

基于 Emacs 的 L^AT_EX 编辑环境

Date: 2011,5

Version: Draft

By: siziki

Email : siziki@gmail.com

目 录

第 1 章	\LaTeX 编辑环境选择	3
第 2 章	环境准备	4
2.1	安装 Emacs	4
2.2	安装和配置 AUCTEX	4
2.2.1	关于快捷键	7
2.2.2	关于代码折叠	8
2.3	RefTeX 的安装和配置	9
2.4	$\text{CD}\LaTeX$ 安装和配置	13
2.5	yasnipet+auto-complete	15
2.5.1	构建 \LaTeX 模版	16
2.5.2	构建自定义环境	17
2.6	关于 preview-latex	18
2.7	我的 el	18

第 1 章 \LaTeX 编辑环境选择

使用 \LaTeX 编辑文档，有很多软件可以选择，构成不同的编辑环境。

在不同的平台环境下，有不同的选择，比如 windows 环境下的 WinEdit, TeXMakerX 等，MacOS 下有 TeXShop 等，GNU/Linux 下有 kile。还有通用的 TeXworks。

这些软件基本都是为 \LaTeX 编辑而生的。优点是上手容易、简单易用，当然简单易用的代价就是配置的灵活性不够。

除此以外还有有些通用的编辑环境，通过适当的配置，就可以实现强大的 \LaTeX 的编辑环境，这种编辑环境的强大程度往往只取决于你的想象力。

本文就是要介绍基于 Emacs 的 \LaTeX 编辑环境。

有几点需要提前说明：

1. 本文使用 \LaTeX 发行版是 **TeXLive2010**。
2. 本文使用的 \TeX 引擎是 $X_{\text{Y}}\TeX$ ， $\LaTeX+CJK$ 的情况不涉及。
3. 操作系统环境是 Debian GNU/Linux，不同系统环境下的插件安装不同。

第 2 章 环境准备

2.1 安装 Emacs

当然首先是要有 Emacs，主流的 GNU/Linux 发行版都可以方便的安装，以 Debian 为例，

```
$ sudo apt-get install emacs
```

裸奔的 Emacs 是看起来不是那么可爱，需要很多很多的配置，Emacs 的功能也取决于你怎么配置它，这里就详细说了。需要好几本书才可以说清，甚至都说不清。

结合下面的配置，就可以建立一个基本的 L^AT_EX 编辑环境了。

2.2 安装和配置 AUCT_EX

然后就是安装 AUCT_EX 了。

Debian 下安装 AUCT_EX 有两个选择，一个是使用 Debian 的工具：

```
$ sudo aptitude install auctex
```

Debian 可能还会安装相应的依赖，它会安装源里面的 TeXLive，而这个 TeXLive 不是最新版的，所以不推荐这样安装 AUCT_EX。

根据在 Debian 下的经验，推荐安装最新版的 AUCT_EX，它对 preview-L^AT_EX 的支持更好。

可以通过下面的方式安装最新版的 AUCT_EX：

```
cvs -d:pserver:anonymous@cvs.sv.gnu.org:/sources/auctex co .
cd auctex
./configure
```

```
make
make install
```

接下来就是要配置 AUCT_EX 了，经过配置的 AUCT_EX 才强大。
首先，设置基本的选项：

```
(load "auctex.el" nil t t)
(load "preview-latex.el" nil t t)
(setq TeX-auto-save t)
(setq TeX-parse-self t)
(setq-default TeX-master nil)
```

将这些配置写入 `.emacs` 中，然后打开一个 T_EX 文件看看，菜单栏上就会有相应的 L^AT_EX 编辑命令和选项了，如图 2.1。

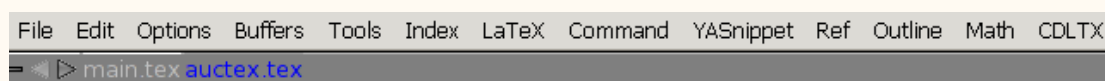


图 2.1 L^AT_EXmode 下的菜单栏

AUCT_EX 有很多选项可以设定，可以到 AUCT_EX 的[主页](#)上看这些选项的说明。

这里列出和我平常编辑 L^AT_EX 关系比较密切的选项设定。

在看下面的选项配置之前，需要说明的是，下面的配置代码都是包含在一个 `hook` 之内的^①：

```
(add-hook 'LaTeX-mode-hook (lambda ()
  .....
))
```

好了，可以看具体的配置了：

^① 不了解 `hook`？找本 Emacs 的介绍来简单看看吧。

```
;; LaTeX 模式下, 不打开自动折行
(turn-off-auto-fill)

;; 显示行数
(linum-mode 1)

;; 打开自动补全
(auto-complete-mode 1)

;; 启动 math mode, 你也可以不用。
(LaTeX-math-mode 1)

;; 打开 outline mode
(outline-minor-mode 1)

;; 接下来是和编译 TeX 有关的
;; 编译的时候, 不在当前窗口中显示编译信息
(setq TeX-show-compilation nil)

(setq TeX-clean-confirm nil)
(setq TeX-save-query nil)

;; 按 \ 后光标跳到 mini-buffer 里面输入命令
;; 看个人习惯, 因为如果有了 auto-complete 和 yasnippet
;; 这个不开启也问题不大。
(setq TeX-electric-escape t)

;; 重新定义 pdf viewer, 我设定为了 evince。
(setq TeX-view-program-list ' ("Evince" "evince %o"))
```

```
(setq TeX-view-program-selection
      '((output-pdf "Evince")))

;; 设置编译引擎为 XeTeX
(setq TeX-global-PDF-mode t
      TeX-engine 'xetex)
;; 使用 XeLaTeX 作为默认程序来编译 LaTeX
(add-to-list 'TeX-command-list
              '("XeLaTeX" "%\xelatex%(mode)%" %t"
                TeX-run-TeX nil t))
(setq TeX-command-default "XeLaTeX")
```

这些设置，基本够用。

打开一个 TeX 文档，然后测试一下编译，默认的编译快捷键是：

```
C-c C-c
```

更多选项，就去看 AUCTeX 手册页吧。

2.2.1 关于快捷键

AUCTeX 自己定义了很多快捷键，这写快捷键如果熟悉之后，可以大大提高速度，你可以浏览一下都有哪些可以用的快捷键，当然也可以自己定义自己顺手的快捷键。

列出几个常用的快捷键：

```
C-c C-c TeX-command-master ;; 编译，当前环境下是 XeLaTeX
C-c C-e LaTeX-environment ;; 插入环境，默认 section
C-c C-f TeX-font ;; 字体设置快捷键前缀，
C-c C-j LaTeX-insert-item ;; 插入 item，
C-c C-k TeX-kill-job ;; 取消编译
```

```
C-c C-v TeX-view ;; 打开 pdf viewer
C-c ; TeX-comment-or-uncomment-region ;; 注释
```

2.2.2 关于代码折叠

如果文档很长，就需要折叠相应的代码，**AUCTeX** 也提供了良好的支持，首先你需要打开 `fold`,

```
(TeX-fold-mode 1)
```

然后到一个编辑的文档的 `buffer` 中，输入下面的快捷键：

```
C-c C-o C-b
```

就可以折叠当前 `buffer` 中的所有支持的环境和 `macro`。

当然，你发现有很多环境没有被 **fold** 起来，不要担心，这些都是可以配置的，通过下面的方式来增加你需要配置的环境：

```
(setq TeX-fold-env-spec-list
      (quote (([figure]" (figure"))
              ([table]" (table"))
              ([itemize]" (itemize"))
              ([overpic]" (overpic))))))
```

然后，你再来试试下面的快捷键，所有自定义的环境应该都可以被折叠了。就可以自定义折叠环境的这点来说，**AUCTeX** 就是很强大的。

```
C-c C-o C-b
```

到目前为止，**Emacs** 和 **AUCTeX** 已经能够完成基本的 **L^AT_EX** 编辑工作了。可是到遇到大的文档编辑时，文档之间的互相引用的问题，**AUCTeX** 是解决不了的，这就需要另外一个工具，那就是 **RefTeX**。

2.3 RefTeX 的安装和配置

RefTeX 的主要作用是在 \LaTeX 方便的实现类似 `ref`, `cite` 的功能, 尤其是当文档比较大, 需要较多 `ref` 的时候, RefTeX 就显得尤其好用。

下载最新版的 RefTeX 包, 然后安装:

```
tar zxvf reftex-4.34.tar.gz
cd reftex-4.34
make
$ sudo make install
```

然后开始配置, 首先需要在 `.emacs` 中写入:

```
(require 'reftex)
(add-hook 'LaTeX-mode-hook 'turn-on-reftex)
(setq reftex-plug-into-AUCTeX t)
(setq reftex-enable-partial-scans t)
(setq reftex-save-parse-info t)
(setq reftex-use-multiple-selection-buffers t)
(autoload 'reftex-mode "reftex"
          "RefTeX Minor Mode" t)
(autoload 'turn-on-reftex "reftex"
          "RefTeX Minor Mode" nil)
(autoload 'reftex-citation "reftex-cite"
          "Make citation" nil)
(autoload 'reftex-index-phrase-mode
          "reftex-index" "Phrase mode" t)
```

这个时候, 重启一下 `emacs` 或者, `eval-buffer` 一下, 你就会看到, 菜单栏多出了一个菜单项:

Ref

点击最上面的

Table of Contents

你就可以看到当前文档的章节目录列表。

更方便的是，使用快捷键：

```
C-c =
```

就可以生成一个新的 **buffer**，在 **buffer** 里面显示当前文档的章节目录，默认是水平的一个 **buffer**，如图 2.2。

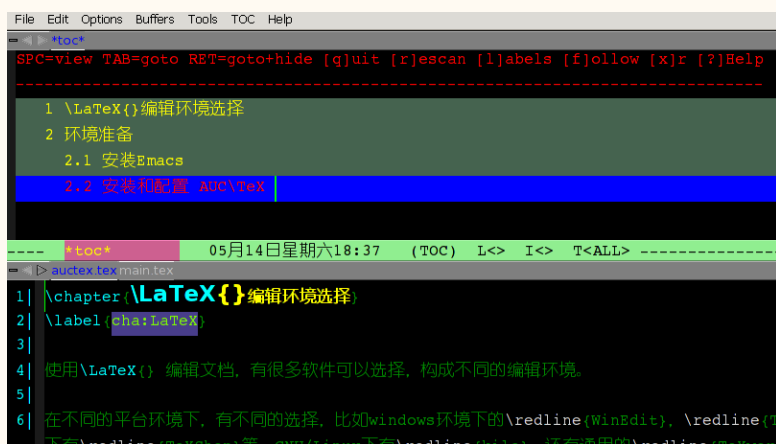


图 2.2 水平的 toc buffer

在这个 **buffer** 里面，可以用快捷键操作实现自己想要的目录显示结果，根据上面的说明操作即可，很简单。

不习惯水平显示 **buffer**？

那就改，这就是 **Emacs** 的灵活强大之处，按照下面的方式来设置 **toc buffer** 的显示方式：

RefTeX 不仅可以修改 **toc buffer** 显示的位置，连宽度都可以设置：

```
;;*toc*buffer 在左侧。
(setq reftex-toc-split-windows-horizontally t)

;;*toc*buffer 使用整个 frame 的比例。
(setq reftex-toc-split-windows-fraction 0.2)
```

这个时候的 toc buffer 就显示在左侧了，如图2.3。

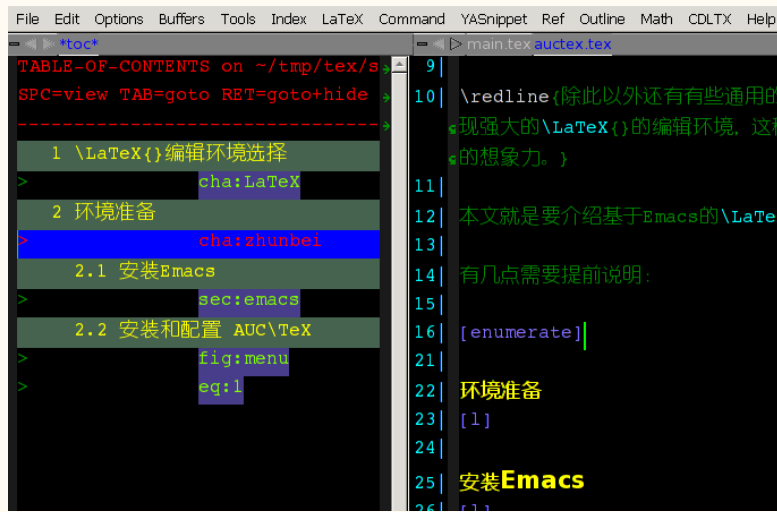


图 2.3 buffer toc 在左侧

多说一句，在 toc buffer 里面，快捷键 1 可以显示相应章节里面的 label。就像图2.3中所显示的那样。

除了在 toc 里面显示章节目录之外，RefT_EX 还提供一个菜单，就是 index 菜单，图2.4所显示的就是当前章节的 index 目录。



图 2.4 index 菜单

怎么开启这个菜单呢？

```
(imenu-add-menubar-index)
```

如果当你新增了一些 `section` 之后，`index` 菜单里或者 `toc-buffer` 里面并没有更新出来相应的章节信息，这时可以使用 `RefTeX` 的一个命令：

```
tex-parse-one
```

来重新分析一下当前文档，或者用

```
tex-parse-all
```

来分析所有的文档。

这个命令可以在 `Ref` 菜单里找到，图 2.5。

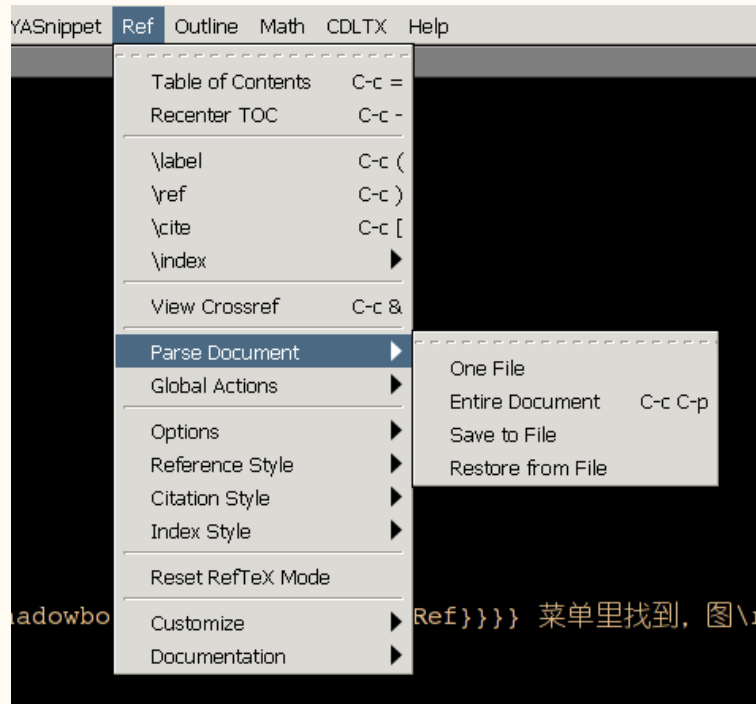
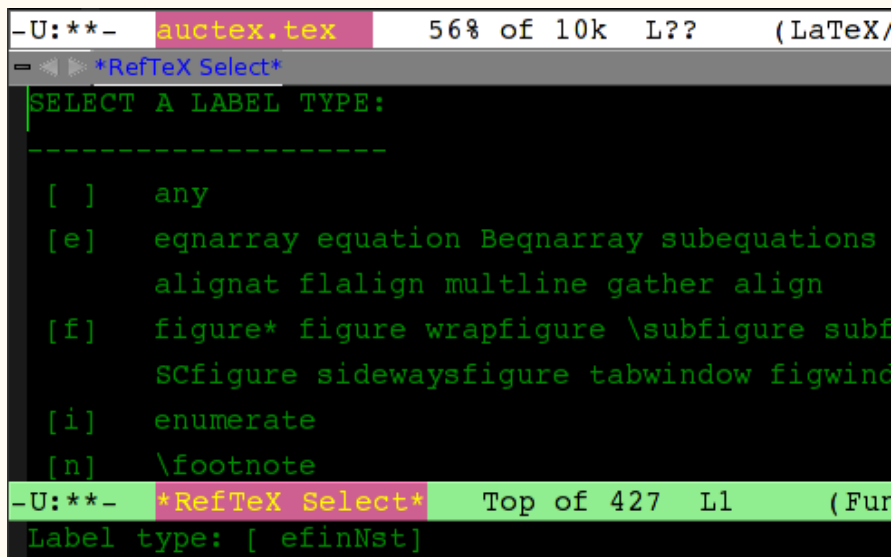


图 2.5 `RefTeX` 的 `parse` 菜单

你可以将这个命令绑定到一个快捷键上，我是这么绑定的：

```
(define-key LaTeX-mode-map (kbd "C-c C-p")
  'reftex-parse-all)
```

$\text{AUC}\text{T}\text{E}\text{X}$ 还支持章节的引用，当你在文档中输入 \backslash 的时候，会自动跳到 mini buffer 中，^① 输入 `ref`，回车之后，你会看到图 2.6 的提示。



```

-U:**- auctex.tex 56% of 10k L?? (LaTeX/
- RefTeX Select*
SELECT A LABEL TYPE:
-----
[ ] any
[e] eqnarray equation Beqnarray subequations
alignat flalign multiline gather align
[f] figure* figure wrapfigure \subfigure subfigure
SCfigure sidewaysfigure tabwindow figwindow
[i] enumerate
[n] \footnote
-U:**- *RefTeX Select* Top of 427 L1 (Fur
Label type: [ efinNst]

```

图 2.6 RefTeX 的 label 提示

其中 `s` 代表 `section` 的 `label`，`f` 代表 `figure` 的 `label`，根据你的需要，输入相应的选项，你就会看到当前文档中的 `label` 列表，然后选择你需要的那个 `label` 即可，如图 2.7。

需要说明的是，有时候需要重新 `parse` 文档，你才可以看到 RefTeX 的 `label` 提示。如果你觉得这个不太好用，可以采用后面的 `auto-complete+yasnippet` 的方式。

2.4 $\text{CD}\text{L}\text{A}\text{T}\text{E}\text{X}$ 安装和配置

然后是安装 $\text{CD}\text{L}\text{A}\text{T}\text{E}\text{X}$ ， $\text{CD}\text{L}\text{A}\text{T}\text{E}\text{X}$ 的主要作用是增加一部分环境 (Environment) 的自动补齐功能，比如说，我要加入一个 `equation` 的环境，我只要输入 `equ`，然后 `TAB`， $\text{CD}\text{L}\text{A}\text{T}\text{E}\text{X}$ 就自动补全出下面的代码，

```

\begin{equation}
\label{eq:1}

```

^① 当然这取决于你的设置，你可以不让他调转。



图 2.7 RefTeX 的 label 选择

```
\end{equation}
```

很方便吧。

在[这里](#)下载最新版的 **CDLaTeX**。这个包很简单，就是一个 **elisp** 包，下载之后，放到你的 **Emacs** 可以找到的地方，比如：

```
~/.emacs.d/site-lisp/
```

然后让 **Emacs** 可以找到它，将下面的语句写入 **.emacs** 中

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

然后开始配置 **CDLaTeX**

```
(add-hook 'LaTeX-mode-hook 'turn-on-cdlatex)
(autoload 'cdlatex-mode "cdlatex" "CDLaTeX Mode" t)
(autoload 'turn-on-cdlatex "cdlatex" "CDLaTeX Mode" nil)
```

整体上，**CDLaTeX** 的功能相对简单，它的功能基本上也可以由 `auto-complete+yasnippet` 来实现。

2.5 yasnippet+auto-complete

说了好久 yasnippet+auto-complete, 具体是怎么回事呢? [ahel](#)对此有个解释, 挺不错, 可以去看看。

yasnippet用在 Emacs 的 \LaTeX 编辑环境中, 基本上也是和补全有关系的。不过它还有更重要的功能, 那就是自定义环境以及文档模版相关。具体配置, 后文说明。

先安装, 在 [yasnipet](#) 主页下载最新版的 yasnippet, 然后安装:

```
tar xjvf yasnippet-0.6.1c.tar.bz2
cp -r yasnippet-0.6.1c ~/.emacs/site-lisp/yasnippet
```

让 emacs 可以找到 yasnippet:

```
(add-to-list 'load-path "~/.emacs.d/site-lisp/yasnippet/")
```

为了能够更好的发挥 yasnippet 的功能, 需要配合 auto-complete。

到 [auto-complete](#) 主页上下载,

```
tar xjvf auto-complete-1.3.1.tar.bz2
cd auto-complete-1.3.1
make
make install
```

在配置 yasnippet 和 auto-complete 的时候, 我借用了 [DEA](#) 的配置, 他已经做了很好的设定, 我直接拿过来用。^①

到 [auto-complete-settings.el](#) 下载最新版的 auto-complete-settings.el, 放入:

```
~/.emacs/site-lisp/
```

^① DEA 的配置中, 有些地方是使用了他自己定义的一些设置, 如果补全不能使用, 看看他的代码, 将需要的 library 下载下来, 加一下就可以使用了。

然后再去 [yasnippet-settings.el](#) 下载 `yasnippet` 的设置，放入

```
~/ .emacs/site-lisp/
```

接下来，在 `.emacs` 中，加入两者的设定：

```
(require 'auto-complete-settings)
(require 'yasnippet-settings)
```

这样的话，已经基本实现了一个具有自动补全、方便 `ref`、可以 `preview` 的 \LaTeX 编辑环境了。

这个时候，如果一切顺利，那你就可以实现自动补全功能了。

需要说明的，`DEA` 自己绑定了一些快捷键，如果你的补全用着不顺手，可以修改一下 `DEA` 对于快捷键的设定。

但是 `yasnippet` 还有别的功能，下面分别介绍。

2.5.1 构建 \LaTeX 模版

如果你喜欢使用 `ctexart` 来构建文档，你可以在 `latex-mode` 下构建一个 `yasnippet` 的片段，大体上是下面这个样子：

```
# name: ctexarttemplate
# key: ctexarttemplate
# --
\documentclass[cs4size,adobefonts]{ctexart}
\begin{document}

\end{document}

%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
```



```
%%% End:
```

然后，当你想构建一个采用 `ctexart` 的文档时，就可以在 `yasnipet` 里面找到你定义这个片段，就会在当前的 `buffer` 中建立一个 `ctexart` 框架。

先说说怎么构建 `yasnipet` 片段吧。

```
首先，切换到相应的 mode 中
然后，M-x yas/new-snippet
输入你希望的片段名字
```

按照 `yasnipet` 的提示，就建立一个当前 `mode` 下的 `yasnipet` 片段。

注意到上述片度里面的 `key` 这个地方那个了吧，如果你定义了这里，在结合 `auto-complete` 的情况下，当你输入 `ctex` 的时候，就会给出你几个补全提示，如图 2.8，不用全部输入了，直接选择即可。

```
414 |
415 | ctex
416 | ctexart
417 | ctexbooktemplate a 好的工作，需要将下面的设置放入 .e
418 | ctex [shell]
477 | CTEXsetup
478 | CTEXoptions
479 | ctexarttemplate a
480 | %%% mode: LaTeX
481 | %%% TeX-master: "main"
482 | %%% End:
483 |
484 |
```

图 2.8 补全示例

2.5.2 构建自定义环境

除了模版的补全之外，你还可以通过 `yasnipet` 来定义一些自己的 `macro` 或者环境，比如如果你定义一个 `figure` 环境，通过 `yasnipet`：

```
#name : figure
#key : figure
# --
\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.8\textwidth]{figure/xx.png}
  \caption{caption}
  \label{fig:label}
\end{figure}
```

这里也定义了一个 `key`，今后如果你想插入一个图片，输入 `fig...`，`TAB` 一下，`yasnipet+auto-complete` 应该会为你补全出来这段代码。

如果一切顺利，你应该已经搭好了一个非常棒的 \LaTeX 环境了。

我用过很长时间的 `kile`，但是最后还是自己动手搭建了基于 `Emacs` 的环境，这个比 `kile` 要方便不少。

2.6 关于 `preview-latex`

本文没有介绍 `preview-latex`，主要是有两个方面的考虑：

- `preview-latex` 对于中文的支持还需要费些精力才可能比较好的使用
- 更重要的是，`preview-latex` 不支持 $\text{Xe}\text{\LaTeX}$ 。

如果你确实有需要，可以去看看 `preview-latex` 的文档，我用了几次，后来觉得还是要使用 $\text{Xe}\text{\LaTeX}$ ，就暂时不用 `preview-latex` 了。

2.7 我的 `el`

最后，我将所有的配置写在一个 `el` 里面，可以到[这里](#)下载。