

emacs 实践笔记

aborn

2016-12-04 11:02

Contents

1	emacs 的书签功能	1
1.1	设置一个书签	1
1.2	列出保存的书签	1
1.2.1	书签列表 *Bookmark List*	1
1.3	跳转到一个书签	2
1.3.1	helm-bookmarks	2
1.3.2	修改默认排序	2
1.4	删除一个书签	2
1.5	保存书签	2
1.6	其他设置	3
1.7	bookmark+	3
2	dired 模式	3
3	常用命令	3
3.1	移动命令	3
3.2	编辑命令	3
3.3	其他命令	4
4	什么是函数？	4
5	函数定义	4
6	检查一个函数是否定义	4
7	函数参数	4

8 函数调用	4
8.1 funcall	4
8.2 apply	5
9 org-capture.el	5
10 magit 模式简介	5
11 常用命令	5
12 分支操作	5
13 org 模式简介	6
14 文档结构	6
14.1 目录结构	6
14.2 显示与隐藏	6
14.3 列表	6
14.4 块结构	6
15 表格	7
16 超链接	7
16.1 链接格式	7
16.1.1 内部链接	7
16.1.2 外部链接	7
16.1.3 链接处理相关命令	7
17 待办事项	7
18 日程表 (Agenda View)	8
18.1 日程文件 (Agenda files)	8
18.2 分发按键	8
18.3 内建 Agenda 视图	8
18.4 计划 Schedule	8
19 Org 快速记录	8
19.1 如何使用 org-capture?	8
19.2 org 条目复制与移动	9
19.3 记录模板	9
19.3.1 快捷键	9
19.3.2 描述	9

19.3.3 类型	9
19.3.4 目标文件	10
19.3.5 模板	10
19.3.6 属性 properties	10
20 Org 的导出功能	10
20.1 导出的 Dispatcher	10
21 包列表	10

1 emacs 的书签功能

emacs 的书签用于记录你在文件中的阅读位置。它有点类似寄存器，跟寄存器一样，因为它也能记录位置位置。但同寄存器有两点不一样：1. 它有比较长的名字；2. 当 emacs 关闭的时候，它会自动持久化到磁盘。

1.1 设置一个书签

当我们阅读一个很长的文档，没能一口气读完时。我们希望记住当前文档的最后阅读的位置，以便下次再用 emacs 阅读的时候能快速地定位到。那么，我们设置一个书签，通过 **bookmark-set** 对应快捷键为 **C-x r m**

1.2 列出保存的书签

bookmark-bmenu-list 对应快捷键为 **C-x r l**，它将打开一个 *Bookmark List* 的 buffer 同时列出所有保存的书签。

1.2.1 书签列表 *Bookmark List*

在 *Bookmark List* 这个 buffer 里，有以下快捷键可以使用：

- a 显示当前书签的标注信息；
- A 在另一个 buffer 中显示所有书签的所有标注信息；
- d 标记书签，以便用来删除 (x - 执行删除)；
- e 编辑当前书签的标注信息；
- m 标记书签，以便用于进一步显示和其他操作 (v - 访问这个书签)；
- o 选中当前书签，并显示在另一个 window 中；

- C-o 在另一个 window 中切换到当前这个书签;
- r 重命名当前书签;
- w 将当前书签的位置显示在 minibuffer 里。

1.3 跳转到一个书签

使用 **bookmark-jump** 函数，可以跳转到一个特定的书签，它绑定的快捷键为 **C-x r b**。如果你的 emacs 中安装了helm 这个插件，你也可以使用 **helm-bookmarks** 这个命令来快速查找书签，并跳转到书签位置。

1.3.1 helm-bookmarks

通过 helm-bookmarks 命令来查找并跳转书签如下图：

1.3.2 修改默认排序

书签查找和跳转的时候，默认的书签排序是按字母排序的。如果想将最近访问的书签放在最前面，将下面代码添加到你的 emacs 配置文件中。

```
(defadvice bookmark-jump (after bookmark-jump activate)
  (let ((latest (bookmark-get-bookmark bookmark)))
    (setq bookmark-alist (delq latest bookmark-alist))
    (add-to-list 'bookmark-alist latest)))
```

1.4 删除一个书签

删除一个书签对应的命令为 **bookmark-delete**。

1.5 保存书签

最新版本 emacs（老版本的书签保存在 `~/.emacs.bmk`），在退出的时候会自动保存书签。如果想手动保存书签的话，可以采用 **bookmark-save** 这个函数命令。默认的情况，emacs 会将书签保存在 **bookmark-default-file** 变量对应的文件中。在我的机器中，对应的文件如下：

```
ELISP> bookmark-default-file
"/Users/aborn/.emacs.d/.cache/bookmarks"
ELISP>
```

1.6 其他设置

有一个变量 **bookmark-save-flag**。如果这个变量的值为一个数值，它表示修改（或新增）多少次书签后，emacs 会自动保存书签到磁盘。当这个变量的值被设置为 1 时，每次对 bookmark 的改动，emacs 就会自动保存内容到磁盘相应位置（这样可以防止 emacs 突然 crash 时 bookmark 的丢失）。如果这个值设置为 nil，表示 emacs 不会主动保存 bookmark，除非用户手动调用 **M-x bookmark-save**。

1.7 bookmark+

bookmark+ 是对 bookmark 的一个扩展的包。它有更多的功能：

1. 原始的 bookmark 只能对文件位置记录,bookmark+ 对孤立的 buffer(没有关联文件的 buffer) 也能保存书签;
2. 支持对书签进行打 tag;
3. 对文档的某个区域保存为书签，而不仅仅是某个位置;
4. 记录了每个书签的访问次数，及最后一次的访问时间，可以基于它们排序;
5. 多个书签可以有相同的名字;
6. 可以对函数、变量等加书签。

更多功能请参考: <https://www.emacswiki.org/emacs/BookmarkPlus#Bookmark%2b>

2 dired 模式

dired 的全称为 Directory Edit，即目录编辑，是一个非常老的模式。

3 常用命令

3.1 移动命令

n 下移 **p** 上移

3.2 编辑命令

R 重命名或者移动文件 **D** 删除文件或者目录 **+** 创建目录 **Z** gzip 压缩文件

3.3 其他命令

RET 打开文件或者目录 **q** 退出

4 什么是函数？

函数是有传入参数的可计算的规则。计算的结果为函数返回值。大部分计算机语言里，函数是有一个名字的。从严格意义来说，lisp 函数是没有名字的。注意：函数也是一个 lisp 对象，把这个对象关联到一个 symbol, 这个 symbol 就是函数名

5 函数定义

定义一个函数的语法如下：“`elisp defun name args [doc] [declare] [interactive] body. . .`”

6 检查一个函数是否定义

检查一个函数是否定义 “`elisp (fboundp 'info) ; t (fboundp 'setq) ; t (fboundp 'xyz) ; nil`”

7 函数参数

有些参数是可选的，当用户没有传是，设置一个默认值，下面是一个例子：
“`elisp (defun piece-meal/fun-option-parameter (a &optional b) (when (null b) (message "parameter b is not provided") (setq b "ddd"))) ; ; set to default value (message "a=%s, b=%s" a b))`”

8 函数调用

函数的调用有两种方式，****funcall**** 和 ****apply****

8.1 funcall

funcall 它的语法如下：

`funcall function &rest arguments`

因为 `funcall` 本身是一个函数，因此 `funcall` 在调用前，它的所有参数都将事先做求值运算。注意参数 **function** 必须为一个 Lisp 函数或者原生函数，不能为 Special Forms 或者宏。

8.2 apply

`apply` 的语法如下：

```
apply function &rest arguments
```

它与 `funcall` 功能类似，唯一不同的是它的 `arguments` 是一个列表对象。

9 org-capture.el

Org 8.0 以后版本采用 `org-capture.el` 取代原有的 `org-remember.el`

10 magit 模式简介

`magit` 是 emacs 下版本管理的强大武器

11 常用命令

- **magit-dispatch-popup** 命令分发器，在 spacemacs 里绑定到 **M-m g m**
- **magit-diff** 相当于 `git diff`，当进入 `diff-buffer` 后按 `g` 更新之
- **magit-status** 相当于 `git status`，进入 `status-buffer` 后按 `s` 添加文件或文件夹到本地仓库
- **magit-checkout** 切换分支
- **magit-branch-and-checkout** 从当前分支切一个新的分支

12 分支操作

常用的分支操作如下：

- `(magit-branch-delete) b k` 删除一个或多个（本地）分支
- `(magit-branch-rename) b r` 对当前 Branch 进行重命名

13 org 模式简介

Emacs 的 org-mode 可用于记笔记、管理自己的待办事项 (TODO lists)，同时，也可用于管理项目。它是一个高效的纯文本编辑系统。

14 文档结构

Org 是基于 Outline-mode，并提供灵活的命令编辑结构化的文档。其文档结构语法跟 markdown 很类似。

14.1 目录结构

Org 的目录结构在每行最左边以星号标记，星号越多，标题层级越深。下面是一些例子：

```
\* 一级目录
\** 二级目录
\*** 三级目录
```

```
\* 另一个一级目录
```

14.2 显示与隐藏

目录结构下的内容可以隐藏起来，通常用采用 **TAB** 和 *S-TAB* 这两个命令来切换。

14.3 列表

Org 提供三种类型的列表：有序列表、无序列表和描述列表

1. 有序列表以 '1.' 或者 '1)'
2. 无序列表以 '-', '+' 或者 '*'
3. 描述列表

14.4 块结构

在 Org 文档中，加入代码块这种类型的块结构，都是采用 begin...end 这种模式，下面是一个例子：

```
\#+BEGIN_EXAMPLE
\#+END_EXAMPLE
```


15 表格

16 超链接

Org 模式提供了比较好用的超链接方式，可以链接到普通网页、文件、email 等。

16.1 链接格式

Org 模式支持两种链接，即，内部链接和外部链接。它们有相同的格式：

`[[链接]][描述]]` 或 当只有链接没有描述 `[[链接]]`

一旦链接编辑完成，在 org 模式下，只显示 **描述**部分，而不会显示整体（后一种只显示链接）。为了编辑链接和描述，需要通过快捷键 **C-c C-l** 来完成。

16.1.1 内部链接

内部链接是指向当前文件的链接，它的链接格式：

`[[#链接ID]]`

其中 **链接 ID** 是文档中唯一的标识 ID

16.1.2 外部链接

Org 支持的外部链接有很多中形式，如文件、网页、新闻组、电子邮件信息、BBDB 数据条目等。它们以一个短的标识字符串打头，紧接着是一个冒号，冒号后面没有空格字符。

16.1.3 链接处理相关命令

- `org-store-link` 保存当前位置的一个链接，以备后面插入使用
- `org-insert-link` 插入链接，绑定的快捷键为 `C-c C-l`，如果光标正在一个链接上，那么 `C-c C-l` 的行为是编辑这个链接及其描述

17 待办事项

Org 模式用来管理自己的 TODO list 非常方便

18 日程表 (Agenda View)

我们可以用 Org 来安排自己的行程

18.1 日程文件 (Agenda files)

变量 `org-agenda-files` 保存了一个文件列表, 这些文件用来记录日程, 下面是一些操作函数: `C-c [` 将当前文件加入到 agenda 文件列表最前页面 `org-agenda-file-to-front C-c]` 将当前文件从 agenda 文件列表中删除 `org-remove-file`

18.2 分发按键

默认采用 **C-c a**, 接下的默认的命令有:

- a 创建一个日程
- t/T 创建一个 TODO items
- L 对当前文件生成 timeline

18.3 内建 Agenda 视图

18.4 计划 Schedule

用 org 来安排日程

- `org-schedule` 将当前 TODO 添加计划时间

19 Org 快速记录

有时候, 突然想到一些待办事项, 或者一些突发的灵感。这时, 我们用 emacs 快速记录它, Org-Capture 提供这个好用的功能。它的前身是 `org-remember.el` (注: 从 org 8.0 开始, `org-remember` 被 `org-capture`) 替代。

19.1 如何使用 org-capture?

快速记录的命令为 **M-x org-capture**, 默认绑定的快捷键为 **C-c c**。当这个命令被调用后, 你可以使用自己定义好的 模板 快速创建记录。一旦完成内容的输入, 按下 **C-c C-c** (`org-capture-finalize`), 来完成。然后, 你就能继续做你当下的事。如果想跳转到刚刚创建的记录的 buffer, 用 **C-u C-c C-c** 来完成。如果想中途中止输入, 只要按下 **C-c C-k** (`org-capture-kill`)。

19.2 org 条目复制与移动

有时候，我们想将当前的某条目转移到其他文件或者其他项目里。这时，我们会用到 `org-copy` 和 `org-refile` 这两个命令。它们对应的快捷键分别是 `C-c M-w` 及 `C-c C-w`。这里有一个问题是，目标文件如何配置？目录文件的配置由一个变量决定，`org-refile-targets`，我自己的配置如下：

```
(setq org-refile-targets
      '((nil :maxlevel . 3)          ;; 当前文件的最大层级
        (aborn-gtd-files :maxlevel . 3)))
```

注意：我这时将文件放在 `aborn-gtd-files` 文件列表里。

19.3 记录模板

记录的模板为一个列表变量，`org-capture-templates`，列表的每条记录由如下几段组成：

```
("t" "Todo" entry (file+headline (expand-file-name org-default-notes-file org-directory)
                                   "* TODO %?\n  创建于:%T  %i\n"))
```

19.3.1 快捷键

如例子中的那样，“t”表示对应按键 `t` 这个快捷键。它能帮助我们快速地选中哪条模板进行快速记录。

19.3.2 描述

接下来是一段简单的描述

19.3.3 类型

第三段表示类型，有五种类型：`entry` `item` `checkitem` `table-line` `plain`

- `entry` 普通的 Org 结点，保证目标文件为 `org-mode` 文件，插入的时候将作为目录结点的子结点

(如果没有，将做为顶级结点)；

- `item` 与 `entry` 类似，不同点在于它的目标文件可以为简单的纯文本文件；
- `checkitem` 复选条目；
- `table-line` 在目标文件中的第一个 `table` 中插入新行；
- `plain` 纯文本记录

19.3.4 目标文件

第四个字段配置目标文件

19.3.5 模板

第五个字段表示模板，模板参数 含义如下：

- %t 只有日期的时间戳
- %T 日期 + 时间的时间戳
- %u,%U 如上，只不过它们是 inactive 的
- %i 初始化文本，当前上下文将作为初始化文本

19.3.6 属性 properties

最后一个字段表示属性列表，支持以下属性配置：

- :prepend 一般一个记录条目插入在目标文件的最后，这个属性可以将条目插入在最前
- :immediate-finish 立刻完成，没有交互
- :clock-in 对这个条目设置闹钟
- :kill-buffer 如果目标文件没有相应的访问 buffer，插入后，自动关闭 buffer

20 Org 的导出功能

Org 文件支持导出多种格式的目标文件，如 ASCII 文件、HTML 文件 (用于发布为 Web)、PDF 文档等。

20.1 导出的 Dispatcher

任何导出命令都有一个前缀按键，我们称之为 Dispatcher，为 **C-c C-e**

21 包列表

1. elisp-slime-nav 写 elisp 代码时，可用于跳转到函数的定义