

Spacemacs帮助文档-中文版

目录

1. [Spacemacs帮助文档-中文版](#)

1. [目录](#)

2. [Quick start](#)

1. [1 Configuration layers](#)
2. [2 Dotfile \(.spacemacs\)](#)
3. [3 Dotdirectory \(~/.spacemacs.d\)](#)
4. [4 Learning Spacemacs](#)
 1. [4.1 Editing Styles](#)
 2. [4.2 The leader keys](#)
 3. [4.3 Evil-tutor](#)
 4. [4.4 Universal argument](#)
 5. [4.5 Configuration layers and Package discovery](#)
 6. [4.6 Key bindings discovery](#)
 7. [4.7 Describe functions](#)

5. [5 How-To's](#)

3. [Core Pillars](#)

1. [1.1 Mnemonic](#)
2. [1.2 Discoverable](#)
3. [1.3 Consistent](#)
4. [1.4 Crowd-configured](#)
5. [2 Highlighted feature](#)
6. [3 Screenshots](#)
7. [4 Who can benefit from this?](#)
8. [5 Update and Rollback](#)

1. [5.1 Update Spacemacs repository](#)

1. [5.1.1 Automatic Updates](#)
2. [5.1.2 Updating from the Spacemacs Buffer](#)
3. [5.1.3 Updating Manually with git](#)

9. [5.2 Update packages](#)

4. [6 Configuration layers](#)

1. [6.1 Purpose](#)
2. [6.2 Structure](#)
3. [6.3 Configure packages](#)
 1. [With a layer](#)
 1. [6.3.1.1 Declaration](#)
 2. [6.3.1.2 Initialization](#)
 3. [6.3.1.3 Exclusion](#)
 2. [6.3.2 Without a layer](#)
4. [6.4 Packages synchronization](#)
5. [6.5 Types of configuration layers](#)
6. [6.6 Submitting a configuration layer upstream](#)

7. 6.7 Example: Themes Megapack example
8. 6.8 Managing private configuration layers
 1. 6.8.1 Using the private directory
 2. 6.8.2 Using an external Git repository
 3. 6.8.3 Using a personal branch
9. 6.9 Tips for writing layers
5. 7 Dotfile Configuration
 1. 7.1 Dotfile Installation
 2. 7.2 Alternative dotdirectory
 3. 7.3 Synchronization of dotfile changes
 4. 7.4 Testing the dotfile
 5. 7.5 Dotfile Contents
 1. 7.5.1 Configuration functions
 2. 7.5.2自定义变量
 6. 7.6 Declaring Configuration layers
 1. 7.6.1 Setting configuration layers variables
 2. 7.6.2 Disabling layer services in other layers
 3. 7.6.3 Selecting/Ignoring packages of a layer
 4. 7.6.4 Excluding packages
6. 8 Concepts
 1. 8.1 Editing Styles
 1. 8.1.1Vim
 2. 8.1.2 Emacs
 3. 8.1.3 Hybrid
 2. 8.2 States
 3. 8.3 Evilified modes
 4. 8.4 Evil leader
 5. 8.5 Universal argument
 6. 8.6 Transient-states
7. 9 Differences between Vim, Evil and Spacemacs
 1. 9.1 The vim-surround case
8. 10 Evil plugins
9. 11 Binding keys
10. 12 GUI Elements
 11. 12.1 Color themes
 12. 12.2 Font
 13. 12.3 GUI Toggles
 1. 12.3.1 Global line numbers
 14. 12.4 Mode-line
 1. 12.4.1 Powerline font installation for terminal-mode users
 2. 12.4.2 Flycheck integration
 3. 12.4.3 Anzu integration
 4. 12.4.4 Battery status integration
 5. 12.4.5 Powerline separators
 6. 12.4.6 Minor Modes
 7. 12.4.7 Customizing the mode-line

15. [13 Layouts](#)
16. [13.1 The default layout](#)
17. [13.2 Project layouts](#)
18. [13.3 Custom Layouts](#)
19. [13.4 Save/Load layouts into a file](#)
20. [13.5 Layout key bindings](#)
21. [13.6 Workspaces](#)
22. [13.7 Workspace key bindings](#)
23. [14 Commands](#)
24. [14.1 Vim key bindings](#)
 1. [14.1.1 Escaping](#)
 2. [14.1.2 Executing Vim and Emacs ex/M-x commands](#)
 3. [14.1.3 Leader key](#)
 4. [14.1.4 Additional text objects](#)
25. [14.2为用户保留前缀命令](#)
26. [14.3 Completion](#)
 1. [14.3.1 Helm](#)
 2. [14.3.1.1 C-z and Tab switch](#)
 3. [14.3.1.2 Helm focus](#)
 4. [14.3.1.3 Helm transient state](#)
27. [14.4 Discovering](#)
 1. [14.4.1 Key bindings](#)
 1. [14.4.1.1 Which-key](#)
 2. [14.4.1.2 Helm describe key bindings](#)
 2. [14.4.2 Getting help](#)
 3. [14.4.3 Available layers](#)
 1. [14.4.3.1 Available packages in Spacemacs](#)
 2. [14.4.3.2 New packages from ELPA repositories](#)
 4. [14.4.4 Toggles](#)
28. [14.5 Navigating](#)
 1. [14.5.1 Point/Cursor](#)
 1. [14.5.1.1 Smooth scrolling](#)
 2. [14.5.2 Vim motions with avy](#)
 1. [14.5.2.1 ace-link mode](#)
 3. [14.5.3 Unimpaired bindings](#)
 4. [14.5.4 Jumping, Joining and Splitting](#)
 1. [14.5.4.1 Jumping](#)
 2. [14.5.4.2 Joining and splitting](#)
 5. [14.5.5 Window manipulation](#)
 1. [14.5.5.1 Window manipulation key bindings](#)
 2. [14.5.5.2 Window manipulation transient state](#)
 3. [14.5.5.3 Golden ratio](#)
 6. [14.5.6 Buffers and Files](#)
 1. [14.5.6.1 Buffers manipulation key bindings](#)
 2. [14.5.6.2 Buffers manipulation transient state](#)
 3. [14.5.6.3 Special Buffers](#)

4. [14.5.6.4 Files manipulations key bindings](#)
 5. [14.5.6.5 Emacs and Spacemacs files](#)
 6. [14.5.6.6 Browsing files with Helm](#)
 7. [14.5.7 Ido](#)
 8. [14.5.8 Ido transient state](#)
 9. [14.5.9 NeoTree file tree](#)
 1. [14.5.9.1 NeoTree navigation](#)
 2. [14.5.9.2 Opening files with NeoTree](#)
 3. [l or # RET |open file in window number](#)
 4. [14.5.9.3 Other NeoTree key bindings](#)
 5. [14.5.9.4 NeoTree mode-line](#)
 6. [14.5.9.5 NeoTree Source Control Integration](#)
 7. [14.5.9.6 NeoTree Theme](#)
 10. [14.5.10 Bookmarks](#)
 11. [14.5.11 DocView mode](#)
29. [14.6 Auto-saving](#)
 1. [14.6.1 Frequency of auto-saving](#)
 2. [14.6.2 Location of auto-saved files](#)
 3. [14.6.3 Disable auto-save](#)
30. [14.7 Searching](#)
 1. [14.7.1 With an external tool](#)
 1. [14.7.1.1 Useful key bindings](#)
 2. [14.7.1.2 Searching in current file](#)
 3. [14.7.1.3 Searching in all open buffers visiting files](#)
 4. [14.7.1.4 Searching in files in an arbitrary directory](#)
 5. [14.7.1.5 Searching in a project](#)
 6. [14.7.1.6 Searching the web](#)
 2. [14.7.2 Persistent highlighting](#)
 3. [14.7.3 Highlight current symbol](#)
 4. [14.7.4 Visual Star](#)
 5. [14.7.5按语义列出符号](#)
 6. [14.7.6 Helm-swoop](#)
31. [14.8编辑](#)
 1. [14.8.1粘贴文本](#)
 1. [14.8.1.1粘贴瞬态](#)
 2. [14.8.1.2自动缩进粘贴文本](#)
 2. [14.8.2文本操作命令](#)
 3. [14.8.3文本插入命令](#)
 4. [14.8.4 Smartparens Strict mode](#)
 5. [14.8.5 缩放\(Zooming\)](#)
 1. [14.8.5.1 文本](#)
 2. [14.8.5.2 框架\(Frame\)](#)
 6. [14.8.6增加/减少数字](#)
 7. [14.8.7 拼写检查](#)
 8. [14.8.8区域选择](#)
 1. [14.8.8.1 扩大区域\(Expand-region\)](#)

- 2. 14.8.8.2 缩进文本对象(Indent text object)
- 9. 14.8.9 区域缩小(Region narrowing)
- 10. 14.8.10 用iedit替换文本(Replacing text with iedit)
- 11. 14.8.10.1 iedit states 键绑定(iedit states key bindings)
- 12. prefix all occurrences with an increasing number (SPC u to choose the starting number).
- 13. 14.8.10.2 例子
- 14. 14.8.11 替换几个文件中的文本
- 15. 14.8.12 重命名目录中的文件
- 16. 14.8.13 评论
- 17. 14.8.14 正则表达式
- 18. 14.8.15 删除文件
- 19. 14.8.16 编辑 lisp 代码
- 20. 14.8.16.1 lisp 键绑定
- 21. 14.8.17 鼠标使用情况
- 32. 14.9 管理项目
- 33. 14.10 寄存器(Registers)
- 34. 14.11 错误处理
- 35. 14.12 编译
- 36. 14.13 模式(Modes) 1. 4.13.1 主模式 leader 键(Major Mode leader key) 2. 14.13.2 Helm
- 37. 14.14 emacs 服务器 1. 14.14.1 连接到 emacs 服务器
- 38. 14.15 保持服务器开着
- 39. 14.16 故障排除(Troubleshoot) 1. 14.16.1 Loading fails 2. 14.16.2 升级/降级 Emacs 版本

Quick start

1 Configuration layers

spacemacs将其配置分为独立的单元（称为***configuration layers***(配置层)）。这些层堆叠在一起，实现自定义配置。

默认情况下，spacemacs 使用称为 `~/ .spacemacs` 的点文件来控制要加载的层。在这个文件中你也可以配置某些功能。

配置层是一个至少包含一个 `packages.el` 文件的目录，该文件使用 emacs 的 `package.el` 内置特性来定义和配置要从 emacs 包存储库下载的包。

如果您已经拥有自己的 emacs 配置，则可以将其移至自己的 layer。

以下命令在专用目录中创建一个图层：

```
SPC : configuration-layer/create-layer RET
```

你创建的任何配置层必须显式加载到 `~/ .spacemacs` 中。

注意：为了您的隐私，私人目录的内容不受源代码管理。请参阅[文档](#)中有关专用配置管理的部分。

2 Dotfile (.spacemacs)

如上所述 .spacemacs 控制要加载的配置层，也是自定义 spacemacs 的一种手段

以下命令将在您的主目录中创建一个 .spacemacs 文件：

```
SPC : dotspacemacs/install RET
```

打开已安装的 dotfile 文件：

```
SPC f e d
```

使用变量 dotspacemacs-configuration-layers 加载一些配置层

```
;; List of configuration layers to load.
dotspacemacs-configuration-layers '(auto-completion smex)
```

一些配置层支持配置变量来暴露对层特定功能的粒度控制，git层就是这样一个例子。变量可以直接在 dotspacemacs-configuration-layers 中设置，如下所示：

```
;; List of configuration layers to load.
dotspacemacs-configuration-layers '(auto-completion
                                     (git :variables
                                         git-magit-status-fullscreen t)
                                     smex)
```

在任何时候，您都可以通过按 `spc f e r` 来应用对点文件或图层所作的更改，而无需重新启动 spacemacs。

dotfile 模板包含有关如何自定义 spacemacs 的更多信息。有关更多详细信息，请参阅文档的 dotfile 配置部分。

3 Dotdirectory (~/.spacemacs.d)

像 emacs 一样，spacemacs 的初始化也可以被包含在特殊目录 ~/.spacemacs.d 中的 init.el 文件中。dotfile 的内容应该被复制到 init.el 文件中。

emacs dotfile 或 dotdirectory 不会被替换，而是被 spacemacs dotfile 或 dotdirectory 所补充。在启动过程中，emacs 仍然使用 ~/.emacs.d/init.el（或 ~/.emacs）进行初始化，而 user-emacs-directory 仍然会指向 ~/.emacs.d/，即使 ~/.spacemacs.d 或 ~/.spacemacs 存在。只有现在 ~/.emacs.d/init.el 是由 spacemacs 提供的（例如在将 spacemacs git repo 克隆到一个空的 ~/.emacs.d/ 之后），并且你自己的个人配置进入 ~/.spacemacs.d/init.el（或 ~/.spacemacs）。

有一个简单的解决方法，以维护（您的以前）原生 emacs 和（你的新的）spacemacs 配置并排，而不需要重命名和备份 ~/.emacs.d/。

4 Learning Spacemacs

4.1 Editing Styles

spacemacs 可以被 vim 用户或 emacs 用户使用，方法是在 dotfile `~/ .spacemacs` 中将 `dotspacemacs-editing-style` 变量设置为 vim，emacs 甚至是 hybrid。

4.2 The leader keys

spacemacs 键绑定使用默认绑定到 vim 的 `spc`（空格键）或混合编辑样式的绑定键和 emacs 样式的 `M-m`。

如果使用 vim 风格或者使用 emacs 风格（这些变量必须在文件 `~/ .spacemacs` 中设置），则可以通过设置变量 `dotspacemacs-leader-key` 来更改它。

为了简单起见，文档始终将 leader 称为 `spc`。

在默认情况下设置，为辅助的 leader key 被称为 major-mode 的 leader key。此键是所有主模式特定命令都绑定的 `spc m` 的快捷方式。

4.3 Evil-tutor

如果你愿意学习 vim 的键绑定（强烈推荐，因为你可以从 emacs 风格中获益），按 `spc h t` 开始一个 Evil-adapted vimtutor。

4.4 Universal argument

在 vim 编辑风格中，通用参数默认为 `spc u` 而不是 `c-u`，因为后者用于在 vim 中向上滚动。

4.5 Configuration layers and Package discovery

通过使用 `helm-spacemacs-help` 与 `spc h spc`，您可以快速搜索包并获取使用它的图层的名称。你也可以轻松地去看一个图层的 `readme.org` 或去看一个包的初始化函数。

4.6 Key bindings discovery

由于 [which-key](#)，每当一个前缀命令被按下（如 `SPC`）一秒钟后出现一个缓冲区列出了这个前缀可能的键。

也可以通过按下来搜索特定的键绑定：

```
SPC ?
```

要将绑定列表缩小到以 `spc` 为前缀的那些列表，请键入如下正则表达式的模式：

```
SPC\ b
```

这将列出所有缓冲区相关的绑定。注意：你在 `helm-descbind` 提示符下，模式由6个字母组成：大写的 `spc`，反斜线，实际的空格和小写的 `b`。

4.7 Describe functions

Describe functions 是强大的 emacs introspection 命令来获取有关函数，变量，模式等信息，因此这些命令是绑定的：

Key Binding	Description
SPC h d f	describe-function
SPC h d k	describe-key
SPC h d m	describe-mode
SPC h d v	describe-variable

5 How-To's

一些快速的方法是编写在[FAQ.org](https://faq.org)文件中。

Core Pillars

四个核心支柱：Mnemonic (助记符)，Discoverable (可发现)，Consistent (一致) 和 "Crowd-Configured" (多人配置)。

如果这些核心支柱中的任何一个遭到违反，我们将尽最大努力解决这个问题。

1.1 Mnemonic

键绑定使用助记符前缀如b缓冲区，p为项目，s为搜索，h为帮助等

1.2 Discoverable

创新的实时显示可用的快捷键绑定。简单的查询系统来快速找到可用的layers，packages等等。

1.3 Consistent

由于有一套明确定义的约定，类似的功能在每个地方都具有相同的快捷键绑定。文档对于随spacemacs提供的任何layer都是必需的。

1.4 Crowd-configured

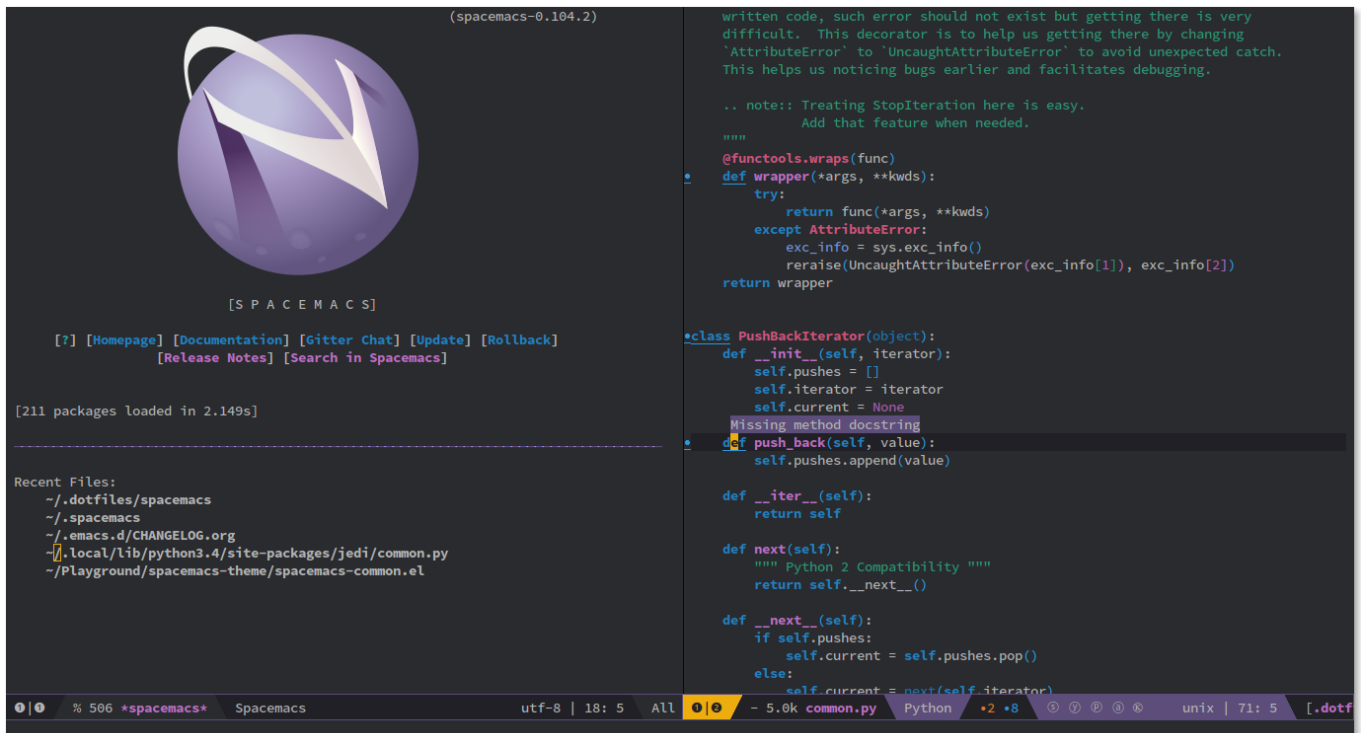
社区驱动的配置提供了由高级用户调整的策划包，并且错误被迅速修复。

2 Highlighted feature

- 将模态编辑的效率带到功能强大的emacs lisp平台。modal ux是可选的，spacemacs只能与emacs快捷键绑定一起使用。
- 与 Evil 状态（VIM模式）很好地融合。
- 保持你的手指放在首页，快速编辑支持 QWERTY 和 BEPO 布局
- 简约和漂亮的图形用户界面保持您的可用屏幕空间重要的是：您的文本文件。
- 快速的启动时间：尽可能的减少软件包和配置的延迟
- 通过大量使用空格键而不是修饰符来降低 RSI 的风险。如果你的拇指有问题，你仍然可以使用修饰符来使用 spacemacs。
- 轻松提供您的改进和新的配置层。

- 在 Gitter 和 IRC 上非常活跃和有帮助的社区（通过 Gitter IRC bridge）。

3 Screenshots



Python

注意：即使屏幕截图经常更新，spacemacs也在迅速发展，屏幕截图可能并不完全反映项目的当前状态。

4 Who can benefit from this?

- 最初，vim 用户希望通过使用 emacs 来使用 spacemacs（请参阅 Vimmers 指南）。但通过选择 emacs 编辑风格，现在非 vim 用户可以很好地使用它。
- 对于希望降低由缺省 emacs 键位绑定引发的 RSI 风险的人来说也是一个很好的选择。（这是一个假设，有没有官方的研究证明这一点！）如果你有你的拇指问题，你仍然可以使用 emacs 编辑风格，把领导键放在一个修改组合。
- emacs 用户希望学习不同的编辑文件的方式，或者想要学习 vim 键绑定，甚至希望通过将其样式设置为混合来混合两种编辑样式。
- emacs 用户希望有一个简单而深入的配置系统，大大降低 .emacs 崩溃的风险
- 对编程用户感谢支持动态切换编辑风格。vim 用户和 emacs 用户可以舒适地使用相同的 spacemacs。

5 Update and Rollback

5.1 Update Spacemacs repository

有几种方法可以更新 spacemacs 的核心文件和 layer 信息。建议先更新软件包;看下一节。

5.1.1 Automatic Updates

每次启动时，spacemacs将自动检查新版本。当它检测到新版本可用时，模式行中将出现一个箭头。点击它来更新spacemacs。更新后必须重新启动emacs。



Update Button

注意：如果你使用spacemacs的develop分支，自动更新被禁用 - 你必须使用git手动更新。

5.1.2 Updating from the Spacemacs Buffer

使用 spacemacs buffer 中标记为“更新 spacemacs ”的按钮。系统会提示您输入要使用的版本。

注意：如果你使用 spacemacs 的 develop 分支，你不能使用这个方法。

5.1.3 Updating Manually with git

手动更新关闭emacs并更新git存储库：

```
$ git pull origin master
```

注意：主分支被认为是不可变的，因为你不能通过添加你自己的提交来修改它。如果你这样做，你会打破主分支上的spacemacs的自动更新。要fork spacemacs代码，您必须使用您手动管理的自定义分支。

5.2 Update packages

更新 spacemacs 使用的 emacs 软件包按回车键（回车）或点击横幅下的启动页面中的链接 [Update Packages]，然后重新启动 emacs。如果你愿意，你可以使用命令 `configuration-layer / update-packages` 而不是按钮。

如果出现任何问题，您应该能够通过按下 RET 或单击启动页面中的 [rollback package update] 链接并选择一个回滚槽（按日期排序）来回滚更新。这个按钮使用命令 `configuration-layer/rollback`

6 Configuration layers

本部分是 layer 的概述。可以在这里找到更多关于编写配置 layers 的介绍（推荐阅读！）。

6.1 Purpose

layers 有助于收集相关的软件包以提供功能。例如，python 层为 python 文件提供自动完成，语法检查和 REPL 支持。这种方法有助于保持配置的组织性，并减少用户的开销，使他们不必考虑要安装哪些软件包。要安装所有的 python 功能，用户只需要将 python 图层添加到他们的 dotfile 文件中即可。

6.2 Structure

配置按层组织。每层都有以下结构：

```
[layer_name]
|__ [local]
| |__ [package 1]
```

```
| | ...
| |__ [package n]
|-- layers.el
|__ packages.el
|__ funcs.el
|__ config.el
|__ keybindings.el

[] = directory
```

Where:

文件	用法
layers.el	声明额外图层的地方
packages.el	packages 列表及其配置函数（init，post-init等）
funcs.el	在 layer 中定义的所有功能（例如在 package 配置中使用）
config.el	Layer 配置（定义 layer 变量默认值并设置一些配置变量）
keybindings.el	一般的 key 绑定不绑定到特定的 package 配置

Packages 可以是：

- 从符合 ELPA 的仓库安装的 ELPA packages
- 本地 packages 在一个 layer 的本地文件夹中
- 使用 [quelpa](#) 从在线来源安装。

6.3 Configure packages

With a layer

6.3.1.1Declaration

Packages 在一个名为 -packages 的变量中声明，其中是 layer 的名称。

Example:

```
(setq <layer>-packages '(package1 package2 ...))
```

来自所有 layers 的所有 packages 都按字母顺序处理，所以有时您必须使用一些加载后的黑魔法来正确配置它们。例如，如果包 a 取决于 b，那么你可以配置一个：

```
(with-eval-after-load 'B ...)
```

有关使用 `quelpa` 或本地软件包安装软件包的详细信息，请参阅 [LAYERS](#)。

6.3.1.2 Initialization

要初始化一个包 `xxx`，用这个格式在 `packages.el` 中定义一个函数：

```
(defun <layer>/init-xxx () ...body )
```

用 `use-package` 宏定义 `body` 是很常见的。

6.3.1.3 Exclusion

有可能在每层的基础上从 `spacemacs` 中排除一些软件包。这在配置 `layer` 旨在替换 `spacemacs layer` 中声明的库存包时非常有用。

要这样做，请将要排除的软件包名称添加到变量 `-excluded-packages` 中

Example:

```
(setq <layer>-excluded-packages '(package1 package2 ...))
```

6.3.2 Without a layer

有时一个 `layer` 可能是一个不必要的开销，如果你只想安装一个与其关联的配置很少的包，就是这种情况。一个很好的例子是一些你只对语法高亮感兴趣的小众语言。

您可以通过将它们添加到 `dotfile` 中的 `dotspacemacs/layers` 函数下的变量 `dotspacemacs-additional-packages` 来安装此类包。

例如，安装 `llvm-mode` 和 `dts-mode`：

```
(defun dotspacemacs/layers ()
  "Configuration Layers declaration..."
  (setq-default
    ;; ...
    dotspacemacs-additional-packages '(llvm-mode dts-mode)
    ;; ...
  ))
```

如果您想为它们添加一些配置，请将配置放置在 `dotspacemacs/user-config` 函数中，或者考虑创建一个 `layer`。

6.4 Packages synchronization

spacemacs 将只安装用户明确使用的 packages。如果使用该 layer（即，在 dotspacemacs-configuration-layers 中列出），则认为该 package 被使用。任何未使用的 package 都被认为是孤立的，并在下次启动 emacs 时被删除。

6.5 Types of configuration layers

有两种类型的配置 layers：

- 分布式 layers（在 layers 目录中，这些 layers 是社区共享的贡献，并在上游合并）
- 私人（在私人目录中，他们被git忽略）

6.6 Submitting a configuration layer upstream

如果您决定提供配置 layer，请首先查看[贡献指南](#)（现在404）。

6.7 Example: Themes Megapack example

这是一个简单的配置 layer，列出了一些你可以在[这里](#)找到的主题。（这里也404）

要安装它，只需添加主题 megapack 到你的 ~/.spacemacs 就好：

```
(setq-default dotspacemacs-configuration-layers '(themes-megapack))
```

添加这个 layer 将安装大约100个主题;卸载它们，从 dotspacemacs-configuration-layers 中删除该层，然后按 `spc f e r`。

6.8 Managing private configuration layers

spacemacs的配置系统足够灵活，让你以不同的方式管理你的私人 layers。

6.8.1 Using the private directory

私人目录中的所有内容都被git忽略，所以它是存储私人 layers 的好地方。这种方法有一个巨大的缺点：你的 layers 不受源代码控制。

6.8.2 Using an external Git repository

这是管理你的私人 layers 的推荐方式。

最好的方法是把你所有的私有 layers 存储到外部的 git 仓库中。如果有的话，将它们存储在 dotfiles 存储库中是一个很好的做法。还把你的 ~/.spacemacs 文件放在里面。

那么你可以自由地将你的 layer 符号链接到 ~/.emacs.d/private 或者让它们在你想要的任何地方，并且在 ~/.spacemacs 的变量 dotspacemacs-configuration-layer-path 中引用父目录。

请注意，您也可以为所有私有 layer 设置专用存储库，然后直接在 ~/.emacs.d/private 中克隆该存储库。

6.8.3 Using a personal branch

管理你的私人 layers 的最后一个主要方法是把他们推到一个你跟上游 master 或者 develop 分支保持同步的个人分支。

6.9 Tips for writing layers

请参考这个介绍，了解一些关于写 layers 的技巧，以及如何最好地使它们符合 spacemacs 理念和加载策略。

7 Dotfile Configuration

用户配置可以存储在你的 `~/ .spacemacs` 文件中。

7.1 Dotfile Installation

spacemacs 第一次启动，它会问你几个问题，然后在 *HOME* 中安装 .spacemacs。

7.2 Alternative dotdirectory

一个 dotdirectory `~/ .spacemacs.d /` 可以用来代替 dotfile。如果你想使用这个选项，把 `~/ .spacemacs` 移动到 `~/ .spacemacs.d/init.el`。

也可以使用环境变量 `SPACEMACSDIR` 来覆盖 `~/ .spacemacs.d/` 的位置。当然你也可以使用符号链接来改变这个目录的位置。

注意： `~/ .spacemacs` 将总是优先于 `~/ .spacemacs.d/init.el`，所以 `~/ .spacemacs` 不能存在于 `~/ .spacemacs.d/init.el` 以供 spacemacs 使用

7.3 Synchronization of dotfile changes

要应用在 `~/ .spacemacs` 中进行的修改，请按 `spc f e r`。它将重新执行 spacemacs 初始化过程。

注意：同步将重新执行函数 `dotspacemacs/init`，`dotspacemacs/user-init` 和 `dotspacemacs/user-config`，根据这个功能的内容，你可能会遇到一些不需要的副作用。例如，如果在 `dotspacemacs/user-config` 中使用切换来启用某些行为，则只要 dotfile 重新同步，就会关闭此行为。为了避免这些副作用，建议使用 `setq` 表达式而不是切换功能，或者使用 `on` 或 `off` 版本来代替 (`spacemacs/toggle-`，使用 `spacemacs/toggle-` 或者 `spacemacs/toggle--off`)。

可以使用通用参数 (`spc u spc f e r`) 跳过执行 `dotspacemacs/user-config`。

7.4 Testing the dotfile

你可以使用命令 `SPC SPC dotspacemacs/test-dotfile` 来检查你的 `~/ .spacemacs` 是否正确。除此之外，这将检查是否可以找到已声明的 layers，以及变量是否具有合理的值。这些测试也会在与 `SPC f e r` 同步时自动运行。

7.5 Dotfile Contents

7.5.1 Configuration functions

`~/ .spacemacs` 文件中的三个特殊函数可用于在 spacemacs 加载过程的开始和结束时执行配置：

- `dotspacemacs/layers` 在 spacemacs 初始化的启动时被调用，这是您设置 spacemacs 分配和声明 layers 以在您的配置中使用的位置。你也可以添加或排除你选择的包，并调整一些 spacemacs 加载行为。

- dotspacemacs/init 在层配置之前的 spacemacs 初始化启动时被调用。除了修改前缀为 spacespacemacs- 的 spacemacs 变量值之外，不应该在其中放置任何用户代码。
- 在进行 layers 配置之前，在 spacespacemacs/init 之后立即调用 dotspacemacs/user-init。此函数对于在加载包之前需要设置的变量非常有用。
- dotspacemacs/user-config 在层配置后的 spacemacs 初始化的最后被调用。这是大部分配置应该完成的地方。除非明确指定应在加载包之前设置变量，否则应将代码放在此处。

7.5.2 自定义变量

自定义变量配置从 M-x 自定义组内置的 emacs 功能会被 emacs 自动保存在 ~/.spacemacs 文件的末尾。

7.6 Declaring Configuration layers

要使用配置层，通过将它添加到 ~/.spacemacs 的 dotspacemacs-configuration-layers 变量中，将它声明在 dotfile 中。

注意：在这个文档中，一个 used layer 相当于一个 declared layer。

例如，RMS 可以像这样添加他的私有配置 layer：

```
(setq-default dotspacemacs-configuration-layers
  '(
    ;; other layers
    ;; rms layer added at the end of the list
    rms
  ))
```

带有 spacemacs 的官方 layers 存储在 ~/.emacs.d/layers 中。目录 ~/.emacs.d/private 是你的私人 layers 的一个插入位置。如果您告诉 spacemacs 在哪里查找它们，可以将 layers 放在您选择的位置。这是通过在 ~/.spacemacs 中设置列表 dotspacemacs-configuration-layer-path 完成的。例如在 ~/.myconfig 中添加一些 layers，像这样设置变量：

```
(setq-default dotspacemacs-configuration-layer-path '("~/myconfig/"))
```

7.6.1 Setting configuration layers variables

一些配置 layers 具有配置变量来启用特定功能。比如 git layer 有几个配置变量，可以像这样直接在 dotspacemacs-configuration-layers 中设置：

```
(defun dotspacemacs/layers ()
  ;; List of configuration layers to load.
  (setq-default dotspacemacs-configuration-layers
    '(auto-completion
      (git :variables
          git-magit-status-fullscreen t
```

```
git-variable-example nil)
smex)))
```

:variables 关键字可以方便地保持 layer 配置接近他们的声明。在 dotfile 的 dotspacemacs/user-init 函数中设置 layer 变量也是配置 layer 的完美方法。

7.6.2 Disabling layer services in other layers

通常 layers 可以启用其他 layers 可以使用的服务。例如，如果使用 layers auto-completion，则每个支持 auto-completion 的其他 layers 都将启用此功能。

有时你可能想要禁用某些特定 layers 中的某个 layers 添加的服务。假设你想在 org 和 git layers 中禁用 auto-completion，你可以用下面的 layer 声明来完成。

```
(defun dotspacemacs/layers ()
  ;; List of configuration layers to load.
  (setq-default dotspacemacs-configuration-layers
    '(org git
      (auto-completion :disabled-for org git))))
```

您还可以使用 :enabled-for 构造对除显式标识的那些层以外的所有 layers 禁用它。

```
(defun dotspacemacs/layers ()
  ;; List of configuration layers to load.
  (setq-default dotspacemacs-configuration-layers
    '(java python c-c++
      (auto-completion :enabled-for java python))))
```

注意 :enabled-for 可能是一个空的列表。

```
(defun dotspacemacs/layers ()
  ;; List of configuration layers to load.
  (setq-default dotspacemacs-configuration-layers
    '(java python c-c++
      (auto-completion :enabled-for))))
```

如果两者都存在，:enabled-for 优先于 :disabled-for。

7.6.3 Selecting/Ignoring packages of a layer

默认情况下，声明的 layer 将安装/配置所有关联的包。你可能只想选择其中的一部分或忽略其中的一部分。这可以通过 :packages 关键字来实现。

例如忽略来自 spacemacs-ui-visual layer 的 neotree 和 fancy-battery 包：


```
(defun dotspacemacs/layers ()  
  ;; List of configuration layers to load.  
  (setq-default dotspacemacs-configuration-layers  
    '(auto-completion  
      (spacemacs-ui-visual :packages (not neotree fancy-battery))))
```

相反的是忽略除neotree和fancy-battery之外的所有软件包：

```
(defun dotspacemacs/layers ()  
  ;; List of configuration layers to load.  
  (setq-default dotspacemacs-configuration-layers  
    '(auto-completion  
      (spacemacs-ui-visual :packages neotree fancy-battery)))
```

注意：忽略 layer 中的包不同于排除包。排除的包将从您的配置中完全删除，而被忽略的包仅在给定 layers 中被忽略，但可以保留在您的系统上。如果给定的 layers 是包的所有者，那么忽略这个包与排除它是一样的（因为包成为孤立的，所以它被认为是 spacemacs 未使用的）。

7.6.4 Excluding packages

您可以通过变量 dotspacemacs-excluded-packages 排除您不想安装的软件包（有关软件包的更多信息，请参阅[配置 layers](#)）。

例如，禁用 rainbow-delimiters 包：

```
(setq-default dotspacemacs-excluded-packages '(rainbow-delimiters))
```

当您排除软件包时，下次启动 emacs 或下次点文件同步时，spacemacs 会自动删除它。所有的孤立依赖也被自动删除。不包括一个软件包有效地删除 spacemacs 的所有引用，而不会破坏其余的配置，这是一个强大的功能，它允许你快速删除 spacemacs 的任何功能。

注意：一些软件包对于 spacemacs 正确运行是必不可少的，这些软件包是受保护的，即使它们成为孤立的或被排除在外，也不能被排除或不被拆除。use-package 是一个不能从 spacemacs 中删除的受保护软件包的例子。

8 Concepts

8.1 Editing Styles

spacemacs 带有一种可动态切换的编辑风格，提供了一种更简单的方式来进行配对编程，例如 vim 用户和 emacs 用户之间。

有三种风格可供选择：

- Vim

- Emacs
- Hybrid(a mix between Vim and Emacs)

8.1.1 Vim

spacemacs 的行为就像在 vim 中使用 Evil mode 包模拟 vim 快捷键绑定。这是 spacemacs 的默认样式;可以通过在 dotfile 中将 dotspacemacs-editing-style 变量设置为 vim 来明确设置。

在 vim 编辑风格 (插入状态) 中绑定键值:

```
(define-key evil-insert-state-map (kbd "C-]") 'forward-char)
```

8.1.2 Emacs

spacemacs 的行为就像在原始 emacs 中使用 Holy mode 配置 Evil, 使 emacs 处于默认状态。在 dotfile 中将 dotspacemacs-editing-style 变量设置为 emacs。

在 emacs 风格 leader 是可用的 M-m。当关闭 vim 风格时, 可以使用 SPC t E e 和 M-m t E e 来打开和关闭。

在 emacs 编辑风格 (emacs state) 中绑定键值:

```
(define-key evil-emacs-state-map (kbd "C-]") 'forward-char)
```

8.1.3 Hybrid

Hybrid 编辑风格就像 vim 风格, 只不过插入状态被称为混合状态的新状态所取代。在混合状态下, 所有 emacs 键绑定都可用;这就像用 emacs 状态替换插入状态, 但提供了一个孤立的键映射 evil-hybrid-state-map。

以 Hybrid 编辑风格 (hybrid state) 绑定键:

```
(define-key evil-hybrid-state-map (kbd "C-]") 'forward-char)
```

这种风格可以调整为更像 Emacs 或更像 Vim 取决于用户的喜好。以下变量可用于更改样式配置:

- hybrid-mode-default-state 打开新缓冲区时的默认状态, 默认是 normal。将其设置为 emacs 以获得更多的流畅风格。
- hybrid-mode-enable-hjkl-bindings 如果不是空, 那么软件包将配置 h j k l 导航的键绑定。
- hybrid-mode-enable-evilified-state, 如果非 nil 缓冲区在被支持的情况是 evilified, 如果 nil 则在这些缓冲区中启用 emacs 状态。

默认配置是:

```
(setq-default dotspacemacs-editing-style '(hybrid :variables
                                              hybrid-mode-enable-evilified-
state t
                                              hybrid-mode-enable-hjkl-bindings
nil
                                              hybrid-mode-default-state
'normal))
```

使用 SPC t E h 和 M-m t E h 来切换混合风格。当关闭vim样式启用。

8.2 States

spacemacs有10种状态:

State	Default Color	Description
normal	orange	像 vim 的 normal mode，用来执行和组合命令
insert	green	像 vim 的 insert mode，用于实际插入文本
visual	gray	像 vim 的 visual mode，用来进行文本选择
motion	purple	Evil 专属，用于导航只读缓冲区
emacs	blue	Evil 专属，使用这种状态就像使用没有 vim 的普通 emacs 一样
replace	chocolate	Evil专属，覆盖点下的字符而不是插入新字符
hybrid	blue	Spacemacs 专属，这就像插入状态，除了所有的 emacs 键绑定是可用的
evilified	light brown	Spacemacs 专属，这是一个 emacs 状态修改，以带来 vim 导航，选择和搜索。
lisp	pink	Spacemacs 专属，用于导航 lisp 代码并修改它 (更多信息)
iedit	red	Spacemacs 专属，用于使用 iedit 在多个文本区域之间导航 (更多信息)
iedit-insert	red	Spacemacs 专属，用于使用 iedit 替换多个文本区域 (更多信息)

注意：从技术上讲还有 operator 的 evil 状态。

8.3 Evilified modes

一些缓冲区不用于编辑文本，并为某些操作提供自己的键绑定。这些经常与vim绑定冲突。为了使这样的缓冲区更像 vim 一致的方式，他们使用一个称为 evilified 状态的特殊状态。在 evilified 状态中，一些键工作在Evil中，有 /, :, h, j, k, l, n, N, v, V, gg, G, C-f, C-b, C-d, C-e, C-u, C-y 和 C-z，所有其他的按键按照底层模式的意图工作。

Shadowed keys 将按照以下模式移动：a → A → C-a → C-A。

例如，如果模式将函数绑定到 `n`，在 `evilified` 状态下在 `C-n` 下被发现，因为 `n` 和 `N` 都是保留的，但 `C-n` 不是。另一方面，任何最初绑定到 `k` 的东西都会在 `K` 上找到，因为 `k` 是保留的，但 `K` 不是。如果在 `K` 上有绑定，那将被移到 `C-k`。

除此之外，`C-g`（作为 `emacs` 中的一个重要的转义键）被跳过。所以任何绑定到 `g` 的东西都会在 `C-g` 上找到，因为 `g`，`G` 和 `C-g` 都是保留的。

8.4 Evil leader

`spacemacs` 使用 `leader` 键来绑定几乎所有的键绑定。

这个 `leader` 键 通常被 `vim` 用户设置在 `spacemacs` 中，`leader` 键 被设置在 `SPC`（空格键，因此命名为 `spacemacs`）上。这个键是键盘上最容易访问的键，用拇指按下这个键是降低 `rsi` 风险的好选择。它可以使用变量 `dotspacemacs-leader-key` 和 `dotspacemacs-emacs-leader-key` 来定制到任何其他键。

使用 `spacemacs` 不需要重新映射你的键盘修改器来试图减少 `RSI` 的风险，当你在正常模式下按下 `SPC leader` 键时，每个命令都可以很容易地执行，这里有几个例子：

- 保存一个buffer: `SPC f s`
- 保存所有打开的buffer: `SPC f S`
- 打开（切换）到一个缓冲区用 `helm`: `SPC b b`

8.5 Universal argument

通用参数 `C-u` 是 `emacs` 中的一个重要命令，但它也是一个非常方便的 `vim` 键绑定向上滚动。

`spacemacs` 绑定 `C-u` 以向上滚动并将通用参数绑定更改为 `spc u`。

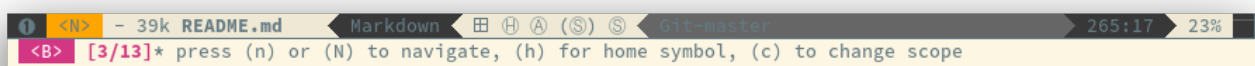
注意：`SPC u` 在 `helm-M-x`（`SPC SPC`）之前不工作。相反，首先调用 `helm-M-x`，选择要运行的命令，然后在按下 `return` 之前按 `C-u`。例如：`SPC SPC org-reload C-u RET`

8.6 Transient-states

`spacemacs` 定义了各种各样的 `transient states`（临时覆盖图）。这可以防止在 `SPC` 键上进行重复和繁琐的按压。

当 `transient state` 处于活动状态时，文档将显示在小型 `buffer` 中。附加信息也可以显示在小 `buffer` 中。

自动高亮符号 `transient state`：



文本缩放 `transient state`：



9 Differences between Vim, Evil and Spacemacs

. 键，“在 vim 中反向重复上一个 f, t, F 或 T 命令，但是在 spacemacs 中，默认情况下它是主要模式特定的 leader 键（可以在 dotfile 中的另一个键绑定上设置）

发送一个 PR 来添加您在本节中找到的差异。

9.1 The vim-surround case

有一个明显的可见区别。它不是在 Evil 和 vim 之间，而是在 spacemacs 和 vim-surround 之间：在 visual 模式下，surround 命令在 vim-surround 中，而在 spacemacs 中则在 S 上。

这是一些可以让一些 vim 用户感到惊讶的东西，所以这里有一些动机背后的变化：

- s 和 c 在 visual state 下做同样的事情，
- s 仅用于删除一个字符并添加一个非常狭窄的用例的多个字符
- c 可以移动，可以做一切 normal 状态下的事情（注意这对于 r 也是如此，但是 r 更有用，因为它保持 normal state）
- surround 命令只是一个比 s 更强大的命令。

如果你不相信，那么这里是恢复到默认的 vim + vim-surround 设置（将它添加到你的 dotspacemacs/user-config 函数或你的 ~/.spacemacs）的片段：

```
(evil-define-key 'visual evil-surround-mode-map "s" 'evil-substitute)
(evil-define-key 'visual evil-surround-mode-map "S" 'evil-surround-region)
```

10 Evil plugins

spacemacs 附带以下 evil 的插件：

Mode	Description
evil-args	motions 和 text 对象的参数
evil-exchange	port of vim-exchange
evil-indent-textobject	添加基于缩进的文本对象
evil-matchit	port of matchit.vim
evil-nerd-commenter	port of nerdcommenter
evil-numbers	like C-a and C-x in vim
evil-search-highlight-persist	emulation of hlsearch behavior
evil-surround	port of vim-surround
evil-visualstar	search for current selection with *
NeoTree	mimic NERD Tree

11 Binding keys

键序列绑定到各个键盘映射中的 emacs 中的命令。最基本的 map 是 global-map。使用 global-set-key 函数来实现 global-map 中的键绑定。示例将键绑定到命令 forward-char：

```
(global-set-key (kbd "C-]") 'forward-char)
```

kbd 宏接受描述键序列的字符串。global-map 经常被其他地图遮蔽。例如，evil 模式定义了目标状态的键盘映射（或 vim 术语中的模式）。这里是一个创建与上面相同的绑定的例子，但只有在插入状态下（define-key 是一个内置函数，evil-mode 有自己定义键的函数）。

```
(define-key evil-insert-state-map (kbd "C-]") 'forward-char)
```

也许对于 spacemacs 来说最重要的是使用 bind-map 包来绑定 leader 键后的键。这是大多数 spacemacs 绑定的地方。绑定在 leader 键后的键是通过函数 spacemacs/set-leader-keys 和 spacemacs/set-leader-keys-for-major-mode 来实现的，例如：

```
(spacemacs/set-leader-keys "C-]" 'forward-char)
(spacemacs/set-leader-keys-for-major-mode 'emacs-lisp-mode "C-]" 'forward-char)
```

最后，应该知道前缀键。实质上，所有键映射都可以嵌套。嵌套的键映射广泛用于 spacemacs，在 vanilla emacs 的这个问题。例如，SPC 指向“applications”的键绑定，例如用于 calc-dispatch 的 SPC a c。嵌套绑定很容易。

```
(spacemacs/declare-prefix "]" "bracket-prefix")
(spacemacs/set-leader-keys "]]" 'double-bracket-command)
```

第一行声明 SPC] 为前缀，第二行将键序列 SPC]] 绑定到相应的命令。第一行实际上并不需要创建前缀，但它会为您的新前缀提供 key-discovery 工具可以使用的名称 (e.g., which-key)。

有很多关于绑定键的说法，但这些都是基础知识。键可以绑定在你的 ~/.spacemacs 文件或者单独的 layers 中。

12 GUI Elements

spacemacs 有一个简约和分布的免费的图形用户界面：

- 自定义 [powerline](#) mode-line 与 [颜色反馈](#) 根据当前 [Flycheck](#) 状态
- Unicode 符号为了 minor mode 不兼容 出现在 mode-line
- 自定义 [fringe bitmaps](#) 和 Flycheck 的错误反馈

12.1 Color themes

官方的 spacemacs 主题是 spacemacs-dark，它是您首次启动 spacemacs 时安装的默认主题。有两个主题的变体，一个是黑色(dark)的，一个是轻(light)的。这些主题的一些方面可以在 `~/spacemacs` 的 `dotspacemacs/user-init` 函数中定制：

- 带有布尔值 `spacemacs-theme-comment-bg` 的注释背景
- `org` 部分标题的高度与 `spacemacs-theme-org-height`

可以使用变量 `dotspacemacs-themes` 在 `~/spacemacs` 中定义默认主题。例如指定 `spacemacs-light`，`leuven` 和 `zenburn`：

```
(setq-default dotspacemacs-themes '(spacemacs-light leuven zenburn))
```

Key Binding	Description
SPC T n	切换到在 <code>dotspacemacs-themes</code> 中列出的下一个主题。
SPC T s	从 <code>helm buffer</code> 选择一个主题。

您可以从 [Rob Merrell](#) 的[主题库](#)中看到所有包含主题的样本。

注意：

- 您不需要在 `layers` 中明确列出您在 `dotspacemacs-themes` 中定义的主题包，但是 spacemacs 非常聪明，可以从孤立列表中删除这些包。
- 由于 emacs 中主题的内部工作，在同一 session 中切换主题可能会有一些奇怪的副作用。虽然这些副作用应该是非常罕见的。
- 在 emacs 的终端版本中，颜色主题将无法正确呈现，因为终端呈现的是颜色，而不是由 emacs 呈现。您可能不得不改变你的终端调色板。更多的解释可以在 [emacs-color-theme-solarized](#) 网页上找到。

提示：如果你是一名 Org 用户，[leuven-theme](#) 是你很好的选择。

12.2 Font

spacemacs 使用的默认字体是 [Source Code Pro](#)。如果你想使用它，建议将它安装在你的系统上。

更改默认字体在您的 `.spacemacs` 文件中设置变量 `dotspacemacs-default-font`。默认情况下它的值是：

```
(setq-default dotspacemacs-default-font '("Source Code Pro"
                                           :size 13
                                           :weight normal
                                           :width normal
                                           :powerline-scale 1.1))
```

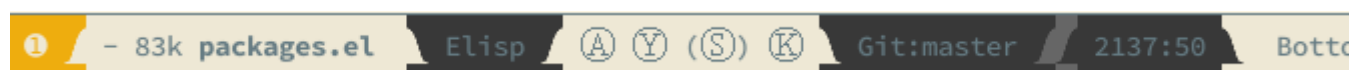
如果没有找到指定的字体，将使用回退之一（取决于您的系统）。还要注意，如果您在终端中运行 emacs，更改此值不起作用。

属性应该非常简单，可以设置 [font-spec](#) 的任何有效属性：

- `:family` 字体族或 `fontset`（一个字符串）。

- `:width` 相对字符宽度。这应该是其中的一个符号：
 - `ultra-condensed`
 - `extra-condensed`
 - `condensed`
 - `semi-condensed`
 - `normal`
 - `semi-expanded`
 - `expanded`
 - `extra-expanded`
 - `ultra-expanded`
- `:height` 字体的高度。在最简单的情况下，这是以1/10点为单位的整数。
- `:weight font weight` - 符号之一（从最密集到最薄弱）：
 - `ultra-bold`
 - `extra-bold`
 - `bold`
 - `semi-bold`
 - `normal`
 - `semi-light`
 - `extra-light`
 - `ultra-light`
- `:slant` 字体倾斜 - 符号之一：
 - `italic`
 - `oblique`
 - `normal`
 - `reverse-italic`
 - `reverse-oblique`
- `:size` 字体大小 - 指定像素大小的非负整数或指定点大小的浮点数。
- `:adstyle` 字体的其他印刷风格信息，如“sans”。该值应该是一个字符串或一个符号。
- `:registry charset` 注册表和字体的编码，如'iso8859-1'。该值应该是一个字符串或一个符号。
- `:script` 字体必须支持的脚本（一个符号）。

特殊属性：`powerline-scale` 是 spacemacs 特有的，用于快速调整模式行高度，以避免像下面截图（默认值为 1.1）那样对分隔符进行糟糕的渲染。



Ugly separators

12.3 GUI Toggles

一些图形用户界面指标可以打开和关闭（切换从 `t` 和 `T` 开始）：

Key Binding	Description
SPC t 8	高亮显示第80列以后的任何字符
SPC t f	显示填充列（默认填充列设置为80）
SPC t h h	切换当前行的高亮显示

Key Binding	Description
SPC t h i	切换突出显示缩进级别
SPC t h c	切换高亮显示缩进当前列
SPC t i	在点处切换缩进指南
SPC t l	切换截断线
SPC t L	切换视线
SPC t n	切换行号
SPC t v	切换平滑滚动
Key Binding	Description
SPC T ~	在空行的边缘显示 ~
SPC T F	全屏切换帧
SPC T f	切换显示边缘
SPC T m	切换菜单栏
SPC T M	切换帧最大化
SPC T t	切换工具栏
SPC T T	切换帧透明度并进入透明度瞬态状态

注意：这些切换都可以通过 `helm-spacemacs-help` 界面获得（按 `SPC h SPC` 显示 `helm-spacemacs-help` 缓冲区）。

12.3.1 Global line numbers

可以通过在 `~/.spacemacs` 中将 `dotspacemacs-line-numbers` 变量设置为与 `nil` 不同的值来在所有编程模式和文本模式缓冲区中切换行号。

```
(setq-default dotspacemacs-line-numbers t)
```

如果它被设置为 `relative`，行号以相对的方式显示：

```
(setq-default dotspacemacs-line-numbers 'relative)
```

12.4 Mode-line

mode line 是一个高度定制的 [powerline](#)，具有以下功能：

- 显示窗口号码
- 当前状态的颜色代码

- 通过 anzu 显示搜索次数
- 切换 flycheck 信息
- 切换电池信息
- 切换 minor mode lighters

提醒各状态的颜色代码：

Evil State	Color
Normal	Orange
Insert	Green
Visual	Grey
Emacs	Blue
Motion	Purple
Replace	Chocolate
Lisp	Pink
ledit/ledit-Insert	Red

一些元素可以动态切换：

Key Binding	Description
SPC t m b	切换电池状态
SPC t m c	切换 org 任务时钟（在 org layer 中可用）
SPC t m m	切换 minor mode lighters
SPC t m M	切换 major mode
SPC t m n	切换 cat！（如果在 dotfile 中声明了 colors layer）
SPC t m p	切换点字符位置
SPC t m t	切换 mode line 本身
SPC t m v	切换版本控制信息
SPC t m V	切换新版本更轻

12.4.1 Powerline font installation for terminal-mode users

在终端模式下运行 emacs 的用户可能需要安装 [powerline 补丁字体](#) 并配置其终端客户端使用它们使 powerline 分隔符正确呈现。

12.4.2 Flycheck integration

当启用 Flycheck 次要模式时，会显示一个新元素，显示错误，警告和信息的数量。



Flycheck integration in mode-line

12.4.3 Anzu integration

Anzu 显示执行搜索时发生的次数。当 n 或 N 被按下时，spacemacs 通过暂时显示 Anzu 状态来很好地集成 Anzu 状态。请参阅以下屏幕截图中的 5/6 细分。



Anzu integration in mode-line

12.4.4 Battery status integration

fancy-battery 显示电池总电量的百分比，以及电池完全充电或放电的剩余时间。

颜色码用于电池状态：

Battery State	Color
Charging	Green
Discharging	Orange
Critical	Red

请注意，这些颜色可能因您的主题而异。

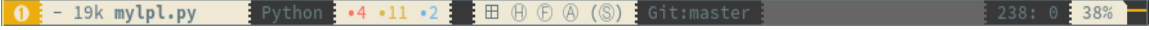
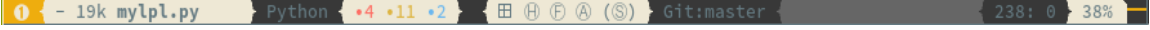
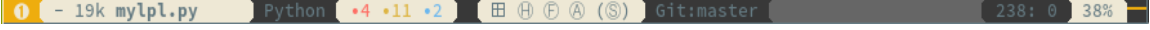
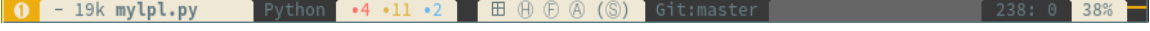
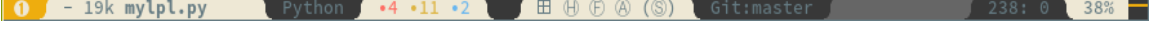
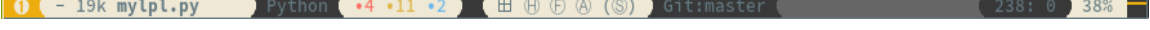
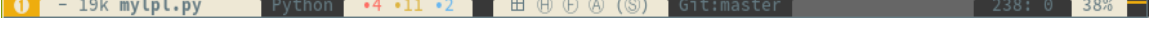
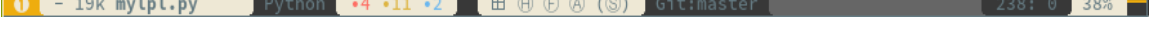

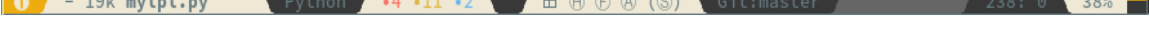
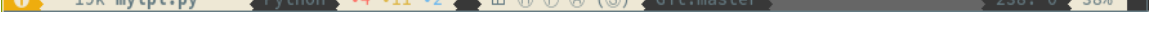
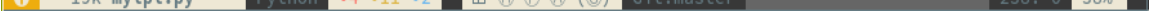
12.4.5 Powerline separators

可以通过在 ~/spacemacs 中设置 powerline-default-separator 变量，然后重新编译模式行来轻松定制 powerline 分隔符。例如，如果您想将分隔符设置回已知的箭头分隔符，请将以下代码段添加到您的配置文件中：

```
(defun dotspacemacs/user-config ()
  "This is where you can ultimately override default Spacemacs
  configuration.
  This function is called at the very end of Spacemacs initialization."
  (setq powerline-default-separator 'arrow))
```

为了节省时间来尝试所有可能的 powerline，下面是一组详尽的截图：

Separator	Screenshot
alternate	
arrow	
arrow-fade	
bar	

Separator	Screenshot
box	
brace	
butt	
hamfer	
contour	
curve	
rounded	
roundstub	
slant	
wave	
zigzag	
nil	

12.4.6 Minor Modes

spacemacs 使用 [diminish mode](#) 来减小 minor mode 指示器的大小：

minor mode 区域可以用 `spc t m m` 来打开和关闭

unicode 符号默认显示。在 `~/spacemacs` 中将变量 `dotspacemacs-mode-line-unicode-symbols` 设置为 `nil` 将会显示 `ascii` 字符（如果你不能设置合适的字体，可能会在终端中使用）。

显示在 `mode-line` 中的字母与用于切换它们的键绑定相对应。

一些切换有两种风格：`local` 和 `global`。可以使用 `control` 键来达到切换的全局版本。

Key Binding	Unicode	ASCII	Mode
SPC t -	⌘	-	centered-cursor mode
SPC t 8	⑧	8	toggle highlight of characters for long lines
SPC t C-8	⑧	8	global toggle highlight of characters for long lines
SPC t C--	⌘	-	global centered cursor
SPC t a	Ⓐ	a	auto-completion
SPC t c	Ⓒ	c	camel case motion with subword mode
none	Ⓔ	e	evil-org mode
SPC t E e	Ⓔe	Ee	emacs editing style (holy mode)
SPC t E h	Ⓔh	Eh	hybrid editing style (hybrid mode)

Key Binding	Unicode	ASCII	Mode
SPC t f	ⓕ	f	fill-column-indicator mode
SPC t F	ⓕ	F a	uto-fill mode
SPC t g	ⓖ	g	golden-ratio mode
SPC t h i	ⓗi	hi	toggle highlight indentation levels
SPC t h c	ⓗc	hc	toggle highlight indentation current column
SPC t i	ⓔ	i	indentation guide
SPC t C-i	ⓔ	i	global indentation guide
SPC t l	ⓔ	l	aggressive indent mode
SPC t K	ⓔ	K	which-key mode
SPC t p	ⓔ	p	smartparens mode
SPC t C-p	ⓔ	p	global smartparens
SPC t s	ⓔ	s	syntax checking (flycheck)
SPC t S	ⓔ	S	enabled in spell checking layer (flyspell)
SPC t C-W	ⓔ	W	automatic whitespace cleanup globally
SPC t y	ⓔ	y	yasnippet mode

12.4.7 Customizing the mode-line

spacemacs 使用 [Spaceline](#) 来提供其 mode-line。它由左右两侧的若干段组成。这些在变量 spaceline-left 和 spaceline-right 中定义。可以使用 spaceline-define-segment 定义段，并将其添加到左侧或右侧变量的相应位置。

请参阅 Spaceline 文件以获取更多信息。

布局是具有缓冲区隔离的窗口配置，每个布局可以定义几个工作空间（将它们视为子布局）共享与其父布局相同的缓冲区列表。

13 Layouts

布局是与缓冲区列表关联的窗口配置。缓冲区列表可以是任意选择的一组缓冲区。spacemacs 提供了一些设施来创建有意义的缓冲区集，例如与一个抛射项目相关的缓冲区。

当前布局的名称出现在最左边的模式行中（模式行的第一个元素）。

要创建一个新的布局键入一个尚不存在的布局编号。例如，如果当前有两个布局，则键入 SPC l 3 以创建第三个布局。

13.1 The default layout

默认布局（在 emacs 启动时创建的布局）不会显示在 mode-line 中，但可以通过将变量 `dotspacemacs-display-default-layout` 设置为 `t` 来显示它。

它的名字默认是“default”，但是可以通过设置变量 `dotspacemacs-default-layout-name` 来改变。

默认布局是特殊的，因为它具有全局范围，意味着所有打开的缓冲区都属于它。所以只使用默认布局感觉就像根本不使用布局。

13.2 Project layouts

一个项目布局是绑定到一个抛射项目。使用 `SPC p l` 创建项目布局。

布局的名称是项目根目录的名称。

13.3 Custom Layouts

可以使用宏 `spacemacs | define-custom-layout` 定义自定义布局，可以通过 `SPC l o` 访问它们。

按照惯例，自定义布局的名称应该以 `@` 开头。

ERC 缓冲区的自定义布局定义示例：

```
(spacemacs|define-custom-layout "@ERC"
  :binding "E"
  :body
  (progn
    ;; hook to add all ERC buffers to the layout
    (defun spacemacs-layouts/add-erc-buffer-to-persp ()
      (persp-add-buffer (current-buffer)
        (persp-get-by-name
          erc-spacemacs-layout-name)))
    (add-hook 'erc-mode-hook #'spacemacs-layouts/add-erc-buffer-to-persp)
    ;; Start ERC
    (call-interactively 'erc)))
```

然后用 `SPC l o E` 来启动它自己的布局中的 ERC。任何新的 ERC 缓冲区将成为自定义布局的一部分。

一些与 spacemacs 一起提供的自定义布局：

Name	Key Binding	Description
@Spacemacs	e	Custom perspective containing all buffers of ~/.emacs.d
@ERC	E	Custom perspective containing all ERC buffers (needs the erc layer enabled)
@RCIRC	i	Custom perspective containing all RCIRC buffers (needs the rcirc layer enabled)
@Org	o	Custom perspective containing all the org-agenda buffers

13.4 Save/Load layouts into a file

使用 `SPC l s` 和 `SPC l L` 可以保存并从一个文件加载布局。

注意：默认情况下，spacemacs 将自动保存名称为 `persp-auto-save` 的布局

将变量 `dotspacemacs-auto-resume-layouts` 设置为 `t` 将自动恢复上次保存的布局。

13.5 Layout key bindings

快捷键绑定被注册为临时态。临时状态的文档字符串显示现有的布局，当前活动的布局有方括号。按布局编号将激活它（或创建一个新的）并退出瞬态。可以用 `Ctrl-` 预览一个布局。按下 `TAB` 将激活之前选择的布局。

按 `?` 切换完整的帮助。

Key Binding	Description
<code>SPC l</code>	activate the transient- state
<code>?</code>	toggle the documentation
<code>[1..9, 0]</code>	switch to nth layout
<code>[C-1..C-9, C-0]</code>	switch to nth layout and keep the transient state active
	switch to the latest layout
<code>a</code>	add a buffer to the current layout
<code>A</code>	add all the buffers from another layout in the current one
<code>b</code>	select a buffer in the current layout
<code>d</code>	delete the current layout and keep its buffers
<code>D</code>	delete the other layouts and keep their buffers
<code>h</code>	go to default layout
<code>C-h</code>	previous layout in list
<code>l</code>	select/create a layout with helm
<code>L</code>	load layouts from file
<code>C-l</code>	next layout in list
<code>n</code>	next layout in list
<code>N</code>	previous layout in list
<code>o</code>	open a custom layout
<code>p</code>	previous layout in list
<code>r</code>	remove current buffer from layout
<code>R</code>	rename current layout

Key Binding	Description
s	save layouts
t	display a buffer without adding it to the current layout
w	workspaces transient state (needs eyebrowse layer enabled)
x	kill current layout with its buffers
X	kill other layouts with their buffers

13.6 Workspaces

工作空间是子布局，他们允许定义多个布局到一个给定的布局，这些布局共享与父布局相同的缓冲区。

在窗口编号之前显示当前激活的工作空间编号，例如“**1**|4”或“1 | 4”表示第一个工作空间的第四个窗口。

任何新的布局都带有一个默认的工作区，它是工作区1。

切换到当前布局中不存在的工作空间将创建一个新的工作空间。例如在启动时，您可以按 SPC l w 2 以默认布局创建工作区2。

当创建一个工作空间是匿名的，你可以给他们一个名字与 SPC l w R.

13.7 Workspace key bindings

快捷键绑定被注册为临时态。临时状态的文档字符串显示现有的工作区，当前活动的工作区有方括号。按下工作区号码将激活它（或创建一个新的）并退出瞬态状态。可以用 Ctrl- 预览一个工作区。按 TAB 将激活之前选择的工作区。

按 ? 切换完整的帮助。

Key Binding	Description
SPC l w	activate the transient state
?	toggle the documentation
[1..9, 0]	switch to nth workspace
[C-1..C-9, C-0]	switch to nth workspace and keep the transient state active
TAB	switch to last active workspace
d	close current workspace
n or l	switch to next workspace
N or p or h	switch to previous workspace
R	set a tag to the current workspace
w	switched to tagged workspace

还有一些方便的全局可用的与工作区相关的键绑定：

Key Binding	Description
gt	go to next workspace
gT	got to previous workspace
SPC b W	go to workspace and window by buffer

14 Commands

14.1 Vim key bindings

spacemacs 是基于 vim 模式的用户界面来导航和编辑文本。如果你不熟悉 vim 编辑文本的方式，你可以随时按下 SPC h T 来试试这个 [evil-tutor](#) 教程。

14.1.1 Escaping

spacemacs 通过快速按下 fd 键，使用 evil-escape 来轻松切换插入状态和正常状态。

fd 的选择使得能够使用相同的序列从 emacs 中的 “everything”:

- 从一切 evil 状态摆脱到正常的状态
- 从 evil-lisp-state 到正常状态
- 从 evil-iedit-state 到正常状态
- 放弃 evil 的命令
- 退出 minibuffer
- 放弃 isearch
- 退出 magit buffers
- 退出 help buffers
- 退出 apropos buffers
- 退出 ert buffers
- 退出 undo-tree buffer
- 退出 paradox
- 退出 gist-list menu
- 退出 helm-ag-edit
- 隐藏 neotree buffer

如果您发现自己处于 Spacemacs (SPC) 或 vim 键 绑定不起作用的缓冲区中，则可以使用它恢复到正常状态（例如，在 SPC SPC customize 按 fd 以使 SPC b b 再次运行）。

这个序列可以在你的 ~/.spacemacs 中定制。例如将其设置为 jj:

```
(defun dotspacemacs/user-config ()
  (setq-default evil-escape-key-sequence "jj"))
```

注意：尽管 jj 或 jk 是 vim 用户的流行选择，但这些关键序列对于 spacemacs 并不是最优的。事实上，在 visual 状态下很快就会很快地按下，不经意地逃到 normal 状态。

14.1.2 Executing Vim and Emacs ex/M-x commands

Command	Key Binding
Vim (ex-command)	:
Emacs (M-x)	SPC SPC

可以使用 `~/./spacemacs` 的变量 `dotspacemacs-emacs-command-key` 来更改 `emacs` 命令键 `SPC` (在 `leader key` 后面执行) 。

14.1.3 Leader key

在 `vim` 模式之上 (模式被称为 `spacemacs` 中的状态) ，有一个特殊的键被称为 `leader` 键，它曾经被按下给出一个全新的键盘层。`leader` 键是默认的 `SPC` (空格) 。可以用变量 `dotspacemacs-leader-key` 来改变这个键。

14.1.4 Additional text objects

在`spacemacs`中定义了额外的文本对象：

Object	Description
a	an argument
g	the entire buffer
\$	text between \$
*	text between *
8	text between /* and */
%	text between %
\vert	text between \vert

14.2为用户保留前缀命令

`SPC o` 和 `SPC m o` 是为用户保留的。在这些背后设置键绑定保证不会与 `spacemacs` 默认键绑定冲突。

例如：在你的 `~/./spacemacs` 文件里放置 (`spacesmacs/ set-leader-keys"oc"org-capture`) 到 `dotspacemacs/user-config` 中，以便能够使用 `SPC o c` 来运行组织模式捕获。

14.3 Completion

`spacemacs` 由两个增量完成和选择缩小框架之一驱动：`helm` (默认) 或 `ivy`。要使用 `ivy`，请将 `ivy layer` 添加到启用的 `layer` 列表中。如果 `ivy layer` 没有启用，`helm` 将自动启用。（ 请注意，由于 `Helm` 是两者中较为成熟的，所以如果选择 `ivy`，某些功能可能无法使用。 ）

这些完成系统是 `spacemacs` 的中央控制塔，他们被用来管理缓冲区(buffers)，项目(projects)，搜索结果(search results)，配置层(configuration)，切换(toggles)和更多...

掌握你完成系统的选择将会使你成为一名超级 `Spacemacs` 用户。

14.3.1 Helm

不要犹豫来阅读[Helm wiki 文档](#)。

14.3.1.1 C-z and Tab switch

绑定到 C-z 的命令比绑定到 Tab 的命令更有用，所以交换它们是有意义的。[这里](#)也推荐。

14.3.1.2 Helm focus

如果您发现自己无法将焦点返回到 Helm（例如粗心的点击鼠标之后），请使用 SPC w b 将焦点返回到 minibuffer。

14.3.1.3 Helm transient state

Spacemacs 为 Helm 定义了一个[临时状态](#)，使其像 [Vim's Unite](#) 插件一样工作。

在 Helm 缓冲区中用 M-SPC 或 s-M-SPC 启动临时状态。

Key Binding	Description
M-SPC or s-M-SPC	initiate the transient state
q	quit transient state
TAB	switch to actions page and leave the transient state
1	execute action 0
2	execute action 1
3	execute action 2
4	execute action 3
5	execute action 4
6	execute action 5
7	execute action 6
8	execute action 7
9	execute action 8
0	execute action 9
a	switch to actions page
g	go to first candidate
G	go to last candidate
h	go to previous source
j	select next candidate
k	select previous candidate

Key Binding	Description
<code>l</code>	go to next source
<code>t</code>	mark current candidate
<code>T</code>	mark all candidates
<code>v</code>	execute persistent action

14.4 Discovering

14.4.1 Key bindings

14.4.1.1 Which-key

每次在正常模式下按下 `SPC` 键时都会显示帮助缓冲区。它列出了可用的键绑定及其相关的命令。

默认情况下，在键被按下后，which-key 缓冲器将被快速显示。你可以通过设置你喜欢的变量 `dotspacemacs-which-key-delay` 来改变延迟（值在秒）。

14.4.1.2 Helm describe key bindings

可以通过按 `SPC ?` 搜索特定的键绑定。

通过使用 `leader` 键类型来将列表缩小到某些键绑定，例如这个正则表达式：`SPC\b`，它将列出所有与缓冲区有关的绑定。

14.4.2 Getting help

`Describe functions` 是强大的 `emacs` introspection 命令来获取有关函数(functions)，变量(variables)，模式(modes)等信息，因此这些命令是绑定的：

Key Binding	Description
<code>SPC h d b</code>	describe bindings in a helm buffer
<code>SPC h d c</code>	describe current character under point
<code>SPC h d d</code>	describe current expression under point
<code>SPC h d f</code>	describe a function
<code>SPC h d F</code>	describe a face
<code>SPC h d k</code>	describe a key
<code>SPC h d K</code>	describe a keymap
<code>SPC h d l</code>	copy last pressed keys that you can paste in gitter chat
<code>SPC h d m</code>	describe current modes
<code>SPC h d p</code>	describe a package (Emacs built-in function)

Key Binding	Description
SPC h d P	describe a package (Spacemacs layer information)
SPC h d s	copy system information that you can paste in gitter chat
SPC h d t	describe a theme
SPC h d v	describe a variable

其他 help 键绑定:

Key Binding	Description
SPC h SPC	discover Spacemacs documentation, layers and packages using helm
SPC h i	search in info pages with the symbol at point
SPC h k	show top-level bindings with which-key
SPC h m	search available man pages
SPC h n	browse emacs news

在 help-mode 下导航键绑定:

Key Binding	Description
g b	or [go back (same as clicking on [back] button)
g f	or] go forward (same as clicking on [forward] button)
g h	go to help for symbol under point

报告问题:

Key Binding	Description
SPC h I	Open Spacemacs GitHub issue page with pre-filled information
SPC u SPC h I	Open Spacemacs GitHub issue page with pre-filled information - include last pressed keys

注意: 如果这两个绑定与 *Backtrace* 缓冲区打开使用, 则自动包括 backtrace

14.4.3 Available layers

所有的层可以很容易地发现通过 helm-spacemacs-help 可访问的 SPC h SPC。。

下面的 helm actions 是可用的:

- 默认: 打开图层 readme.org
- 第二: 打开图层 packages.el

14.4.3.1 Available packages in Spacemacs

helm-spacemacs-help 还列出了 Spacemacs 中可用的所有软件包。入口格式是 (layer) 包。如果你输入 flycheck, 你将能够看到所有使用 flycheck 的 layers。

软件包中有以下 helm 操作:

- 默认: 去包初始化函数

14.4.3.2 New packages from ELPA repositories

package-list-packages 是您可以在不同的 Elpa 软件仓库中浏览所有可用软件包的地方。可以从那里升级软件包, 但不建议使用 spacemacs 启动页面上的 [Update Packages] 链接。

Spacemacs 使用 [Paradox](#) 而不是 package-list-packages 来列出可用的 ELPA 包。Paradox 增强了更好的反馈, 新的过滤器和 github 信息, 如星星数量的软件包列表缓冲区。也可以选择直接在缓冲区中打包。

重要的注意事项1: 从 **Paradox** 安装一个新的包不会使其持久。要持久地安装软件包, 必须将其明确添加到配置层。

重要说明2: 不要从 **Paradox** 或 **package-list-packages** 更新软件包, 因为它们不支持 Spacemacs 的回滚功能。

Key Binding	Description
SPC a k	launch paradox
/	evil-search
f k	filter by keywords
f r	filter by regexp
f u	display only installed package with updates available
h	go left
H	show help (not accurate)
j	go down
k	go up
l	go right
L	show last commits
n	next search occurrence
N	previous search occurrence
o	open package homepage
r	refresh
S P	sort by package name
S S	sort by status (installed, available, etc...)

Key Binding	Description
S *	sort by Github stars
v	visual state
V	visual-line state
x	execute (action flags)

14.4.4 Toggles

helm-spacemacs-help 也是发现可用切换的中心位置。只显示切换源按下 C-l (或在 [helm 临时状态](#) 下按 l)。

软件包中有以下 helm 操作：

- 默认值：打开/关闭

提示使用 SPC h l 来恢复最后的 helm 会话。快速打开和关闭切换是很方便的。

14.5 Navigating

14.5.1 Point/Cursor

导航是使用 Vi 键绑定 hjkl 来执行的。

Key Binding	Description
h	move cursor left
j	move cursor down
k	move cursor up
l	move cursor right
H	move cursor to the top of the screen
L	move cursor to the bottom of the screen
SPC j 0	go to the beginning of line (and set a mark at the previous location in the line)
SPC j \$	go to the end of line (and set a mark at the previous location in the line)
SPC t -	lock the cursor at the center of the screen

14.5.1.1 Smooth scrolling

[smooth-scrolling](#) 防止点到达屏幕的顶部或底部时跳转。它是默认启用的。

在 Windows 上，你可能想禁用它。禁用平滑滚动，将 `~/spacemacs` 中的 `dotspacemacs-smooth-scrolling` 变量设置为 `nil`：

```
(setq-default dotspacemacs-smooth-scrolling nil)
```

您也可以使用 `SPC t v` 切换平滑滚动。

14.5.2 Vim motions with avy

Spacemacs 使用 `avy` 的 `evil` 的整合，使运行期间 `avy` 的调用。

例如，从当前行删除一组视觉线是有用的。在包含一些文本的缓冲区中尝试以下顺序：`d SPC j l`，然后选择一个 `avy` 候选项。

Key Binding	Description
<code>SPC j b</code>	go back to the previous location (before the jump)
<code>SPC j j</code>	initiate avy jump char
<code>SPC j w</code>	initiate avy jump word
<code>SPC j l</code>	initiate avy jump line

14.5.2.1 ace-link mode

类似于 `avy`，`ace-link` 允许用户通过两个按键跳转到 `help-mode` 和 `info-mode` 下的任何链接。

Key Binding	Description
<code>o</code>	在 <code>help-mode</code> 和 <code>info-mode</code> 下启动 <code>ace-link</code>

14.5.3 Unimpaired bindings

Spacemacs 带有一个内置的 [tpepe's vim-unimpaired](#) 端口。

这个插件提供了几对括号映射，使用 `[` 表示前一个，`]` 表示下一个。

Key Bindings	Description
<code>[SPC</code>	Insert space above
<code>] SPC</code>	Insert space below
<code>[b</code>	Go to previous buffer
<code>] b</code>	Go to next buffer
<code>[f</code>	Go to previous file in directory
<code>] f</code>	Go to next file in directory
<code>[l</code>	Go to the previous error
<code>] l</code>	Go to the next error
<code>[h</code>	Go to the previous vcs hunk
<code>] h</code>	Go to the next vcs hunk

Key Bindings	Description
[q	Go to the previous error
] q	Go to the next error
[t	Go to the previous frame
] t	Go to the next frame
[w	Go to the previous window
] w	Go to the next window
[e	Move line up
] e	Move line down
[p	Paste above current line
] p	Paste below current line
g p	Select pasted text

14.5.4 Jumping, Joining and Splitting

SPC j 前缀用于跳转(jumping), 连接(joining)和分割(splitting)。

14.5.4.1 Jumping

Key Binding	Description
SPC j 0	go to the beginning of line (and set a mark at the previous location in the line)
SPC j \$	go to the end of line (and set a mark at the previous location in the line)
SPC j b	undo a jump (go back to previous location)
SPC j d	jump to a listing of the current directory
SPC j D	jump to a listing of the current directory (other window)
SPC j f	jump to the definition of an Emacs Lisp function
SPC j i	jump to a definition in buffer (imenu)
SPC j l	jump to a definition in any buffer (imenu)
SPC j j	jump to a character in the buffer (works as an evil motion)
SPC j J	jump to a suite of two characters in the buffer (works as an evil motion)
SPC j k	jump to next line and indent it using auto-indent rules
SPC j l	jump to a line with avy (works as an evil motion)
SPC j q	show the dumb-jump quick look tooltip

Key Binding	Description
SPC j u	jump to a URL in the current buffer
SPC j v	jump to the definition/declaration of an Emacs Lisp variable
SPC j w	jump to a word in the current buffer (works as an evil motion)

14.5.4.2 Joining and splitting

Key Binding	Description
J	join the current line with the next line
SPC j k	go to next line and indent it using auto-indent rules
SPC j n	split the current line at point, insert a new line and auto-indent
SPC j s	split a quoted string or s-expression in place
SPC j S	split a quoted string or s-expression, insert a new line and auto-indent

14.5.5 Window manipulation

14.5.5.1 Window manipulation key bindings

每个窗口都有一个 在mode-line 开始处显示的数字，可以使用 SPC number 快速访问。

Key Binding	Description
SPC 1	go to window number 1
SPC 2	go to window number 2
SPC 3	go to window number 3
SPC 4	go to window number 4
SPC 5	go to window number 5
SPC 6	go to window number 6
SPC 7	go to window number 7
SPC 8	go to window number 8
SPC 9	go to window number 9
SPC 0	go to window number 0

Windows操作命令（以 w 开头）：

Key Binding	Description
SPC w =	balance split windows

Key Binding	Description
SPC w b	force the focus back to the minibuffer (usefull with helm popups)
SPC w c	maximize/minimize a window and center it
SPC w C	maximize/minimize a window and center it using ace-window
SPC w d	delete a window
SPC u SPC w d	delete a window and its current buffer (does not delete the file)
SPC w D	delete another window using ace-window
SPC u SPC w D	delete another window and its current buffer using ace-window
SPC w t	toggle window dedication (dedicated window cannot be reused by a mode)
SPC w f	toggle follow mode
SPC w F	create new frame
SPC w h	move to window on the left
SPC w H	move window to the left
SPC w j	move to window below
SPC w J	move window to the bottom
SPC w k	move to window above
SPC w K	move window to the top
SPC w l	move to window on the right
SPC w L	move window to the right
SPC w m	maximize/minimize a window (maximize is equivalent to delete other windows)
SPC w M	swap windows using ace-window
SPC w o	cycle and focus between frames
SPC w p m	open messages buffer in a popup window
SPC w p p	close the current sticky popup window
SPC w r	rotate windows forward
SPC w R	rotate windows backward
SPC w s or SPC w -	horizontal split
SPC w S	horizontal split and focus new window
SPC w u	undo window layout (used to effectively undo a closed window)
SPC w U	redo window layout
SPC w v or SPC w /	vertical split

Key Binding	Description
SPC w V	vertical split and focus new window
SPC w w	cycle and focus between windows
SPC w W	select window using ace-window

14.5.5.2 Window manipulation transient state

一个方便的窗口操作临时状态允许执行上面列出的大部分操作临时状态允许额外的行为，如窗口大小调整。

Key Binding	Description
SPC w .	initiate transient state
?	display the full documentation in minibuffer
0	go to window number 0
1	go to window number 1
2	go to window number 2
3	go to window number 3
4	go to window number 4
5	go to window number 5
6	go to window number 6
7	go to window number 7
8	go to window number 8
9	go to window number 9
/	vertical split
-	horizontal split
[shrink window horizontally
]	enlarge window horizontally
{	shrink window vertically
}	enlarge window vertically
d	delete window
D	delete other windows
g	toggle golden-ratio on and off
h	go to window on the left
j	go to window below

Key Binding	Description
k	go to window above
l	go to window on the right
H	move window to the left
J	move window to the bottom
K	move bottom to the top
L	move window to the right
o	focus other frame
r	rotate windows forward
R	rotate windows backward
s	horizontal split
S	horizontal split and focus new window
u	undo window layout (used to effectively undo a closed window)
U	redo window layout
v	vertical split
V	horizontal split and focus new window
w	focus other window
Any other key	leave the transient state

14.5.5.3 Golden ratio

如果你像疯了一样调整窗口大小，你可能想尝试一下[黄金比例](#)。

黄金比例动态调整窗口的大小，取决于它们是否被选中。默认情况下，黄金比例关闭。

可以使用 `SPC t g` 来打开和关闭模式。

14.5.6 Buffers and Files

默认情况下，Spacemacs 使用 `helm` 来打开文件。

14.5.6.1 Buffers manipulation key bindings

缓冲区操作命令（以 `b` 开头）：

Key Binding	Description
SPC TAB	switch to alternate buffer in the current window (switch back and forth)
SPC b b	switch to a buffer using <code>helm</code>

Key Binding	Description
SPC b d	kill the current buffer (does not delete the visited file)
SPC u SPC b d	kill the current buffer and window (does not delete the visited file)
SPC b D	kill a visible buffer using ace-window
SPC u SPC b D	kill a visible buffer and its window using ace-window
SPC b C-d	kill buffers using a regular expression
SPC b e	erase the content of the buffer (ask for confirmation)
SPC b h	open <i>spacemacs</i> home buffer
SPC b n	switch to next buffer avoiding special buffers
SPC b m	kill all buffers except the current one
SPC u SPC b m	kill all buffers and windows except the current one
SPC b M	kill all buffers matching the regexp
SPC b p	switch to previous buffer avoiding special buffers
SPC b P	copy clipboard and replace buffer (useful when pasting from a browser)
SPC b R	revert the current buffer (reload from disk)
SPC b s	switch to the <i>scratch</i> buffer (create it if needed)
SPC b w	toggle read-only (writable state)
SPC b Y	copy whole buffer to clipboard (useful when copying to a browser)
z f	Make current function or comments visible in buffer as much as possible

14.5.6.2 Buffers manipulation transient state

一个方便的缓冲区操作瞬态允许通过打开的缓冲区快速循环并杀死它们。

Key Binding	Description
SPC b .	initiate transient state
K	kill current buffer
n	go to next buffer (avoid special buffers)
N	go to previous buffer (avoid special buffers)
Any other key	leave the transient state

14.5.6.3 Special Buffers

不像 vim，emacs 创建了大多数人不需要看的缓冲区。一些例子是 *messages* 和 *compile-log*。Spacemacs 会尝试自动忽略无用的缓冲区。不过，您可能需要更改 spacemacs 将缓冲区标记为有用的方式。有关说明，请参阅

[特殊缓冲区 howto](#)。

14.5.6.4 Files manipulations key bindings

文件操作命令（以f开头）：

Key Binding	Description
SPC f b	go to file bookmarks
SPC f c	copy current file to a different location
SPC f C d	convert file from unix to dos encoding
SPC f C u	convert file from dos to unix encoding
SPC f D	delete a file and the associated buffer (ask for confirmation)
SPC f E	open a file with elevated privileges (sudo edit)
SPC f f	open file with helm
SPC f F	try to open the file under point helm
SPC f h	open binary file with hexl (a hex editor)
SPC f j	jump to the current buffer file in dired
SPC f J	open a junk file, in mode determined by the file extension provided (defaulting to fundamental mode), using helm (or ivy)
SPC f l	open file literally in fundamental mode
SPC f L	Locate a file (using locate)
SPC f o	open a file using the default external program
SPC f R	rename the current file
SPC f s	save a file
SPC f S	save all files
SPC f r	open a recent file with helm
SPC f t	toggle file tree side bar using NeoTree
SPC f v d	add a directory variable
SPC f v f	add a local variable to the current file
SPC f v p	add a local variable to the first line of the current file

Key Binding	Description
SPC f y	show and copy current file absolute path in the minibuffer

14.5.6.5 Emacs and Spacemacs files

方便的键绑定位于前缀 SPC f e 下，以便在 emacs 和 spacemacs 特定文件之间快速导航。

Key Binding	Description
SPC f e d	open the spacemacs dotfile (~/.spacemacs)
SPC f e D	open ediff buffer of ~/.spacemacs and .spacemacs.template
SPC f e f	discover the FAQ using helm
SPC f e i	open the all mighty init.el
SPC f e l	locate an Emacs library
SPC f e R	resync the dotfile with spacemacs
SPC f e v	display and copy the spacemacs version

14.5.6.6 Browsing files with Helm

在 vim 和 hybrid 风格中，Spacemacs 重映射 Helm 查找文件中的导航，以保持在主行上。

Key Binding	Description
C-h	go up one level (parent directory)
C-H	describe key (replace C-h)
C-j	go to previous candidate
C-k	go to next candidate
C-l	enter current directory

14.5.7 Ido

Spacemacs 垂直显示 ido minibuffer 感谢 [ido-vertical-mode](#).

基本的 ido 操作可以用 Ctrl 键完成：

Key Binding	Description
C-	open a dired buffer
M-	open a dired buffer in terminal
C-d	delete selected file (ask for confirmation)

Key Binding	Description
C-h	go to parent directory
C-j	select next file or directory
C-k	select previous file or directory
C-l	open the selected file
C-n	select next file or directory
C-o	open selected file in other window
C-p	select previous file or directory
C-s	open selected file in a vertically split window
C-t	open selected file in a new frame
C-v	open selected file in a horizontally split window
C-S-h	go to previous directory
C-S-j or C-S-n	next history element
C-S-k or C-S-p	previous history element
C-S-l	go to next directory

14.5.8 Ido transient state

Spacemacs 为 ido定义了一个[临时装态](#)。

在一个 ido 缓冲区中用 M-SPC 或 s-M-SPC 启动临时装态。

Key Binding	Description
M-SPC or s-M-SPC	initiate or leave the transient state
?	display help
e	open dired
h	delete backward or parent directory
j	next match
J	sub directory
k	previous match
K	parent directory
l	select match
n	next directory in history
o	open in other window

Key Binding	Description
p	previous directory in history
q	quit transient state
s	open in a new horizontal split
t	open in other frame
v	open in a new vertical split

14.5.9 NeoTree file tree

Spacemacs 提供了一个快速和简单的方法在 [NeoTree](#) 中导航未知的项目文件树。

切换 NeoTree 缓冲区按 SPC f t 或 SPC p t (后者打开 NeoTree, 根目录设置为子弹项目根目录)。

NeoTree 窗口始终有数字 0, 所以它不会移动其他窗口的当前数量。选择 NeoTree 窗口然后使用 spc 0。

支持 VSC 集成, 文件颜色将根据当前状态而改变。用默认的 spacemacs-dark 主题:

- 绿色: 新文件
- 紫色: 修改过的文件

14.5.9.1 NeoTree navigation

导航以 hjkl 键为中心, 希望能提供像 [ranger](#) 一样的快速导航体验:

Key Binding	Description
h	collapse expanded directory or go to parent node
H	select previous sibling
j	select next file or directory
J	select next expanded directory on level down
k	select previous file or directory
K	select parent directory, when reaching the root change it to parent directory
l or RET	expand directory
L	select next sibling
R	make a directory the root directory

注意: 点自动设置为节点的第一个字母, 以获得更流畅的体验。

14.5.9.2 Opening files with NeoTree

默认情况下在最后一个活动窗口中打开一个文件。可以通过使用数字参数来选择要打开文件的窗口号, 例如 2 l 或 2 RET 将打开窗口 2 中的当前文件。也可以使用 | 和 - :

Key Binding	Description
l or RET	open file in last active window
##### l or # RET	open file in window number #
	open file in an vertically split window
-	open file in an horizontally split window

14.5.9.3 Other NeoTree key bindings

Key Binding	Description
TAB	toggle stretching of the buffer
c	create a node
d	delete a node
gr	refresh
s	toggle showing of hidden files
q or fd	hide NeoTree buffer
r	rename a node
?	show help

14.5.9.4 NeoTree mode-line

mode-line 具有以下格式 [x/y] d (d: a, f: b) 其中:

- x是当前所选文件或目录的索引
- y当前目录中项目（文件和目录）的总数
- d当前目录的名称
- a当前目录中的目录数量
- b当前目录中的文件数量

14.5.9.5 NeoTree Source Control Integration

如果你想 NeoTree 显示源控制信息，你可以使用 neo-vc-integration 设置。它是一个包含可能值的列表：

Setting	Description
face	通过改变文件/目录名称的颜色来显示信息。
char	在文件/目录名称的左侧显示带有字符的信息。

默认是 nil（不显示源控制信息），这是推荐的。

例如，

```
(setq neo-vc-integration 'face)
```

注意：目前，不建议将其设置为除 nil 之外的任何值。否则，源码树越大，速度越慢。有关更多信息，请参阅 <https://github.com/jaypei/emacs-neotree/issues/126>。

14.5.9.6 NeoTree Theme

你可以使用设置 neo-theme 来改变 NeoTree 主题。可能的值是：

Setting	Description
classic	使用图标显示项目 - 只适用于 gui 模式。
ascii	最简单的样式，它将使用x, - 来显示折叠状态。
arrow	使用 unicode 箭头来显示折叠状态。
nerd	使用 NERDTree 缩进模式和箭头。

默认是 classic。使用 nerd，如果你想在 VIM 看起来最像 NERDTree。例如：

```
(setq neo-theme 'nerd)
```

14.5.10 Bookmarks

书签可以在文件的任何地方设置。书签是持久的。他们是非常有用的跳转到/打开一个已知的项目。Spacemacs 使用 helm-bookmarks 来管理它们。

按下：SPC f b，打开当前书签的 helm 窗口

然后在 helm-bookmarks 缓冲区中：

Key Binding	Description
C-d	删除选中的书签
C-e	编辑选定的书签
C-f	切换文件名位置
C-o	在另一个窗口中打开选定的书签

要保存新的书签，只需输入书签的名称，然后按下 RET。

14.5.11 DocView mode

doc-view-mode 是一个内置的主要模式来查看 DVI, PostScript (PS), PDF, OpenDocument 和 Microsoft Office 文档。

Key Binding	Description
/	search forward
?	search backward
+	enlarge
-	shrink
gg	go to first page
G	go to last page
gt	go to page number
h	previous page
H	adjust to height
j	next line
k	previous line
K	kill proc and buffer
l	next page
n	go to next search occurrence
N	go to previous search occurrence
P	fit page to window
r	revert
W	adjust to width
C-d	scroll down
C-k	kill proc
C-u	scroll up
C-c C-c	toggle display text and image display
C-c C-t	open new buffer with doc's text contents

14.6 Auto-saving

14.6.1 Frequency of auto-saving

默认情况下，每300个字符和每30秒的空闲时间执行文件自动保存，可以通过分别将变量 `auto-save-inteval` 和 `auto-save-timeout` 设置为新值来更改这些文件。

14.6.2 Location of auto-saved files

修改文件的自动保存可以在原始文件本身或缓存目录中原位执行（在这种情况下，原始文件将保持未保存状态）。默认情况下，Spacemacs 自动将文件保存在缓存目录中。

修改位置将变量 `dotspacemacs-auto-save-file-location` 设置为 `original` 或 `cache`。

本地文件将自动保存在缓存目录中的一个名为站点的子目录中，而远程文件（即通过 TRAMP 编辑的文件）将自动保存在名为 `dist` 的子目录中。

14.6.3 Disable auto-save

禁用自动保存将变量 `dotspacemacs-auto-save-file-location` 设置为 `nil`。

您可以通过调用命令 `auto-save-mode` 来切换自动保存在缓冲区中。

14.7 Searching

14.7.1 With an external tool

Spacemacs 可以与不同的搜索工具连接，如：

- `ack`
- `grep`
- `ag`
- `pt`

Spacemacs 中的搜索命令在 `SPC s` 前缀下组织，下一个键是要使用的工具，最后一个键是范围。比如 `SPC s a b` 会使用 `ag` 在所有打开的缓冲区中搜索。

如果最后一个键（确定范围）是大写的，那么点下的当前区域或符号将被用作搜索的默认输入。例如 `SPC s a B` 将在符号下搜索符号（如果没有活动区域）。

如果省略了工具键，则会自动选择默认工具进行搜索。这个工具对应于在列表 `dotspacemacs-search-tools` 的系统上找到的第一个工具，默认顺序是 `ag`，`pt`，`ack`，然后是 `grep`。例如，如果在系统中没有找到 `ag`，`spc` 将使用 `pt` 在打开的缓冲区中搜索。

工具键是：

Tool	Key
ag	a
grep	g
ack	k
pt	t

可用的范围和相应的键是：

Scope	Key
打开缓冲区	b

Scope	Key
文件在给定的目录中	f
当前的项目	p

可以通过双击序列的第二个键在当前文件中进行搜索，例如 SPC s a a 将使用 ag 在当前文件中搜索。

Notes:

- ag 和 pt 被优化用于源代码控制库中，但是它们也可以在任意目录中使用。
- 也可以通过在 helm 缓冲区中标记它们来一次搜索多个目录。

当心如果你使用 pt, [TCL parser tools](#) 也会安装一个名为 pt 的命令行工具。

14.7.1.1 Useful key bindings

Key Binding	Description
F3	在 helm 或 ivy 缓冲区中，将结果保存到常规缓冲区
SPC r l	恢复上一个完成缓冲区
SPC r s or SPC s l	恢复搜索缓冲区（完成或转换的搜索缓冲区）
SPC s `	回到以前用 helm-ag 达到的地方
Prefix argument	将要求文件扩展名

当结果保存在 f3 的常规缓冲区中时，该缓冲区支持通过与 Spacemacs 的下一个错误和先前错误绑定（SPC e n 和 SPC e p）以及错误瞬态（SPC e）进行浏览。

14.7.1.2 Searching in current file

Key Binding	Description
SPC s s	用第一个找到的工具搜索
SPC s S	用默认输入搜索第一个找到的工具
SPC s a a	ag
SPC s a A	ag with default input
SPC s g g	grep
SPC s g G	grep with default input

14.7.1.3 Searching in all open buffers visiting files

Key Binding	Description
SPC s b	用第一个找到的工具搜索

Key Binding	Description
SPC s B	用默认输入搜索第一个找到的工具
SPC s a b	ag
SPC s a B	ag with default text
SPC s g b	grep
SPC s g B	grep with default text
SPC s k b	ack
SPC s k B	ack with default text
SPC s t b	pt
SPC s t B	pt with default text

14.7.1.4 Searching in files in an arbitrary directory

Key Binding	Description
SPC s f	用第一个找到的工具搜索
SPC s F	用默认输入搜索第一个找到的工具
SPC s a f	ag
SPC s a F	ag with default text
SPC s g f	grep
SPC s g F	grep with default text
SPC s k f	ack
SPC s k F	ack with default text
SPC s t f	pt
SPC s t F	pt with default text

14.7.1.5 Searching in a project

Key Binding	Description
SPC / or SPC s p	用第一个找到的工具搜索
SPC * or SPC s P	用默认输入搜索第一个找到的工具
SPC s a p	ag
SPC s a P	ag with default text
SPC s g p	grep with default text

Key Binding	Description
SPC s k p	ack
SPC s k P	ack with default text
SPC s t p	pt
SPC s t P	pt with default text

提示：也可以在项目中搜索，而无需事先打开文件。在一个给定的项目中使用 SPC p p 然后 C-s 来直接搜索它，就像使用 SPC s p 一样。

14.7.1.6 Searching the web

Key Binding	Description
SPC s w g	在 emacs 中获得谷歌建议。在浏览器中打开google结果。
SPC s w w	在 emacs 中获得维基百科建议。在浏览器中打开维基百科页面。

14.7.2 Persistent highlighting

Spacemacs 使用 evil-search-highlight-persist 来保持搜索到的表达式直到下一次搜索。也可以通过按 SPC s c 或执行 ex 命令来清除突出显示：noh。

14.7.3 Highlight current symbol

Spacemacs 支持按需高亮显示当前符号（由 [auto-highlight-symbol](#) mode 提供），并添加瞬态以轻松导航并重命名此符号。

也可以将飞行中的导航范围更改为：

- buffer
- function
- visible area

在点按 SPC s h 下启动当前符号的突出显示。

高亮的符号之间的导航可以通过以下命令完成：

Key Binding	Description
*	在当前符号上启动导航暂态，并向前跳转
####	在当前符号上启动导航暂态并向后跳转
SPC s e	编辑所有出现的当前符号（/）
SPC s h	高亮显示当前符号及其在当前范围内的所有出现
SPC s H	转到上次搜索到的最后一个突出显示的符号
SPC t h a	在 ahs-idle-interval 秒后切换点下的符号自动高亮

在'spacemacs'中高亮显示符号瞬态：

Key Binding	Description
e	edit occurrences (*)
n	go to next occurrence
N	go to previous occurrence
d	go to next definition occurrence
D	go to previous definition occurrence
r	change range (function, display area, whole buffer)
R	go to home occurrence (reset position to starting occurrence)
Any other key	leave the navigation transient state

(*) 使用 [iedit](#) 或 `auto-highlight-symbol` 的默认实现

minibuffer 中的瞬态文本显示如下信息：

```
<M> [6/11]* press (n/N) to navigate, (e) to edit, (r) to change range or
(R)
for reset
```

其中 [x/y] *是：

- M：当前的范围模式
- ：整个缓冲区范围
- ：当前的显示范围
- ：当前的功能范围
- x：当前高亮显示的事件的索引
- y：发生的总次数
- *：如果至少有一个当前不可见的情况出现。

14.7.4 Visual Star

用 [evil-visualstar](#) 你可以搜索下一个出现的当前选择。

它与 [expand-region](#) 绑定相结合非常有用。

注意：如果当前状态不是 visual 状态，那么按 * 使用自动高亮符号及其瞬态状态。

14.7.5按语义列出符号

使用 helm 的 `helm-semantic-or-imenu` 命令可以在缓冲区中的符号之间快速导航。

列出一个缓冲区的所有符号按：SPC s j

14.7.6 Helm-swoop

这与 moccur 非常相似，它显示了一个 helm 缓冲区，其中包含所有出现的单词。您可以实时更改搜索查询并轻松地在它们之间导航。

您甚至可以直接在 helm 缓冲区中编辑事件，并将修改应用到缓冲区。

Key Binding	Description
SPC s s	execute helm-swoop
SPC s S	execute helm-multi-swoop
SPC s C-s	execute helm-multi-swoop-all

14.8编辑

14.8.1粘贴文本

14.8.1.1粘贴瞬态

可以通过将变量 dotspacemacs-enable-paste-transient-state 设置为 t 来启用粘贴瞬变状态。默认情况下它被禁用。

当启用暂态时，再次按下 p 将会替换粘贴的文字与先前在 kill (ring) 上被 yanked (copied) 的文字。

例如，如果复制 foo 和 bar，然后按 p 将粘贴文本栏，再次按 p 将用 foo 替换 bar。

Key Binding	Description
p or P	粘贴文本之前或之后的文本，并启动粘贴瞬态
p	在瞬态：用先前复制的粘贴文本替换
P	在瞬态中：用下一个复制的文本替换粘贴文本
.	粘贴相同的文本，并离开瞬态
Any other key	离开瞬态

14.8.1.2自动缩进粘贴文本

默认情况下，任何粘贴的文本都将被自动缩进。粘贴文本不加缩进使用通用的参数。

可以通过在您的 dotspacemacs/user-config 函数的变量 spacemacs-indent-sensitive-modes 中添加 major-modes 来禁用特定 major-mode 的自动缩进。

14.8.2文本操作命令

文本相关的命令（以x开头）：

Key Binding	Description
-------------	-------------

Key Binding	Description
SPC x a &	align region at &
SPC x a (align region at (
SPC x a)	align region at)
SPC x a ,	align region at ,
SPC x a .	align region at . (for numeric tables)
SPC x a :	align region at :
SPC x a ;	align region at ;
SPC x a =	align region at =
SPC x a a	align region (or guessed section) using default rules
SPC x a c	align current indentation region using default rules
SPC x a r	align region using user-specified regexp
SPC x a m	align region at arithmetic operators (+-*/)
SPC x a	align region at
SPC x c	count the number of chars/words/lines in the selection region
SPC x d w	delete trailing whitespaces
SPC x g l	set languages used by translate commands
SPC x g t	translate current word using Google Translate
SPC x g T	reverse source and target languages
SPC x j c	set the justification to center
SPC x j f	set the justification to full
SPC x j l	set the justification to left
SPC x j n	set the justification to none
SPC x j r	set the justification to right
SPC x J	move down a line of text (enter transient state)
SPC x K	move up a line of text (enter transient state)
SPC x l s	sort lines
SPC x l u	uniquify lines
SPC x o	use avy to select a link in the frame and open it
SPC x O	use avy to select multiple links in the frame and open them
SPC x t c	swap (transpose) the current character with the previous one

Key Binding	Description
SPC x t w	swap (transpose) the current word with the previous one
SPC x t l	swap (transpose) the current line with the previous one
SPC x u	set the selected text to lower case
SPC x U	set the selected text to upper case
SPC x w c	count the number of occurrences per word in the select region
SPC x w d	show dictionary entry of word from wordnik.com
SPC x TAB	indent or dedent a region rigidly

14.8.3 文本插入命令

文本插入命令（从 i 开始）：

Key binding	Description
SPC i l l	insert lorem-ipsum list
SPC i l p	insert lorem-ipsum paragraph
SPC i l s	insert lorem-ipsum sentence
SPC i u	Search for Unicode characters and insert them into the active buffer.
SPC i U 1	insert UUIDv1 (use universal argument to insert with CID format)
SPC i U 4	insert UUIDv4 (use universal argument to insert with CID format)
SPC i U U	insert UUIDv4 (use universal argument to insert with CID format)

14.8.4 Smartparens Strict mode

[Smartparens](#)有一个严格的模式，如果结果不平衡，可以防止删除括号。

这种模式可能会让新手感到沮丧，这就是为什么它没有默认启用。

可以通过使用 `~/spacemacs` 的变量 `dotspacemacs-smartparens-strict-mode` 来轻松启用所有编程模式。

```
(setq-default dotspacemacs-smartparens-strict-mode t)
```

14.8.5 缩放(Zooming)

14.8.5.1 文本

当前缓冲区的字体大小可以通过以下命令进行调整：

Key Binding	Description
-------------	-------------

Key Binding	Description
SPC z x +	scale up the font and initiate the font scaling transient state
SPC z x =	scale up the font and initiate the font scaling transient state
SPC z x -	scale down the font and initiate the font scaling transient state
SPC z x 0	reset the font size (no scaling) and initiate the font scaling transient state
+	increase the font size
=	increase the font size
-	decrease the font size
0	reset the font size
Any other key	leave the font scaling transient state

请注意，只有当前缓冲区的文本被缩放，其他缓冲区，模式行和小缓冲区不受影响。使用缩放框架绑定缩放框架的全部内容（请参阅下一节）。

14.8.5.2 框架(Frame)

您可以使用以下命令放大和缩小框架的全部内容：

Key Binding	Description
SPC z f +	放大框架内容并启动框架缩放瞬态
SPC z f =	放大框架内容并启动框架缩放瞬态 scaling transient state
SPC z f -	缩小框架内容并启动框架缩放瞬态
SPC z f 0	重置框架内容大小并启动框架缩放瞬态
+	放大
=	放大
-	缩小
0	重置缩放
任何其他键	保持缩放框架的瞬态状态

14.8.6增加/减少数字

Spacemacs 使用 [evil-number](#) 来轻松地增加或减少数字。

Key Binding	Description
SPC n +	increase the number under point by one and initiate transient state
SPC n -	decrease the number under point by one and initiate transient state

在瞬态中：

Key Binding	Description
+	increase the number under point by one
-	decrease the number under point by one
Any other key	leave the transient state

提示：你可以通过使用前缀参数多次增加或减少一个值（即，10 SPC n + 将点数加10）。

14.8.7 拼写检查

通过在你的 dotfile 中包含 spell checking layer 来启用拼写检查。
键绑定列在 layer 文档中。

14.8.8区域选择

evil 支持 Vi 中所有的 Visual modes

14.8.8.1 扩大区域(Expand-region)

Spacemacs 通过 [expand-regin](#) mode 添加了另一种 Visual mode。

Key Binding	Description
SPC v	启动 expand-regin 模式，然后...
v	用一个语义单位来扩展该区域
V	通过一个语义单元来收缩区域
r	将区域重置为初始选择
ESC	离开 expand-regin 模式

14.8.8.2缩进文本对象(Indent text object)

[evil-indent-plus](#) 以下文本对象可用：

- ii - 内部缩进：周围的文本块具有相同的缩进
- il - 上面和缩进：ii + 上面的行用不同的缩进
- iJ - 上面，下面和缩进+：ii +下面的行用不同的缩进

还有一个包含空白的变体.例子（| 表示点）：

```
(while (not done)
  (message "All work and no play makes Jack a dull boy."))
(1+ 41)
```

- vii 会选择带消息的行
- vil 会选择整个while循环
- viJ 将选择整个片段

14.8.9 区域缩小(Region narrowing)

缓冲区的显示文本可以用命令缩小（从 n 开始）：

Key Binding	Description
SPC n f	将缓冲区缩小到当前的功能
SPC n p	将缓冲区缩小到可见页面
SPC n r	将缓冲区缩小到选定的文本
SPC n w	扩大，即再次显示整个缓冲区

14.8.10 用iedit替换文本(Replacing text with iedit)

Spacemacs 通过 evil-iedit-state 使用强大的 iedit 模式来快速编辑多个符号或选项。

evil-iedit-state 定义了两个新的 evil stataes：

- iedit state
- iedit-insert state

这些状态的颜色代码是红色的。

evil-iedit-state 与 expand-region 也有很好的集成，通过按 e 可快速编辑当前选定的文本。

14.8.10.1 iedit states 键绑定(iedit states key bindings)

1. 状态转换
- 2.

Key Binding	From	To
SPC s e	normal or visual	iedit
e	expand-region	iedit
ESC	iedit	normal
C-g	iedit	normal
fd	iedit	normal
ESC	iedit-insert	iedit
C-g	iedit-insert	normal
fd	iedit-insert	norma

总结，在 `iedit-insert` 状态，你必须按两次 `ESC` 回到 `normal` 状态。你也可以随时按 `C-g` 或 `fd` 回到 `normal` 状态。

注意：切换到 `insert` 状态的 `evil` 命令会切换到 `iedit-insert` 状态。

2. 在 `iedit` 状态 `iedit` 状态从 `normal` 状态继承，下面的键绑定是特定于 `iedit` 状态的。

Key Binding	Description
<code>ESC</code>	go back to normal state
<code>TAB</code>	toggle current occurrence
<code>0</code>	go to the beginning of the current occurrence
<code>\$</code>	go to the end of the current occurrence

prefix all occurrences with an increasing number (`SPC u` to choose the starting number).

A	go to the end of the current occurrence and switch to <code>iedit-insert</code> state
<code>D</code>	delete the occurrences
<code>F</code>	restrict the scope to the function
<code>gg</code>	go to first occurrence
<code>G</code>	go to last occurrence
<code>I</code>	go to the beginning of the current occurrence and switch to <code>iedit-insert</code> state
<code>J</code>	increase the editing scope by one line below
<code>K</code>	increase the editing scope by one line above
<code>L</code>	restrict the scope to the current line
<code>n</code>	go to next occurrence
<code>N</code>	go to previous occurrence
<code>p</code>	replace occurrences with last yanked (copied) text
<code>S</code>	(substitute) delete the occurrences and switch to <code>iedit-insert</code> state
<code>V</code>	toggle visibility of lines with no occurrence
<code>U</code>	Up-case the occurrences
<code>C-U</code>	down-case the occurrences

注意：`0`，`$`，`a`和我有默认的 `vim` 行为在事件之外使用。

3. 在 `iedit-insert` 状态

Key Binding	Description
-------------	-------------

Key Binding	Description
ESC go	back to iedit state
C-g go	back to normal state

14.8.10.2 例子

- 手动选择几个字然后替换: `v w w SPC s e S "toto" ESC ESC`
- 将文本附加到两行的单词上: `v i w SPC s e J i "toto" ESC ESC`
- 用 `expand-region` 替代符号: `SPC v v e S "toto" ESC ESC`
- 用扩展区域替换符号与被抽出 (复制) 的文本: `SPC v e p ESC ESC`

14.8.11 替换几个文件中的文本

如果您安装了 `ag`, `pt` 或 `ack`, 则可以通过 [helm-ag](#) 来替换多个文件中的文本。

假设你想在当前项目中用 `bar` 替换所有的 `foo` 事件:

- 用 `SPC /` 开始搜索
- 用 `C-c C-e` 编辑模式进入
- 用 `SPC s e` 去发生并进入 `iedit state`
- 编辑事件, 然后离开 `iedit state`
- 按 `C-c C-c`

注意: 在 Spacemacs 中, 尽管 `helm-ag` 的名字与 `ack` 和 `pt` 一起工作 (但不与 `grep` 一起)。

14.8.12 重命名目录中的文件

可以使用 `helm` 会话中的 `wdired` 批量重命名目录中的文件:

- 使用 `SPC f f` 浏览目录
- 用 `C-c C-e` 进入 `wdired`
- 编辑文件名并使用 `C-c C-c` 确认更改
- 使用 `C-c C-k` 放弃所有更改

14.8.13 评论

评论由 [evil-nerd-commenter](#) 处理, 它被绑定到下面的键。

Key Binding	Description
SPC ;	comment operator
SPC c l	comment lines
SPC c L	invert comment lines
SPC c p	comment paragraphs
SPC c P	invert comment paragraphs

Key Binding	Description
SPC c t	comment to line
SPC c T	invert comment to line
SPC c y	comment and yank
SPC c Y	invert comment and yank

提示：有效地评论一行使用组合 SPC ; SPC y

14.8.14 正则表达式

Spacemacs 使用包 [pcre2el](#) 来操作正则表达式。在使用 Emacs Lisp 缓冲区时它很有用，因为它允许轻松地将 PCRE (Perl 兼容 RegExp) 转换为 Emacs Regexp 或 rx。它也可以用来“解释”一个 PCRE RegExp 以 rx 形式表示的点。

键绑定以 SPC x r 开始并具有以下助记符结构：

- SPC x r 从源代码转换为目标代码
- spc x r 做我的意思

Key Binding	Function
SPC x r /	Explain the regexp around point with rx
SPC x r '	Generate strings given by a regexp given this list is finite
SPC x r t	Replace regexp around point by the rx form or vice versa
SPC x r x	Convert regexp around point in rx form and display the result in the minibuffer
SPC x r c	Convert regexp around point to the other form and display the result in the minibuffer
SPC x r e /	Explain Emacs Lisp regexp
SPC x r e '	Generate strings from Emacs Lisp regexp
SPC x r e p	Convert Emacs Lisp regexp to PCRE
SPC x r e t	Replace Emacs Lisp regexp by rx form or vice versa
SPC x r e x	Convert Emacs Lisp regexp to rx form
SPC x r p /	Explain PCRE regexp
SPC x r p '	Generate strings from PCRE regexp
SPC x r p e	Convert PCRE regexp to Emacs Lisp
SPC x r p x	Convert PCRE to rx form

14.8.15 删除文件

删除配置为将删除的文件发送到系统垃圾箱。

在 OS x 上需要垃圾程序。它可以通过 homebrew 以下命令与自制软件一起安装：

```
$ brew install trash
```

要禁用垃圾箱，您可以在 `~/spacemacs` 中将变量 `delete-by-moving-to-trash` 设置为 `nil`。

14.8.16 编辑 lisp 代码

lisp 代码的编辑由 `evil-lisp-state` 提供。

命令会将当前状态设置为 lisp 状态，其中可以重复不同的命令组合，而无需按下 SPC k。

当处于 lisp 状态时，mode-line 的颜色变为粉红色。

例子：

- 在正常状态下啜饮三次：SPC k 3 s
- 在圆括号中包装一个符号，然后啜饮两次：SPC k w 2 s

注意：lisp 状态命令可以在任何模式下使用！试试看。

14.8.16.1 lisp 键绑定

1. lisp 状态键绑定 这些命令自动切换到 lisp 状态。

Key Binding	Function
SPC k %	evil jump item
SPC k :	ex command
SPC k (insert expression before (same level as current one)
SPC k)	insert expression after (same level as current one)
SPC k \$	go to the end of current sexp
SPC k ` k	hybrid version of push sexp (can be used in non lisp dialects)
SPC k ` p	hybrid version of push sexp (can be used in non lisp dialects)
SPC k ` s	hybrid version of slurp sexp (can be used in non lisp dialects)
SPC k ` t	hybrid version of transpose sexp (can be used in non lisp dialects)
SPC k 0	go to the beginning of current sexp
SPC k a	absorb expression
SPC k b	forward barf expression
SPC k B	backward barf expression
SPC k c	convolute expression

Key Binding	Function
SPC k ds	delete symbol
SPC k Ds	backward delete symbol
SPC k dw	delete word
SPC k Dw	backward delete word
SPC k dx	delete expression
SPC k Dx	backward delete expression
SPC k e	unwrap current expression and kill all symbols after point
SPC k E	unwrap current expression and kill all symbols before point
SPC k h	previous symbol
SPC k H	go to previous sexp
SPC k i	switch to insert state
SPC k I	go to beginning of current expression and switch to insert state
SPC k j	next closing parenthesis
SPC k J	join expression
SPC k k	previous opening parenthesis
SPC k l	next symbol
SPC k L	go to next sexp
SPC k p	paste after
SPC k P	paste before
SPC k r	raise expression (replace parent expression by current one)
SPC k s	forward slurp expression
SPC k S	backward slurp expression
SPC k t	transpose expression
SPC k u	undo
SPC k U	got to parent sexp backward
SPC k C-r	redo
SPC k v	switch to visual state
SPC k V	switch to visual line state
SPC k C-v	switch to visual block state
SPC k w	wrap expression with parenthesis

Key Binding	Function
SPC k W	unwrap expression
SPC k y	copy expression

2. emacs lisp 特定的键绑定

Key Binding	Function
SPC m e \$	go to end of line and evaluate last sexp
SPC m e b	evaluate buffer
SPC m e c	evaluate current form (a def or a set)
SPC m e e	evaluate last sexp
SPC m e f	evaluate current defun
SPC m e l	go to end of line and evaluate last sexp

Key Binding	Function
SPC m g g	go to definition
SPC m g G	go to definition in another window
SPC m h h	describe elisp thing at point (show documentation)
SPC m t b	execute buffer tests
SPC m t q	ask for test function to execute

14.8.17 鼠标使用情况

有一些添加的鼠标功能为行号边距设置（如果显示）：

- 在行号边界中单击可直观地选择整个行
- 拖过行号边距可以直观地选择区域
- 双击行号边界，直观地选择当前的代码块

14.9 管理项目

在 Spacemacs 中的项目是用 [projectile](#) 来管理的。在 projectile 项目中是隐式定义的，例如，当在文件树中遇到 .git 存储库或 .projectile 文件时找到项目的根。

只要有可能，就使用 Helm。

在项目中搜索查看 [project searching](#)。

projectile 命令以 p 开头：

Key Binding	Description
-------------	-------------

Key Binding	Description
SPC p '	open a shell in project's root (with the shell layer)
SPC p !	run shell command in project's root
SPC p &	run async shell command in project's root
SPC p %	replace a regexp
SPC p a	toggle between implementation and test
SPC p b	switch to project buffer
SPC p c	compile project using projectile
SPC p d	find directory
SPC p D	open project root in dired
SPC p f	find file
SPC p F	find file based on path around point
SPC p g	find tags
SPC p C-g	regenerate the project's etags / gtags
SPC p h	find file using helm
SPC p I	invalidate the projectile cache
SPC p k	kill all project buffers
SPC p o	run multi-occur
SPC p p	switch project
SPC p r	open a recent file
SPC p R	replace a string
SPC p t	open NeoTree in projectile root
SPC p T	test project
SPC p v	open project root in vc-dir or magit
SPC /	search in project with the best search tool available
SPC s p	see search in project
SPC s a p	run ag
SPC s g p	run grep
SPC s k p	run ack
SPC s t p	run pt

注意windows用户：启用快速索引 GNU find 或 Cygwin find 必须要在你的 PATH 中。

14.10 寄存器(Registers)

对各个寄存器的访问命令以r开头:

Key Binding	Description
SPC r e	show evil yank and named registers
SPC r m	show marks register
SPC r r	show helm register
SPC r y	show kill ring

14.11 错误处理




Spacemacs 使用 [Flycheck](#) 实时提供错误反馈。检查仅在保存时间默认情况下执行。

错误管理命令 (以e开头) :

Key Binding	Description
SPC t s	toggle flycheck
SPC e c	clear all errors
SPC e h	describe a flycheck checker
SPC e l	toggle the display of the flycheck list of errors/warnings
SPC e n	go to the next error
SPC e p	go to the previous error
SPC e v	verify flycheck setup (useful to debug 3rd party tools configuration)
SPC e .	error transient state

下一个/上一个错误绑定和错误瞬态可用于浏览来自 flycheck 的错误以及来自编译缓冲区的错误，甚至可以用来支持 emacs 的下一个错误 API 的任何事情。这包括例如已经被保存到单独的缓冲区的搜索结果。

自定义边缘位图:

Symbol	Description
	Error
	warning
	Info

14.12 编译

Spacemacs 绑定了一些命令来支持编译项目。

Key Binding	Description
-------------	-------------

Key Binding	Description
SPC c c	use helm-make via projectile
SPC c C	compile
SPC c d	close compilation window
SPC c k	kill compilation
SPC c m	helm-make
SPC c r	recompile

14.13 模式(Modes)

4.13.1 主模式 leader 键(Major Mode leader key)

特定于当前主要模式的键绑定以spc m开头。为方便起见，默认情况下会设置一个称为主模式 leader 键的快捷键，这可以节省一笔宝贵的击键。

可以通过在 `~/.spacemacs` 中定义变量 `dotspacemacs-major-mode-leader-key` 来更改主模式的 leader 键。例如在列表中设置键：

```
(setq-default dotspacemacs-major-mode-leader-key "<tab>")
```

14.13.2 Helm

Spacemacs 将 hjkl 导航添加到 helm 缓冲区：

Key Binding	Description
C-h	go to next source
C-H	describe key (replace C-h)
C-j	go to previous candidate
C-k	go to next candidate
C-l	same as return

14.14 emacs服务器

Spacemacs 在启动时启动服务器。这个服务器在你关闭你的 emacs 窗口时会被终止。

14.14.1 连接到 emacs 服务器

您可以使用 `emacsclient` 从终端在 emacs 中打开一个文件。使用 `emacsclient -c` 在 Emacs GUI 中打开文件。使用 `emacsclient -t` 在终端内的 Emacs 中打开文件。

如果您希望您的 Linux/OS X 系统默认使用 Emacs 作为任何提示，您需要将其设置为您的 shell 配置，例如，`~/bashrc` 或 `~/zshrc`：

```
export EDITOR="emacsclient -c"
```

请注意，如果您使用 OS X，则可能需要引用您的 GUI Emacs 附带的 `emacsclient`，例如：

```
export EDITOR="/Applications/Emacs.app/Contents/MacOS/bin/emacsclient -c"
```

提示：记得在 emacs 中编辑文件后使用：`wq` 或 `C-x #`。

有关更多详细信息，请参阅官方 Emacs 手册中的 [Emacs 作为服务器](#)。

14.15 保持服务器开着

通过在 `~/spacemacs` 中将变量 `dotspacemacs-persistent-server` 设置为 `t` 关闭 Emacs，可以使服务器保持活动状态。

```
(setq-default dotspacemacs-persistent-server t)
```

当此变量设置为 `t` 时，退出 Emacs 并终止服务器的唯一方法是使用以下绑定：

Keybinding	Description
SPC q	Quit Emacs and kill the server, prompt for changed buffers to save
SPC q Q	Quit Emacs and kill the server, lose all unsaved changes.
SPC q r	Restart both Emacs and the server, prompting to save any changed buffers
SPC q s	Save the buffers, quit Emacs and kill the server
SPC q z	Kill the current frame

14.16 故障排除(Troubleshoot)

14.16.1 Loading fails

如果在加载过程中发生错误，mode-line 将变为红色，并且错误应该在启动缓冲区中内联显示。Spacemacs 应该仍然可用；如果不是，那么用 `emacs --debug-init` 重新启动 Emacs，并用 `backtrace` 打开一个 [Github issue](#)。

14.16.2 升级/降级 Emacs 版本

为了确保软件包能够针对您安装的新 Emacs 版本进行正确编译，请务必使用 `SPC SPC spacemacs/recompile-elpa` 运行交互式命令 `spacemacs/recompile-elpa`。