Day01 - Spacemacs 的安装

在开始之前，确保你的电脑上已经安装了 Emacs，如果是 Windows 系统的话，确保你已经添加了 HOME 环境变量 如果不知道 Windows 系统怎么添加 HOME 环境变量，可以查看 Emacs 疑难杂症板块

本文介绍如何安装 Spacemacs

> Spacemacs 是一套 Emacs 配置文件，准确来说，应该是安装 Spacemacs 这套配置

Windows 安装 Spacemacs

- 通过 git 安装

如果你的电脑安装了 git 的话，打开 PowerShell，然后执行下面这条命令来安装 Spacemacs：

```
shell git clone https://github.com/syl20bnr/spacemacs $home/.emacs.d
```

因为要从 GitHub 下载，速度可能有点慢。如果速度太慢的话，可以从 CodeChina 的镜像进行安装：

```
shell git clone https://codechina.csdn.net/mirrors/syl20bnr/spacemacs/-/tree/develop.git $home/.emacs.d
```

Windows 系统默认并没有安装 git,你可以采用下面的方式来手动安装。

- 手动安装

可以在 Spacemacs 的 GitHub 仓库下载 Spacemacs 的源码压缩包，如果速度太慢的话，可以在 Spacemacs 的镜像源仓库下载，下载完成后解压缩并将文件夹命名为 `.emacs.d` ，然后放置到主目录下即可。

Linux 和 macOS 安装 Spacemacs

Linux 和 macOS 通常都默认安装了 git，打开终端，然后执行下面这条命令来安装：

```
shell git clone https://github.com/syl20bnr/spacemacs ~/.emacs.d
```

如果速度太慢的话，可以从 CodeChina 的镜像进行安装：

```
shell git clone https://codechina.csdn.net/mirrors/syl20bnr/spacemacs/-/tree/develop.git ~/.emacs.d
```

第一次启动 Spacemacs

安装完 Spacemacs 之后，启动 Emacs 就会自动从 Melpa 下载包(packages)了，但是 Melpa 的服务器在国外，所以包下载的速度会特别慢。我们可以更改镜像源解决这个问题，我们这里使用清华的镜像源为例。

启动 Emacs 时，它会问你想要使用那种编辑模式 vim(默认)或是 emacs，根据自己的喜好选择好之后回车即可，接着会询问你是想安装标准版的 Spacemacs(默认) 还是精简版的 Spacemacs-base，同样自己选择之后回车即可。

此时应该会开始下载包了，并且会在主目录下生成一个 `.spacemacs` 文件。通过任务管理器或系统监视器关闭 Emacs，然后在 `.spacemacs` 中找到 `dotspacemacs/user-init()` （可以按 `C-s` 进行文本搜索），在这个函数里面添加如下的代码：

```
lisp (setq configuration-layer-elpa-archives '(("melpa-cn" . "http://mirrors.tuna.tsinghua.edu.cn/elpa/melpa/") ("org-cn" . "http://mirrors.tuna.tsinghua.edu.cn/el
```

添加完成后应该如下图所示：

此时再启动 Emacs 就会从清华源开始下载包，速度应该会快很多，然后耐心等待它下载完就可以了。全部下载完之后，重启一下 Emacs，就可以看到 Spacemacs 的图标了。

如果你选择的不是 Emacs 的编辑模式，那么在安装完所有包后可能会有一条关于 `evil-want-keybinding` 的警告，你需要在 .spacemacs 中的 `dotspacemacs/user-init` 添加：

```
lisp (setq evil-want-keybinding nil)
```

Day02 - Spacemacs 的简单配置

在阅读本文前，确保你已经正确安装好了 Spacemacs，且启动之后没有任何报错。

设置适合自己的编辑模式

在上一节安装 Spacemacs 时，Spacemacs 让我们选择自己的编辑模式（Vim或Emacs），安装完 Spacemacs 后如果我们对当前的编辑模式不满意，可以在 `.spacemacs` 文件中进行更改。

打开 .spacemacs 文件，按 `C-s` ，搜索 `editing-style` ，可以找到编辑模式的配置选项。可选的值有 Vim、Emacs、Hybrid，根据自己的喜好选择，也可以根据官方文档进行进一步的配置。

显示行号

> 在 Spacemacs 中，SPC（即空格键）是 Vim 编辑模式下默认的 leader 键，如果你使用 Emacs 编辑模式，默认的 leader 键是 `M-m` ，本教程统一使用 SPC 表示

按 `SPC f e d` 可以快速打开 .spacemacs 文件，按 `C-s` 搜索 `line-numbers` 可以找到一个叫做 `dotspacemacs-line-numbers` 的选项，这个选项可以显示行号并设置行号的风格，可选的值有 `visual` 、 `relative` 还有 `t` ，visual 和 relative 都是显示相对行号，但是 visual 虽然显示相对行号，但是在使用命令来跳转时还是会按绝对行号来进行跳转。t 显示普通的行号（绝对行号）

设置字体和字体大小

按 `C-s` 搜索 `font` 一般可以找到一个名为 `dospacemacs-default-font` 的选项，更改双引号中的内容可以修改字体，Spacemacs 默认的字体是 Source Code Pro，建议根据自己的喜好改成自己喜欢的**等宽**字体

如果你使用 Windows 系统的话，Consolas 应该是你最熟悉的等宽字体了，如果你是用 macOS 系统的话，可以试试 Menlo

Windows 系统默认并没有安装 Source Code Pro 字体，可以搜索进行安装，或者更改为已经安装的字体

`:size` 选项可以更改字体的大小，不同的字体大小不一样，所以个值并不是固定的，得根据自身需要选择， `:weight` 可以修改字体的字重， `:width` 可以修改字体的宽度。

另外，大多是等宽字体并不包含中文字体，因此 Emacs 中的中文大多会以宋体来显示，可以用下面的方法单独设置中文字体。

在 .spacemacs 中搜索 `user-config` 定位到用户配置，然后添加下面的代码：

```
lisp (dolist (charset '(kana han cjk-misc bopomofo)) (set-fontset-font (frame-parameter nil 'font) charset (font-spec :family "Your Font" :size 14)))
```

请将 `Your Font` 修改为你自己的中文字体，其中 `:size` 可以更改字体的大小。

Emacs 默认无法显示 Emoji 颜文字，你可以单独设置 Emoji 字体，例如：

```
lisp (set-fontset-font t 'unicode (font-spec :family "Noto Color Emoji" :size 16))
```

字体名依据你使用的 Emoji 字体而定

更改主题

在 .spacemacs 搜索 `theme` 可以找到一个叫做 `dotsapcemacs-themes` 的选项，在里面可以选择自己想要的主题，最靠前的是默认主题，键入主题包名后，下次启动 Emacs 时会自动下载并安装主题，默认的 spacemacs-dark 和 spacemacs-light 已经很好看了，如果你不喜欢的话，可以去 Emacs Galley

按 `SPC T s` 可以快速切换主题

用户配置

Spacemacs 默认提供了很多的可配置选项，如果想像普通的 Emacs 那样增添自己的配置，可以在 .spacemacs 中搜索 `user-config` ，可以定位到一个叫做 `dotspacemacs/user-config` 的选项，将你的配置填写在里面即可，比如显示时间的配置 `(display-time-mode t)` ，添加完成后代码一般如下所示：

```
lisp (defun dotspacemacs/user-config () "Configuration for user code: This function is called at the very end of Spacemacs startup, after layer configuration. Put )
```

Day03 - Spacemacs 中的 layer（上）

本文介绍了 layer 这一概念，并引导读者添加一些常用的 layer 。

layer 是什么？

layer 的中文意思是“层”，Spacemacs 中的一个 layer，即一个配置层。配置层的大体含义是将多个你需要的包以及包的配置等集中到一个地方，就是层。如果需要使用 Spacemacs 中对应的功能，就只需要添加对应的配置层就可以了。

layer 极大的简化了 Emacs 用户安装和使用配置包的过程。如果你希望 Spacemacs 可以编写 Python 程序，你需要安装对应的语法检查工具、自动补全后端和对应的语言服务器，而使用 layer 的话，你只需要在 .spacemacs 文件的某个地方写上 python 这个词就可以了。

同时，得益于高度整合的功能，layer 也更便于管理，无论是安装还是卸载，都只需要在对应的地方添加或删除某些文本。

有关于 layer 的更多信息，可以查看官方文档。

添加一些常用的 layer

在 .spacemacs 文件中找到 `dotspacemacs-configuration-layers` ，将自己需要的 layer 填写在里面，下次启动 Spacemacs 时就会开始安装了。以下是对一些常用的 layer 的介绍

- auto-completion

提供自动补全功能，添加 lsp 和对应语言的支持之后，在编写代码时会有相应的语法提示。建议添加

- better-defualts

优化 Emacs 的一些常用操作，可以使 Emacs 用起来更顺手。建议添加

- emacs-lisp（默认已安装）

提供对 elisp 语言的支持。建议添加

- lsp

提供了对微软语言服务器协议的支持，添加了这个 layer 和对应的语言支持之后，在编写代码时将会有更智能的语法提示。建议添加

- markdown

提供了对 markdown 的支持，包括生成预览和目录，如有相关需要，建议添加

- multiple-cursors（默认已安装）

提供了多光标支持，即在编辑文件时可以使用多个光标，可以根据自己需要添加

- org

提供了 Spacemacs 中的 org 模式，提供了很多丰富的功能，如有需要，建议添加

- syntax-checking

提供了针对许多语言的语法检查。建议添加

- spell-checking

提供了拼写检查功能。建议添加

- treemacs（默认已安装）

提供了文件导航功能和文件操作支持，可以在 Spacemacs 左侧显示当前目录下的文件。建议添加

- helm（默认已安装）

一个重量级的补全解决方案，在搜索文件、文本或者管理项目和 layer、输入命令时都提供了强大的补全功能。建议添加

- ivy

和 helm 相似，但更加的轻量的替代品

> 值得注意的是，如果同时添加 helm 和 ivy，helm 将会被 ivy 替换掉，两者不能共存，因此要根据自己的需要进行取舍，但不论是 helm 还是 ivy，强烈建议在两者之中选择一个，这将会对你使用 Spacemacs 提供极大的便利

- git

提供了对 git 的强大支持，如有相关需要，建议添加

- chinese

提供了对中文的强大支持，包括但不限于输入法、词典、更好的中文显示等，如有需要，建议添加

- unicode-fonts

提供了 unicode 字符支持，包括更好的字体显示和字体连字支持

- eaf

提供了 Emacs Application Framework的支持，添加此 layer 之后可以使用功能完成的 Chrome 浏览器、PDF 阅读器、视频播放器等

此外，还有一些针对常用编程语言的 layer，在你打开相应的文件时，Spacemacs 将会询问你是否需要添加相应的 layer，可以根据自己的需要添加

另外，添加相应的语言支持后，会列出可以使用的语言服务器列表供你安装，一些语言服务器并不可以通过 Emacs 安装，如果你想使用别的语言服务器， 你可以通过查看 lsp-mode 的官方文档获取安装方法

在你安装一个相应语言的服务器之后，Spacemacs 将不会再询问你是否需要安装对应的语言服务器。完整的 layer 列表，请查看 Spacemacs layers list，有关对应 layer 的配置，同样可以参考这份文档。

Day04 - Spacemacs 中的 layer（下）

本文将介绍一些使用频率较高的 layer 的配置

> 本文将会长期更新

Treemacs

基本操作

> 以下操作请在 treemacs 缓冲区内完成，可以按 `SPC 0` 来切换到 treemac 缓冲区

在 treemacs 中可以使用 `C-n` 和 `C-p` 来上下移动选中文件，按回车键即可打开一个文件

按键	对应的操作
?	查看帮助
cf	新建文件
cd	新建目录
R	重命名
q	隐藏/显示 treemacs 目录
Q	退出 treemasc
ov	垂直分屏来打开一个文件
oh	水平分屏来打开一个文件
th	隐藏或显示隐藏的文件
m	移动一个文件

另外， `SPC f t` 可以快速打开/关闭 treemacs

treemacs 提供了一个简单而强大的文件导航目录，通常情况下，它会自动根据当前的文件来决定应该显示哪个 目录，但大多数时候，treemacs 并不会这么做，因此你需要手动切换目录。

| 按键 | 对应的操作 | | :--: | :----------: | | M-H | 导航到上一级目录 | | M-L | 导航到下一级目录 |

> 注意，H 和 L 是大写的，因此你还需要同时按下 Shift！

或者，你可以使用 `M-x treemacs-root-up` 和 `M-x treemacs-root-down` 来移动目录

自动刷新

在你删除或者新建一些文件之后，treemacs 不会立即刷新当前的目录试图，你需要手动刷新。如果希望自动 刷新的话，请更改 dotspacemacs-configuration-layers 中的 treemacs

```
lisp (treemacs :variables treemacs-use-filewatch-mode t)
```

主题

你可以使用 all-the-icons 主题来显示 treemacs 上的图标，配置方式同上

```
lisp (treemacs :variables treemacs-use-all-the-icons-theme t)
```

如果你同时修改了“自动刷新”和“主题”两个配置，修改后的代码应该类似于：

```
lisp (treemacs :variables treemacs-use-filewatch-mode t treemacs-use-all-the-icons-theme t)
```

LSP

安装语言服务器

在打开相应的程序文件后，你可以通过输入 `M-x lsp` 手动来安装对应的语言服务器，这种方法适用于无法自动 安装语言服务器的情况

header line 的显示和隐藏

启用 lsp 并安装对应语言的支持之后，通常会在相关的代码文件上显示显示当前文件的路径和图标，如果你想要 隐藏的话，可以在 user-config 中添加 `lsp-headerline-breadcrumb-enable nil` ，你也 可以通过 `lsp-headerline-breadcrumb-segments` 来进行更细微的调整

- `project` 显示当前项目的名称
- `file` 显示当前文件的名称
- `path-up-to-project` 显示当前项目的路径
- `symbols` 显示图片

可以修改为下面这样

```
lisp (setq-default dotspacemacs-configuration-layers '((lsp :variables lsp-headerline-breadcrumb-segments '(symbols)))) ;; 只显示图标 ;; 这只是用于在文档中说明的代码配置示
```

或者

```
lisp (setq-default dotspacemacs-configuration-layers '((lsp :variables lsp-headerline-breadcrumb-segments '(project file)))) ;; 显示项目名称和文件名称
```

代码错误统计

默认情况下，lsp 会统计当前项目下所有的错误，如果要禁用，应该修改 `lsp-modeline-code-actions-enable`

```
lisp (setq-default dotspacemacs-configuration-layers '((lsp :variables lsp-modeline-code-actions-enable nil)))
```

此外，可以通过使用 `lsp-modeline-code-actions-segments` 进行更细微的修改

- `icon` 显示相应的错误的图标，代码错误显示为红色圆点，警告显示为橙色圆点
- `name` 显示出现错误的代码的名称
- `count` 显示出现错误的代码的数量

```
lisp (setq-default dotspacemacs-configuration-layers '((lsp :variables ;; default segments lsp-modeline-code-actions-segments '(count icon))))
```

Syntax Checking

syntax-checking 使用 flycheck 作为语法检查工具，默认情况下，flycheck 会在一些具有检查意义的 地方提供语法检查，如各种各样的源代码文件 javascrpt、java、c/c++ 等，如果你希望它在 lisp 等源代码 文件上也提供语法检查的话，可以在 user-config 中添加：

```
lisp (global-flycheck-mode 1)
```

Day05 - Spacemacs 的进阶配置（上）

本文将会详细讲解 .spacemacs 文件中的大部分配置，读者可以根据自己的需要自行修改。

.spacemacs 文件基本介绍

.spacemacs 文件一般会自动生成在主目录下，这个文件是配置 Spacemacs 的入口，有关于 Spacemacs 本身的配置基本都能在里面进行修改，用户设置同样在这个文件中修改。

.spacemacs 中，内容一般被分为以下几个部分，每个部分都封装在一个函数中：

- `dotspacemacs/layers`

在这里可以声明一些 layer，以及删除、增添一些包，在这里还可以调整 Spacemacs 加载时的一些行为

- `dotspacemacs/init`

Spacemacs 绝大部分的配置都位于此，你可以在此修改配置中可选的选项，但绝对不能将自己的用户配置代码添加在这里

- `dotspacemacs/user-init`

这里的内容会在 Emacs 启动前开始加载，一般在这里设置你需要使用的 elpa 源，你应该尽量把用户配置放在 `dotspacemacs/user-config` 中

- `dotspacemacs/user-config`

在这里可以添加你的用户配置代码，你自己的定义的大部分配置一般都在这里完成

- `dotspacemacs/emacs-custom-settings`

Spacemacs 自己生成的配置，同样不建议自己去修改

开始配置

接下来开始讲解，可配置的选项按出现的位置进行排序（从上到下）

dotspacemacs/layers

该部分主要是关于 Spacemacs 中 layer 的声明和配置，和 Spacemacs 安装软件包的行为的配置

- `dotspacemacs-distribution`

可选的值有 spacemacs-base 和 spacemacs，spacemacs-base 可以理解为是一个精简版 的 Spacemacs，默认为 spacemacs ，即一个完整的 Spacemacs

- `dotspacemacs-enable-lazy-installation`

> 惰性安装： 本配置的主要功能是当你打开一个文件时，spacemacs 会自动根据文件的类型安装相应的 layer,就比如打开一个 java 源代码文件，spacemacs 便会询问你是否需要安装对应的 layer，类似于 vscode，当你打开一个文件时，将会询问你是否需要安装相应的扩展

该选项可选的值有 all、unused 和 nil，all 将会惰性安装所有支持惰性安装的 layer，unused 只会安装你没有使用的 layer，nil 将禁用惰性安装这一功能，默认为 unused

- `dotspacemacs-ask-for-lazy-installation`

对应于上面的选项，在安装相应的 layer 前会询问你是否需要安装，可选的值有 t 和 nil，默认为 t

- `dotspacemacs-configuration-layer-path`

如果你配置了自己的 layer，但是却把它放在了 .emacs.d 以外的文件夹中，为了让 spacemacs 可以找到这个 layer，就需要配置这个选项。Spacemacs 将会额外从这个目录中寻找 layer。值为一个目录的路径，结尾要带上 "/"

- `dotspacemacs-configuration-layers`

声明 layer 的地方，在此声明过的 layer 将会在下次启动 Spacemacs 时自动安装，也可以进行一些 layer 的配置，如：

```
lisp dotspacemacs-configuration-layers '(html python auto-completion better-defaults emacs-lisp (lsp :variables lsp-headerline-breadcrumb-enable nil) markdown org
```

- `dotspacemacs-additional-packages`

添加单独的包的地方将包名填入括号中后，下次启动 Spacemacs 时就会自动开始安装，也可以在此进行一些简单的包配置。**不在此选项中出现的包将会在下次启动 Spacemacs 被删除，因此不建议使用 package-install 安装包**

- `dotspacemacs-frozen-packages`

将包名添加到这个选项中后，该包将会禁止更新

- `dotspacemacs-excluded-packages`

主要用于移除一些 layer 中你不想要的包，填写在这里这里的包将会被移除

- `dotspacemacs-install-packages`

定义了 Spacemacs 安装包的一般行为，可选的值有 used-only、used-but-keep-unused 和 all

> 关于包的定义： 你使用的包：你声明过的 layer 中包含的包和在 dotspacemacs-additional-packages 中添加的包。 你没有使用的包：没有填写在 dotspacemacs-additional-packages 中的包，这些包将会被删除。

used-only 会安装*你使用的包*，而删除那些*你没有使用的包*，used-but-keep-unused 会安装*你使用的包*，但不会删除*你没有使用的包*，all 会安装 Spacemacs 支持的**所有包**

dotspacemacs/init

这里包含了 Spacemacs 中绝大部分的可配置选项，你不应该把你自己的配置代码添加在这里，并且这个部分的内容会在 Spacemacs 启动的最开始生效。注：有部分不影响使用的配置没有讲到（其实是因为不知道）

- `dotspacemacs-elpa-https`

在访问 elpa 存储库时使用 https 域名，可选的值有 nil 和 t，默认为 t，建议为 t

- `dotspacemacs-elpa-timeout`

设置访问 elpa 的超时时间，单位为秒，默认值为5

- `dotspacemacs-use-spacelpa`

可选的值有 t 和 nil，如果是 t，将默认从 Spacelpa 安装稳定的包，如果是 nil，将默认从 Melpa 安装最新的包，默认为 nil

- `dotspacemacs-verify-spacelpa-archives`

可选的值有 t 和 nil，如果是 t 并且如果你开启了从 Spacelpa 安装包，则会验证下载的包的签名，默认为 t

- `dotspacemacs-check-for-update`

可选的值有 t 和 nil，如果是t，并且你使用的 Spacemacs 不是 develop 分值的话，将会在启动时检查更新。默认为 nil

> 注意，我们默认安装的 Spacemacs 就是 develop 分支，develop 不能通过这个来更新 Spacemacs 版本，我们也没有必要频繁的更新 Spacemacs。同时，我也并不推荐使用 master 分支的 Spacemacs，因为 master 分支的 Spacemacs 和文档距离上次更新都已经将近五年了。

- `dotspacemacs-editing-style`

设置 Spacemacs 的编辑风格，可选的值有 vim、emacs 和 hybrid，hybrid 更像是 vim 和 emacs 的结合体，它将 vim 的插入模式改为了 emacs 的编辑模式，你同样可以通过 `:variables` 来修改一些默认的选项

- `dotspacemacs-startup-buffer-show-version`

在启动时是否显示 Spacemacs 和 Emacs 的版本，可选的值有 t 和 nil，默认为 t

- `dotspacemacs-startup-banner`

设置 Spacemacs 启动时显示的 Logo，可选的值有 official、random、nil 和你自己图像的路径，这个路径必须是一个字符串。默认为 offical，如果是 nil，则不显示 Logo

- `dotspacemacs-statup-lists`

设置 Spacemacs 启动时显示的列表，值有 recents、recent-by-project、bookmarks、projects、agenda、todos，可以显示最近打开的文件、项目、书签和 todo 等，接受一个列表，项后面的数字是该项目显示的长度，修改为 nil 可以不显示

- `dotspacemacs-startup-buffer-responsive`

可选的值有 t 和 nil，如果是 t 的话，那么你就可以调整 buffer 的大小，默认为 t

- `dotspacemacs-show-start-list-numbers`

在开始界面的列表的项前面显示数字（如果你开启了显示列表的话），这样的话，直接按数字就可以打开对应的文件、项目等，可选的值有 t 和 nil，默认为 t

- `dotspacemacs-startup-buffer-nulti-digit-delay`

按下数字键后的最小延迟时间（即等待时间），单位为秒，默认为 0.4

- `dotspacemacs-new-empty-buffer-major-mode`

打开一个新的缓冲区后默认的模式，可选的值有 text-mode 和 nil，如果设置成 nil，则使用基本模式，默认为 text-mode

- `dotspacemacs-scratch-mode`

用来设置 scratch 这个 buffer 默认的模式，可选的值有 text-mode 和 nil，默认为 text-mode

- `dotspacemacs-scratch-buffer-persistent`

可选的值有 t 和 nil，如果是 t，你在 scratch 缓冲区写下的所有内容都会被自动保存，默认为 nil

- `dotspacemacs-scratch-buffer-unkillable`

可选的值有 t 和 nil，如果是 t， `kill-buffer` 不会 kill 掉 scratch 缓冲区，而是 bury（机翻是埋葬？我不好怎么翻译） 它，默认为 nil

因为要讲的东西实在太多，一部分内容迁移到了 Day06

Day06 - Spacemacs 的进阶配置（下）

书接上文⬇

- `dotspacemacs-initial-scratch-message`

用来自定义 scratch 缓冲区上显示的内容，可选的值有 nil 和一个字符串，就比如: "Welcome to Spacemacs!"

默认为 nil

- `dotspacemacs-themes`

用来设置 Spacemacs 的主题，Spacemacs 默认使用该列表中的第一个主题，可以按 `SPC T n` 来切换这个列表中的主题

> 如果使用的是 Emacs 的编辑模式，SPC(leader 键)应该是 `M-m` ！

- `dotspacemacs-mode-line-theme`

用来设置 Spacemacs 中 mode-line 的主题，就是 Spacemacs 下面的那根长条，可选的主题有

- spacemacs
- all-the-icons
- custom
- doom
- vim
- vanilla

custom 是用来自定义的主题，vanilla 是原生 Emacs 默认的主题，显示效果如下

> 显示效果会受主题和字体的影响

spacemacs: 

all-the-icons: 

doom: 

vim: 

vanilla: 

关于 custom 主题的自定义和 mode-line 的一些值的配置，这里就不过多赘述了，想要了解的话,可以参见官方文档

- `dosspacemacs-colorize-cursor-according-to-state`

可选的值有 t 和 nil，如果是 t 的话，光标的颜色会和 Spacemacs 中的状态的颜色一致，Spacemacs 中常用的状态一般有

- Emacs 蓝色
- Vim 正常模式 橙色
- Vim 插入模式 绿色

默认值为 t

- `dotspacemacs-default-font`

用来设置 Spacemacs 的字体， `:size` 可以用来调整字体的大小，如果值是一位小数的话，则根据字号来调整字体的大小，如果是整数的话，则根据像素多少来调整字体的大小，默认为 10.0，如果你想用整数的话，一般 14 是标准的大小，依据你使用的字体而定。

`:weight` 用来调整字体的字重， `:width` 用来调整字体的宽度， `:weight` 用来调整字体的高度

- `dotspacemacs-leader-key`

用来设置 leader 键，一般用于 Vim 和 Hybrid 编辑模式，leader 键主要用来自定义自己的快捷键。默认为 "SPC"

- `dotspacemacs-emacs-command-key`

用于在 Vim 和 Hybrid 编辑模式中，按下 leader 键后，用来代替 Emacs 中 `M-x` 的键，默认为"SPC"

- `dotspacemacs-ex-command-key`

用来设置 Vim 编辑模式下 Ex 命令使用的键，默认为 ":"

- `dotspacemacs-emacs-leader-key`

用来设置 Emacs 的编辑模式和 Vim 的插入模式下的 leader 键，默认为 "M-m"

- `dotspacemacs-major-mode-leader-key`

用来设置一个快捷键，相当于按下 `<leader 键> m` ，如 `SPC m` ，可以用来方便一些操作，默认为 ","

- `dotspacemacs-major-mode-emacs-leader-key`

使用 Vim 编辑模式或者 Hybrid 编辑模式后，在 Emacs 模式或者插入模式时下使用的 leader 键，这样的话使用 leader 键就不必再进入正常模式后再使用。如果是从终端使用 Spacemacs，默认是 `C-M-m` ，如果是以图形化界面启动，则默认是 `M-RET` （RET 就是回车键），另外， `M-RET` 在终端模式和图形化界面中都可以使用。

- `dotspacemacs-distinguish-gui-tab`

可选的值有 t 和 nil，如果是 nil，在图形化界面的 Spacemacs 中，命令可以被绑定到 C-i 、 `TAB` 、 `C-m` 、 `RET` 这些键中，如果是 t 则不能，默认为 nil

- `dotspacemacs-default-layout-name`

用来设置 Spacemacs 默认的布局的名称，默认值为"Default"

> 布局就是你打开的 buffer

- `dotspacemacs-display-default-layout`

在 Spacemacs 的 mode-line 中显示当前布局的名称，默认值为 nil

- `dotspacemacs-auto-resume-layouts`

可选的值有 t 和 nil，如果是 t 的话，下次启动 Spacemacs 时将会恢复你上次退出 Spacemacs 时的布局，即重新打开你上次打开的所有 buffer

- `dotspacemacs-auto-generate-layout-names`

可选的值有 t 和 nil，如果是 t，在创建新的 buffer 会自动生成名称，只有开启"按序号跳转 buffer"，这项功能才会生效

> 通常情况下，Spacemacs 的 mode-line 最左边会显示 buffer 的序号，可以通过按 `M-m` <序号> 来进行跳转

- `dotspacemacs-large-file-size`

用来设置打开的文件超过多大的大小时，Spacemacs 将会显示打开的文件的大小，单位为 MB，默认值为 1。

- `dotspacemacs-auto-save-file-location`

用来是设置自动保存的方式，可选的值有 original 和 cache，original 会直接自动保存文件，cache 会把文件自动保存到当前目录下的另一个文件名中，通常以 `filename~.***` 来命名，设置为 nil 可以禁用自动保存。默认值为 cache

- `dotspacemacs-max-rollback-slots`

TODO...

- `dotspacemacs-enable-paste-ransient-state`

TODO...

Day07 - 善用 leader 键

用 leader 键定义自己的快捷键

在 Spacemacs 中按下 leader 键后，等待一会应该会出现如下图的界面，其中包括 `user bindings` ，这是专门用来给用户定义快捷键的区域。如果你想要方便的执行一些函数，而这些函数又是 Spacemacs 本身不带有的或者不符合你口味的，你可以试试用这个来定义你自己的快捷键。

下面举两个例子

定义一个重启 Emacs 的快捷键

在 .spacemacs 文件中修改一些配置后，通常需要重启 Spacemacs 才会生效，但按 `C-x C-c` 退出 Spacemacs 再打开实在麻烦。 `restart-emacs` 可以快速重启 Emacs，且 Spacemacs 中已经定义好了相关的快捷键 `M-m q r` ，我们在这里使用 `restart-emacs` 命令来举个例子

```
lisp (global-set-key (kbd "<key>") '<function>)
```

其中 \ 是你要绑定的键，\ 是你希望执行的命令（函数），我希望把 `restart-emacs` 绑定到 `M-m o r` 中去，那么我可以这样写

```
lisp (global-set-key (kbd "M-m o r") 'restart-emacs)
```

然后就可以用 `M-m o r` 来重启 Spacemacs 了。同时，之前的快捷键也并没有失效

绑定一个键到 helm-swoop

如果添加了 ivy 这个 layer，那么你现在随意在某个文件按下 `C-s` 就可以进行更好的全文搜索，helm 中的 `helm-swoop` 同样可以实现这个功能，你可以按 `M-x helm-swoop` ，但是更好的方法是将它绑定一个键。ivy 可以直接按 `C-s` 进行全文搜索是因为 layer 中已经添加了键绑定相关的配置，我们可以手动为 `helm-swoop` 绑定一个键，方法同上

```
lisp (global-set-key (kbd "C-s") 'helm-swoop)
```

现在可以使用 `C-s` 调用 `helm-swoop` 进行全文搜索了

事实上，leader 键的内容远远不止这些，这还只是冰山一角，我不可能讲得面面俱到，否则这个教程也没有存在的意义。更多的内容，还需要你自己去探索

Day08 - Emacs Server

本文介绍了 Emacs Server 和通过 Emacs Server 快速启动 Emacs 的方法

Emacs Server 是什么？

Emacs Server 可以在你的内存中建立一个类似于服务器的东西（把你的配置加载到内存中），这样的你要是想使用 Emacs 就可以直接开启一个 Emacs 客户端来连接到这个服务器，而不需要打开 Emacs 后再加载一边配置，可以极大的提高 Emacs 的启动速度，通常情况下**一秒不到**便可启动！

使用教程

在终端中运行

```
shell emacs --daemon
```

接着便会开始加载配置，加载完成后，使用 `emacsclient` 命令可以直接连接到刚才加载的配置，但后面通常用带上文件名，如: `emacsclient test.sh` 。使用 `emacsclient -c` 可以直接以图形化界面运行 Spacemacs。演示如下:

除此之外，如果你在远程服务器上启动了 Emacs Server, 同时你的系统支持 SSH X11 转发（比如多数 Linux 发行版），你可以用如下命令在本机启动带图形界面的 Emacs Client 进行远程开发

```
shell ssh -Y user@server-address -f emacsclient --eval '"(make-frame-on-display \"$DISPLAY\")"'
```

下面这条命令可以让系统在开机时自动建立起一个 Emacs Server，这样的话就不必每次都执行 `emacs --daemon` 这条命令，可以直接使用 `emacsclient`

```
shell systemctl --user enable emacs
```

你还可以直接将 emacsclient 设置为终端的默认编辑器

```
shell export EDITOR='emacsclient -c'
```

.spacemacs 文件中同样有相关的配置，如果你使用的系统无法使用上面的命令来自动启动 Emacs Server，可以在 .spacemacs 修改下面的配置，这会在 Spacemacs 启动时自动开启一个 Emacs Server

```
lisp (setq-default dotspacemacs-enable-server t)
```

如果想要关闭已经开启的服务器，可以从 emacsclient 启动 Spacemacs，然后执行 `SPC q q` 来退出 Emacs 并关闭掉服务器

目前已知的问题

i. 正常情况下修改 .spacemacs 文件中的配置后，在 emacsclient 中并不会生效，而是要重新加载配置之后配置才会生效。严格来说这并不算是问题
ii. 一些关于字体的配置在 emacsclient 中无效，如单独设置中英文字体的配置
iii. Linux 下 emacsclient 无法使用 Fcitx 输入法，可以使用 pyim 或者 emacs-rime 解决
iv. 有关 Spacemacs 全屏启动的设置并不会在启动 emacsclient 时生效

Day09 - 编写代码

本文介绍了 Spacemacs 中有关于代码编写等方面的使用技巧和操作优化。

项目管理

Spacemacs 使用 Projectile 管理项目

在你打开一个文件的时候，Spacemacs 应该会自动这个文件所在的目录添加为你的项目，你可以在开始界面的 Project 一栏找到你的项目，也可以使用 `SPC p p` 来切换项目，使用 `SPC p f` 可以在项目里查找文件，你也可以使用 `M-x projectile-add-known-project` 来手动添加项目。Spacemacs 中所有的 Projectile 操作基本都在 `SPC p` 中，读者可以自行查看

Treeemacs 同样可以进行项目管理，但是它并不会自动把当前项目添加到侧边栏中，你需要手动添加： `M-x treemacs-add-project-to-workspacemacs`

Company-mode 的操作优化

Spacemacs 的 auto-completion layer 默认使用 Company-mode 作为补全后端

Spacemacs 中出现自动补全建议栏时，默认使用 Tab 补全代码中公共的部分和移动到下一个补全项，使用回车键来选中当前的项，如果你更习惯使用 Tab 来直接补全的话（即按 Tab 和回车都是直接补全自动完成列表的第一个项），你可以使用下面的代码：

```
lisp (setq-default dotspacemacs-configuration-layers '((auto-completion :variables auto-completion-tab-key-behavior 'complete)))
```

你还可以让完成列表根据用户的使用习惯来进行排序，不过可能会对其性能造成影响

```
lisp (setq-default dotspacemacs-configuration-layers '((auto-completion :variables auto-completion-enable-sort-by-usage t)))
```

如果在写代码时需要使用帮助文档，可以使用 `auto-completion-enable-help-tooltip`

```
lisp (setq-default dotspacemacs-configuration-layers '((auto-completion :variables auto-completion-enable-help-tooltip t))) ;; 设置为 t 可以在选中一个完成项时自动显示其文
```

打开文档后，你可以使用 C-M-v 来向下翻看文档， C-M-V 则为向上

使用 Company-box 可以让 Company 使用更加现代的外观和更好的文档显示支持（我的 KDE Plasma 桌面上默认的文档显示起来会很模糊）

```
lisp (setq-default dotspacemacs-configuration-layers '((auto-completion :variables auto-completion-use-company-box t)))
```

关于 Company-box 又有一大堆的说明，这里就不过多赘述了。另外，可以使用 SPC m l 来查看一个 layer 的帮助文档，关于 auto-completion 的其他技巧，大家可以自行探究

括号着色

Spacemacs 默认启动了括号着色的插件，如果你使用的是 spacemacs-dark、spacemacs-light 等主题，那么不同代码块对应的括号可能比较好区分，但对于一些主题来说，括号的颜色几乎相同，这样一来括号着色插件就成了累赘。如果你不喜欢括号着色的话，可以在 dotspacemacs-excluded-packages 中添加这两个包 `rainbow-delimiters` `highlight-parentheses` 来删除它们

Emacs 有一个自带的包来高亮括号，那就是 `show-paren-mode` ，但它只会在编辑器的光标处在括号上时才会生效，我们可以使用子龙山人的代码来使光标在括号内时高亮括号。将下面的代码添加到 user-config 中。

```
lisp (define-advice show-paren-function (:around (fn) fix-show-paren-function) "Highlight enclosing parens." (cond ((looking-at-p "\\s(") (funcall fn)) (t (save-ex
```

然后开启 `show-paren-mode` 即可

```
lisp (show-paren-mode 1)
```

打开代码长度基准线

一般情况下，一行代码的长度不宜超过 80 个字符，如果你希望在第 80 列显示一条竖线来提醒你，可以使用 `spacemacs/toggle-fill-column-indicator` SPC t f，如果需要全局启用，在 user-config 中添加下面的代码，需要注意的是，加入这行代码后，Spacemacs的开始界面也会显示这条竖线，可能看起来不是很美观

```
lisp (display-global-fill-column-indicator-mode 1)
```

标签栏

Emacs 的 buffer 切换不够直观、方便，可以使用 `tab-bar-mode` ，它会在 Emacs 顶部显示一个标签栏。Spacemacs 中默认已经安装了这个包

可以采取命令或者使用鼠标的方式来完成标签栏的切换，Spacemacs 默认并没有绑定相关的快捷键，大家可以自行绑定

- `tab-bar-switch-to-tab` 根据名字来切换标签
- `tab-bar-switch-to-recent-tab` 切换到最近的标签
- `tab-bar-switch-to-next-tab` 切换到下一个标签
- `tab-bar-close-tab` 关闭标签
- `tab-bar-new-tab` 新建标签

错误跳转

Emacs 有自己的错误跳转函数，兼容 flycheck、flymake

- next-error SPC c n 或] q 跳转到下一个错误
- previous-error SPC c N 或 [q 跳转到上一个错误

Spacemacs 默认使用 flycheck 进行语法检查，以下为 flycheck 的跳转函数

- flycheck-next-error
- flycheck-previous-error

默认的跳转快捷键可能有点麻烦，你可以自己绑定相关的键，如：

```
lisp (global-set-key (kbd "M-n") 'flycheck-next-error) (global-set-key (kbd "M-p") 'flycheck-previous-error)
```

如果想要显示错误列表，可以使用 M-x flycheck-error-list-mode

如果你想使用 flymake 代替 flycheck 的话，可以在 `dotspacemacs-excluded-packages` 里面加上 flycheck 的包 `flycheck` `flycheck-package` `flycheck-pos-tip` `flycheck-elsa` ，下次启动时便会删除这些包。然后在 user-config 中开启 flymake

```
lisp (flymake-mode 1)
```

快速运行代码

spacemacs 附带了 quickrun，可以直接编译运行代码，支持大部分的语言。直接使用 M-x quickrun 即可，或者 SPC x x 。使用 quickrun 运行的程序，会在10秒后自动关闭，可以通过设置 `quickrun-timeout-seconds` 来防止它关闭：

```
lisp (setq quickrun-timeout-seconds nil) ;; 注意，它可以设置为 nil，但不能设置为 0！
```

程序运行完之后，还会留下一个 buffer 不会自动关闭，可以按 C-g 将其关闭。

Emacs 的 `compile` 命令同样可以编译运行代码文件，不过可能需要你手动输入命令

终端支持

默认的终端有 eshell 和 shell，但两者都有些许一言难尽的地方，如果需要更好的终端支持，可以添加 shell 到你的 layer 中

```
lisp (setq-default dotspacemacs-configuration-layers '(((shell :variables shell-default-height 30 shell-default-position 'bottom))))
```

然后，SPC ' 即可在 Spacemacs 中打开系统的 shell。另外，SPC ！可以在 minibuffer 中临时执行 shell 命令

Day10 - Emacs 中的输入法

本文介绍了 Emacs 中的中文输入法

为什么要使用 Emacs 中的输入法？

i. 有一些 Linux 发行版（如 Arch Linux）下的 Emacs 一开始可能并不能使用系统的 Fcitx 输入法和 iBus 输入法
ii. Emacs 中的输入法可以和 Emacs 更好的兼容
iii. 输入命令、搜索文件、切换缓冲区时可以自动关闭输入法，在编写代码、文档、GTD 时可以根据上下文来自动切换中英文模式

如果你使用的 Linux 发行版不能使用系统的 Fcitx 输入法，一般可以使用 ArchWiki 中的办法解决，也可以使用 Spacemacs 中的 Chinese Layer 解决。使用方法可以参见其文档（SPC h l chinese RET）

Emacs 常用的输入法

常用的 Emacs 输入法是 pyim 和 emacs-rime

> chinese layer 中已经附带了 pyim

- pyim 是纯 elisp 实现的输入法，因此速度会比较慢，但是并不折腾，配置简单，不过它默认的词库很小，因此打起字来体验不是很好，而如果添加的词库较大的话，速度会减慢。
- emacs-rime 调用的是系统的 Rime 输入法，因此不会有卡顿的问题，同时使用起来和系统的 Rime 输入法几乎没有区别，可以直接使用系统中 Rime 输入法或 Fcitx 输入法的词库。但是要使用它，你得先在系统中安装 Rime 输入法，因为调用的是系统的输入法，所以配置的时候也得用 Rime 的配置文件来，因此你可能需要学习配置 Rime。

以上两个输入法都支持自动切换中英文，安装方法和配置方法可以参见这两个输入法的文档

emacs-rime 默认会在 minibuffer 显示候选框，这也是推荐的方式，如果你希望候选框跟随显示，可以配置 `rime-show-candidate`

```
lisp (setq rime-show-candidate 'minibuffer) ;; 可以使 emacs-rime 用 posframe 绘制候选框，可选的值有 nil、minibuffer、message、popup、posframe、sidewindow ;; 具体可参见文档
```

如果你不知道怎么配置 Rime 输入法，可以选择直接使用现成的 Rime 配置，如 Clover，这是一个很优秀的 Rime 输入方案，且内置了搜狗词库。emacs-rime 不会使用系统 Rime 输入法的配置文件，它有自己的配置文件，

可以通过 `M-x rime-open-configuration` 打开，安装好 Clover 之后，在 emacs-rime 的配置文件中添加下面的内容就可以在 emacs-rime 中使用 clover 输入方案了，其中 `"menu/page_size"` 用来设置候选词的个数

```
yaml patch: "menu/page_size": 7 schema_list: - schema: clover
```

如果是使用 pyim 的话，可以试试这个词库，在提供了较流行的词库的同时，防止 pyim 的速度过慢。

禁用系统输入法（针对 Linux 系统）

你可能不想在 Spacemacs 中使用系统的输入法（如 Fcitx 和 iBus），可以通过修改 .spacemacs.env 文件来进行禁用。

在你安装好 Spacemacs 之后，你应该可以发现主目录出现了一个 .spacemacs.env 文件，它用来单独设置 Spacemacs 的环境变量，在里面分别找到下面的变量

- GTK_IM_MODULE
- QT_IM_MODULE
- SDL_IM_MODULE
- XMODIFIERS=@im

然后将它们的值都设置为 und 即可，可能需要重启系统，如果你使用的 Emacs 本来就不可以使用系统的输入法，那么可以略过此步骤

2022-1-18 记：上面禁用系统输入法的时常会失效，现在并没有完美的方案

Day11 - Spacemacs 中的窗口操作和文件操作

分屏操作一直是 Emacs 相比与其他编辑器的一大优势，对分屏操作的良好支持可以使 Emacs 和使用者同时处理更多的信息，下面来介绍 Spacemacs 中的分屏操作。

> 首先需要注意的是，Spacemacs 中窗口操作的所有快捷键都在 `SPC w`

窗口操作

快速分屏

如果你使用的是 Emacs 的编辑风格，你应该对 `C-x 2`（上下分屏）和 `C-x 3`（左右分屏）再熟悉不过。你还可以通过 Leader 键来完成分屏操作，它们可能更好按一点

命令	快捷键	描述
split-window-below	`SPC w -`	上下分屏
split-window-right	`SPC w /`	左右分屏

而且，`-` 和 `/` 作为快捷键的一部分非常的直观

`SPC w` 后面跟上数字可以快速调整窗口的布局以实现快速分屏

命令	快捷键	描述
window-split-single-column	`SPC w 1`	调整为一个窗口
window-split-double-columns	`SPC w 2`	调整为左右两个窗口
window-split-triple-columns	`SPC w 3`	调整为横向排列的三个窗口
window-split-grid	`SPC w 4`	调整为 4x4 的窗口布局

快速切换到相应的窗口

无论你在 Spacemacs 中使用的是 spaceline 还是 doom-modeline，只要你分了多个屏，都可以在屏幕左下角（modeline最左边）看到一个数字，这个数字对应的是该窗口的序号，Emacs 中切换窗口通常会使用 `C-x o`，但窗口一多就不是很方便了。在 Spacemacs 中，可以直接按 `SPC` 窗口序号 来切换到对应的窗口

如果你是用的是 Vim 风格的快捷键，还可以通过 h j k l 来快速切换窗口

命令	快捷键	描述
evil-window-left	`SPC w h`	切换到左边的窗口
evil-window-down	`SPC w j`	切换到下面的窗口
evil-window-up	`SPC w k`	切换到上面的窗口
evil-window-right	`SPC w l`	切换到右边的窗口

如果只存在两个窗口，也可以使用 `SPC w w` 来切换到另一个窗口，效果和 `C-x o` 相同

快速删除窗口

Emacs 中可以使用 `C-x 0` 来删除一个窗口，Spacemacs 中也可以使用 `SPC w d` 来删除一个窗口，如果你希望删除窗口的同时删除对应的 buffer，可以使用 `kill-buffer-and-window` `SPC w x`

快速转换分屏类型

如果要将上下或左右分屏的窗口转换为另一个类型，可以使用 `window-layout-toggle` `SPC w +` 它可以位于焦点上的上下分屏转换为左右分屏，将左右分屏转换为上下分屏

文件操作

Spacemacs 中所有文件操作都可以通过按 `SPC f` 找到

查找文件

Emacs 快捷键的用户一般可以通过 `C-x C-f` 来查找文件

常用的涉及 leader 键的查找文件的操作如下表所示

命令	快捷键	描述
counsel-find-file	`SPC f f`	使用 counsel 来查找文件
find-file-literally	`SPC f l`	使用一个更轻量的工具来查找文件
counsel-rencentf	`SPC f r`	使用 counsel 来查找最近打开的文件

编辑文件

对于 Linux 用户，可能有使用 sudo 编辑文件的需要，此时可以使用 `sudo -E emacs <filename>` 来保留用户环境并加载当前用户（而不是 root 用户）的 Emacs 配置

常用的编辑文件的操作如下

命令	快捷键	描述
rename-current-buffer-file	`SPC f R`	重命名当前的文件
insert-file	`SPC f i`	将另一个文件中的内容插入到当前文件
save-buffer	`SPC f s`	保存当前的 buffer
save-as	`SPC f c`	另存为...文件
delete-current-buffer-file	`SPC f D`	删除当前 buffer 的文件

复制相关

这里的复制不是指文件的内容，而是与文件名、路径相关的，这些操作可以通过 `SPC f y` 找到

命令	快捷键	描述
copy-buffer-name	`SPC f y b`	复制当前 buffer 名
copy-file-name	`SPC f y n`	复制当前文件名
copy-file-path	`SPC f y y`	复制当前文件的路径

Day12 - 编辑模式的选择

本文介绍了 Spacemacs 中的 Emacs、Vim、Hybrid 三种键位。

Vim 键位

Leader 键

注：默认的 Leader 键是 SPC 空格键

如果你使用 Vim 键位的话，你可以通过输入冒号来在 Spacemacs 中使用类似于 Vim 中的各种命令，但 Spacemacs 的开发者可能更喜欢你使用 leader 键来完成 Vim 键位下的大多数特殊操作，因为大部分 Emacs 中的命令都已经与 leader 键相关的键绑定了。笔者也是最近才切换到 Vim 的键位，然后才明白了 Spacemacs 的精髓。冒号或者 leader 键，到底主要使用哪个，根据你的使用习惯来决定即可。

以下列出了大部分 leader 键后面跟一个键的快捷键，这些一般是比较常用的

快捷键	描述
`SPC SPC`	相当于 Emacs 编辑模式中的 `M-x`
`SPC TAB`	快速回到上一个 buffer
`SPC !`	用于临时执行 shell 命令
`SPC '`	启动系统的默认终端
`SPC *`	在项目的所有文件中搜索光标所在处的单词
`SPC /`	在项目的所有文件中搜索指定字符串（跟上一个功能一样）
`SPC 0`	把焦点转到 Treemacs 栏中（如果开启了 Treemacs 的话）
`SPC 1..9`	快速切换对应序号的窗口，范围为 1 至 9
`SPC ;`	快速注释当前行（按完后，还要再按一次；进行确认）
`SPC ?`	列出所有的快捷键
`SPC ` `	通过序号来切换窗口

常见的 Vim 下的操作都有了对应的快捷键

Vim 下的命令	Spacemacs 中的快捷键	描述
`:q`	`SPC q Q`	退出 Emacs
`:w`	`SPC f s`	保存当前的文件
`:split`	`SPC -`	上下分屏
`:vsplit`	`SPC /`	左右分屏
`:e`	`SPC f f`	查找文件并编辑
`/`	`SPC s s`	打开 swiper

如果你对 leader 键足够熟练的话，你可以尝试使用 leader 键来替代 `:`

更多的快捷键，可以自行探索（SPC ?）

退出到正常模式

你可以通过按 ESC 或 `C-[` 或 `f d` 来退出到正常模式中，但更为常见的做法时把 jj 或 jk 映射到 ESC 键，在 Spacemacs 中，你可以这样配置

```
lisp (setq evil-escape-key-sequence "jk") ;; 默认值为 fd
```

你还可以设置按下这个键后停留的时间，以用来给你足够的时间来按下第二个键

```
lisp (setq evil-escape-delay 0.5) ;; 默认为 0.1，单位为秒
```

Hybrid 键位

Hybrid，顾名思义——即混合编辑模式，这个模式与 Vim 键位相似，最大的区别是 Vim 下的插入模式换成了 Emacs 原生的编辑模式。打开一个文件时，默认是正常模式，然后按下 i 就会进入 Hybrid 模式，这个模式下可以正常使用 Emacs 下的快捷键。

这原本应该是极富效率的键位，因为相比于 Vim，Emacs 没有模式的概念，因此编辑文本的时候，移动光标非常方便，不需要总是按 Esc 键，而 Vim 在正常模式下的一些命令很强大，使用 Ctrl 和 Meta 键进行一些操作的时候，也可以按 Esc 键进入正常模式来进行一些更复杂的操作。我之前使用过一段时间这个键位，先来给出我的感受吧：对 Emacs 原生键位的用户来说，作用微乎其微，但对于习惯 Vim 的用户来说，可能会有所帮助。

对于 Emacs 原生键位用户

我比较喜欢 Emacs 原生的按键，使用 Hybrid 编辑模式的时候，光是每次编辑都要按一下 i 键都让我很不爽。我熟悉一些 Vim 的操作就比如：dd、caw、yy 等，但是它们都只能在正常模式下才能使用，而我习惯了 Emacs 的编辑模式，而 Emacs 也有一些类似的操作，我习惯直接使用 Emacs 的操作来完成，这就导致我无论如何也不会想着按一下 Esc 键，然后再输入命令。在有些时候，Emacs 和 Vim 中都有相关的操作，然后你刚好犯了选择困难症，不知道应该用哪一个。所以说，Hybrid 按键对 Emacs 原生按键绑定的用户来说，Hybrid 编辑模式可能并不能起到提升效率的作用，甚至还可能降低你的效率，不过如果你对编辑模式有着自己的理解，你也可以尝试使用它。

对于 Vim 键位用户

Hybrid 编辑模式对习惯使用 Vim 按键绑定的用户来说可能会更友好一些。使用 Vim 的时候，如果是在插入模式当中，要移动光标，你必须先退出到正常模式，然后才能移动光标。对部分人来说可能会造成不便。Hybrid 编辑模式正好解决了这一痛点，可以在插入模式中直接使用 `C-f`、`C-b`等快捷键来移动光标，不光是移动光标的快捷键，所有 Emacs 编辑模式下的快捷键都可以正常使用。如果你为 Vim 编辑模式的一些痒点所苦恼，那么可以试试这个编辑模式

建议

如果除了 Emacs，你还在使用其他编辑器或 IDE 的话，那么你没有必有太过依赖于 Hybrid 编辑模式，因为其他编辑器并没有它的按键绑定，可能会对日常使用其他编辑器或 IDE 造成不便，如果不会对你造成影响的话，那么你可以继续深度使用下去。

Emacs 键位

Emacs 编辑模式最大的优点就是可以时刻保持快速的编辑体验，但是因为**大部分**的快捷键都要用到修饰键（Ctrl 和 Meta 等），很多使用小拇指按 Ctrl 键的人都会感到小拇指痛，同时也可能会造成腕管综合征（俗称：鼠标手），对于这个问题，可以试试用手掌外侧或小拇指的尾骨来按压 Ctrl 键，也可以试试将大写锁定键（CapsLock）和 Ctrl 键对调，或者对调到其他你按着舒服的键。当然，最好的解决办法当然是：远离电脑！在使用电脑之余，适当的休息和体育锻炼并保持健康的身体才是持续 hacking Emacs 的不竭动力！

Day13 - Markdown 和 Org-mode

本文介绍 Spacemacs 中 Markdown 和 Org-mode 的相关操作，由于我并不经常使用 org-mode，所以只会点到为止，如果你是一名 org-mode 资深用户，有自己的使用经验的话，可以克隆这个仓库并创建拉取请求。

Markdown

Spacemacs 本身集成了一些加强 markdown 操作的包，如 `markdown-mode`，一般情况下，打开一个扩展名为 .md 或 .markdown 的文件便会自动进入 markdown-mode，一些常用的操作都可以通过按 `C-c` 来找到。这个按键对于使用 Emacs 键位的用户来说比较友好，如果你使用 Vim 或 Hybrid 键位的话，可以按 `SPC m` 或者 `,`，但是相应的操作相对于 `C-c` 较少。

Markdown 中可以使用 `来添加代码片段，通过这种方式添加的代码默认是没有语法高亮的，如果你需要语法高亮的话，可以添加 `markdown` 这个 layer。如果你对 markdown 编辑的需求比较强烈的话，是非常建议你添加这个 layer 的。这个 layer 除了提供语法高亮以外，还提供了一些额外的功能，如：markdown 实时预览、目录、emoji 符号补全等。添加完这个 layer 后，可以通过 `markdown-live-preview-mode`，`c P` 来对 markdown 文件进行实时预览。这个 mode 会在你每次保存文件的时候在旁边显示预览的内容，同时会在当前文件所处的目录中生成一个 html 文件。因为这个 mode 使用 eww 来渲染网页，所以预览内容的效果差强人意，可以按照这个 layer 附带的文档教程来美化成类似于 GitHub 的风格，但是所要用到的包已经不能正常安装了，所以并不推荐再去折腾这个了。

更多有关 `markdown` layer 的内容，可以通过 `SPC h l markdown RET` 来查看。

Org-mode

Spacemacs 本身对 Org-mode 的支持已经很完善了，如果你想得到更好的体验，可以添加 `org` layer。因为我并不熟练 org-mode，这些内容就留给读者自己探索吧……

中英文和表格对齐

注：此处的中英文对齐是指中英文 2:1 对齐，也就是两个西文字符的宽度正好对于一个中文符号的宽度。

大多数 Emacs 用户在 markdown 或 org-mode 中编辑表格时会发现同时包含英文和中文的表格并不能很好地对齐，会显得很凌乱，这是因为大多数 Emacs 用户使用的英文字体和中文字体并不是按照中英文对齐来设置的。如果你不使用表格，但也希望中英文能对齐的话，也可以看看这里的办法。

解决中英文对齐最简单的办法就是换一个支持中英文对齐的字体，例如很多人在使用的更纱黑体，或者 unifont。如果你希望全局使用这个字体的话，直接在 Spacemacs 的配置文件中把字体修改一下就可以了。也可以选择只把字体用来显示中文或者只用作显示表格中的字体。

unifont 本身包含中文的字符，并且中英文都是 2:1 等宽的，而且，**这个字体和大多数流行的等宽字体，例如：DejaVu Sans Mono、Source Code Pro、Fira Mono 也是能够做到中英文对齐的**，所以，你可以使用这个字体来单独显示中文。如果你默认的字体大小是 14 像素的话，把这个字体的大小设置为 16 像素后观感会好一些。

```
lisp (dolist (charset '(kana han cjk-misc bopomofo)) (set-fontset-font (frame-parameter nil 'font) charset (font-spec :family "unifont" :size 16)))
```

单独用作表格字体的话，可以参考懒猫的办法：https://manateelazycat.github.io/emacs/2020/04/02/org-font.html

Day14 - 学习还将继续

Spacemacs 14 Days 的教程已经接近了尾声，在我能力范围内能讲的内容也基本讲完了。但是，Emacs 和 Spacemacs 背后仍然有大量的内容等待大家去探索。到目前为止，你现在应该也具备了一定的自我学习能力，闲暇之余，你可以看看 Spacemacs 的文档。

> 需要注意的是，我给出的是 Spacemacs 开发版本的文档，你应该可以看到这个链接的域名是 https://develop.spacemacs.org，如果在搜索引擎中搜索的话，最靠前的结果大概是 https://www.spacemacs.org，后者是 Spacemacs 稳定版本的文档，但是已经年久失修了，所以我更建议你看开发版本的文档。

如果你在 Spacemacs 中遇到了一些使用上的问题，可以在论坛上发一个帖子来寻求帮助，例如：Emacs China，在编写帖子的时候，你需要准确的描述你所遇到的问题，这样别人才可以更好地帮助到你，另外，语气要相对友好一些。

GitHub 上的 Spacemacs 官方讨论区也是一个好去处，Spacemacs 的开发者会在上面积极回答你提出的问题，你同样需要用简洁的语言来准确描述出你所遇到的问题。而且，这个讨论区是英文的，所以你可能需要一定的英文功底。

Spacemacs 在 gitter 上有一个聊天室，你可以在上面用英文与其他的 Spacemacs 开发者和使用者聊天，也可以在上面向别人咨询一些问题。

如果你在使用 Spacemacs 的途中遇到了一个 Bug，你可以按 `SPC h I`来向 Spacemacs 的 GitHub 仓库提交一个 Issue，不过，你需要用英文详尽、准确的描述出问题所在——按照 Issue 中的模板，用英文填写出相关的内容即可。如果你英文不是很好的话，你可以使用一些在线的翻译器，不过最重要的是学好英语。

前面所讲的都是你作为一名学生，向别人寻求帮助或继续学习的途径，当你学得足够多，你还可以试着当一名老师，去回答别的初学者的问题。你可以先学习一下 Elisp，这里有一份入门的教程：https://learnxinyminutes.com/docs/zh-cn/elisp-cn/。或者看看 Elisp 的官方文档：https://www.gnu.org/software/emacs/manual/html_node/elisp/index.html。

Spacemacs 是一个开源的软件，它的持续发展离不开广大开源贡献者的努力，学习完 Elisp 后，你可以试着为 Spacemacs 做一些贡献，这里是一份关于如何为 Spacemacs 贡献的文

档：https://develop.spacemacs.org/CONTRIBUTING.html。在决定好做贡献之前，你需要认真地阅读这份文档！

贡献的方式主要有两种：编写 layer 和参与贡献 Spacemacs 的源代码。你可以参与改进已存在 layer 的代码，也可以自己创建一个全新的 layer。对于 layer 的编写，这里有一份详尽的文档：https://develop.spacemacs.org/doc/LAYERS.html。总之，不要认为自己技不如人便退缩，不管是谁，都是从菜鸟过来的，在不断的学习中你最终也一定可以写出令人刮目相看的代码。参与 Spacemacs 的开发并不是什么可怕的事情，你可以从提交最简单的一行代码开始。

愿你在学习的路上终有所成！