 常用的头部参数:
  (a) :results output(捕捉输出), :results value (default, 捕捉最后一个表达式的结果, 代码块当做一个函数)

(b) :file aa.png (将输出存入文件, 下面会加入链接, 这对于输出是图片 (eg:gunplot) 时非常方便)

(c) :dir ~/Documents (将 ~/Documents 作为运行代码的进程的当前目录)

```
#+headers: :dir ~/Documents/blog
#+BEGIN_SRC sh
echo $PWD
#+END_SRC

#+RESULTS:
: /home/yangyu/Documents/blog
```

(d) :var n=5 (传递一个变量 n 给 code block, 并且其值为 5)

(e) :exports both (输出 code 与 results), :exports code(默认), :exports results, :exports none

现在我放入几个例子:

3. dot dot 语言可以用来画流程图，和 Babel 结合起来非常方便，这是一份 dot 语言的guide

1.1.8 cua-mode 以及矩形操作的快捷键

enable: M-x cua-mode
C-RET: 激活矩形操作，然后就可以常规移动光标来进行列编辑
C-v : past rectangle

1.1.9 mutiple cursors 多光标

1. C-<: mc/mark-previous-like-this
2. C->: mc/mark-next-like-this
3. C-c <: mc/mark-all-like-this (上面三项功能以 word 为单位，必须先 mark-word)
4. C-c c a: 在所有行的开头加入光标
5. C-c c e: 在所有行的结尾加入光标
6. C-c c c: 在所有的行加入一个光标 (上面的三项功能以行为单位，必须先 mark 数行)

1.1.10 ace-jump-mode

类似于火狐的 vim 插件，输入一个 head char，就会在所有匹配的地方放入一个字母

1. C-; ： ace-jump-char-mode
2. C-: : ace-jump-word-mode

1.1.11 w3m

w3m 是一个文本浏览器，它不支持 CSS 与 js，非常适合浏览文本很多的页面，以及避免在 emacs 与 firefox 切换带来的烦恼。。

links

- g: Prompt for a url in minibuffer (w3m-browse-url)
- G: same as **g** excepte it open a new session(a new tab)
- R: reload the page
- S: search engion
- H: goto home page
- u: display the under the point in the echo area
- RET: Display the page pointed by the link under point (w3m-view-this-url)

: move point to previous form

- : move point to next form
- TAB: move point to next link
- M-TAB, S-TAB: move point to previous link
- d: download thee url under the point
- M-d: Download the url

scroll

- SPC: scroll downwards
- >: scroll to the right
- <: scroll to the left

- DEL: scroll upwards

bookmarks

- v: show all bookmarks
- a: add current url to bookmarks
- M-a: Add the url under point to the bookmark.
- C-k: kill a bookmark
- E: edit bookmark

move in page

- hjkl: like vim

switch tabs

- C-c C-n: next tab
- C-c C-p: previous tab
- C-c C-t: new tab

images

- I: Display the image under point in the external viewer.
- M-i: Save the image under point to a file.
- t: Toggle the visibility of an image under point
- T: Toggle the visibility of all images
- M-T: turn off to display all images
- M-[: zoom in an image on the point
- M-]: zoom out an image on the point

1.1.12 ansi-term

- C-c C-j: 进入 line mode，可以复制
- C-c C-k: 回到 character mod

1.1.13 我设置的快捷键以及我常用的快捷键，函数

M-x browse-url: 通过浏览器打开当前 url
C-g C-/: redo
C-x C-v: find-alternate-file(fresh buffer)
C-x C-m: 替代 M-x, 按 M 很别扭
M-u : upcase-word
M-c : capticalize-word
M-l : downcase-word(将行上下移动，如果选定则移动 buffer)
C-c p: 复制粘贴一行
M-up: Shift lines up
M-down: Shift lines down
M-x occur: 创建一个新的 buffer，然后将当前 buffer，所有匹配 regex 的内容显示其中, 进入该 buffer，enter 就可以跳转
M-x imenu：可以根据类型跳转（变量，函数,include 等等）
C-k: kill the current line
C-o: new line and indent,like the 'o' in vim
M-;: comment or uncomment the region
C-c C-c: same as M-;
C-/: undo (same as C-x u and C-_)
C-c d: translate the word using sdcv
C-c f: 迭代的搜索字符，eg：C-c f g 会移到第一个 g，在按 g 移到
第二个 g
% : jump to the matched parenthesis
C-c -: fold the code
C-c =: unfold the code
C-M-f: 表达式的首部
C-M-b:
C-M-a: 函数的首部
C-M-e：
C-x backspace: 删除到行首
C-M-h: 标记一个函数
M-u: 一个字改为大写
M-l：一个字改为小写
C-x C-u：region to uppercase
C-x C-l：区域小写
C-t：将光标前后的字符交换
C-x C-t: 光标所在行与上一行交换

1.1.14 emacs 导出 pdf 的中文支持

org-mode 默认的导出系统对中文支持不好, 我参考这篇文章, 将导出中文 pdf 的方法总结如下:

1. 安装 textlive, 如果是 ubuntu, 运行以下命令:

   ```
   sudo apt-get install texlive texlive-xetex textlive-latex-extra
   ```

   textlive-latex-extra 一定要装, 不装会有 File 'wrapfig.sty' not found. 的错误

2. 在 emacs 配置文件中添加

   ```
   ;; org-mode < 8.0
   (setq org-latex-to-pdf-process '("xelatex -interaction nonstopmode %f"
                                    "xelatex -interaction nonstopmode %f"))
   ;;  org-mode 8.0
   (setq org-latex-pdf-process '("xelatex -interaction nonstopmode %f"
                                 "xelatex -interaction nonstopmode %f"))
   ```

3. 在 org 文档的开头部分添加:

   ```
   #+LATEX_HEADER: \usepackage{xeCJK}
   #+LATEX_HEADER: \setCJKmainfont{SimSun}
   ```

1.1.15 emacs 的奇淫技巧 face

- M-x flush-lines RET ^$ RET : 删除所有的空行（正则匹配）
- M-x list-faces-display: 显示 faces，在很多需要颜色，字体样式的命令中需要 face 参数
- 绑定快捷键时用 C-h k 查询得到的内容可以直接放在 kbd 后
- indent-region 可以格式化所有的代码

1.2 elisp 学习笔记

1.2.1 elisp 基本语法

1. help: + M-x apropos (find symbol)
   - C-h k(descripte key)
   - C-h f (function)

- C-h v (variable)

2. control flow

   (a) if

   logical expression the function **or** works like the logical "or" in most languages: if all the arguments are false, it return nil, otherwith it will return the value of the last argument whose value is non-nil. eg: (or nil nil 3 2 1) return 3 (not t) so (if a a b) is identical to (or a b)

3. let

4. lists car cdr cons list: (list 'a "b" 1) ==> (a "b" 1) append: (append '(a b) '(c d)) ==> (a b c d) reverse: (reverse '(a b c d)) ==> (d c b a) nthcdr: call cdr n times length: get the length of the list mapcar: equal: only test the object's structure and content eq: test if the two arguments is the same object assoc: 当 list 类似于关联数组时，用 key 来寻找 value，用 equal 测试 key assq：与 assoc 相似，只是用 eq 测试 key

   setcar: change the car-element of the list setcdr: change the cdr-element of the list

1.2.2 常用的函数

1. **lisp** 的基本语法

   (a) list

   - property list

   ```
   (setq alist '(:publish-dir "~/Documents" :base-dir "~/Documents/note"))
   (plist-get alist :base-dir)
   ```

   - assoc list

2. 光标：

3. 文本编辑：

4. 字符串操作：

5. **buffer** 相关：

6. **file** 相关：

7. **other useful function**

```
(save-excursion)    ;; 保存并恢复当前 point
(interactive)       ;;function is a command
(thing-at-point THING) ;;THING can be 'word' 'line' 'sentence' 'list'
                       ;; and so on
(thing-at-point 'word)
(thing-at-point 'sexp)
(current-time-string) ;; 返回当前时间字符串
(format-time-string "%1.%M %p" (current-time))
(symbol-name 'sym)                      ;conmvert symbol to string
(intern "sym")                          ;convert string to symbol
```

8. **useful variable**

```
mark-active
last-command
```

1.3 一些我遇到的 emacs 让人困惑的问题

1.3.1 overwrite mode

该 mode 打开的话，会替换光标后面的字符，而不是插入，非常令人不爽