

SapceVim中文文档

书栈(BookStack.CN)

目 录

致谢

介绍

核心思想

显著特性

运行截图

谁将从 SpaceVim 中获益？

更新和回滚

用户配置

概念

优雅的界面

常规快捷键

- 窗口管理器

- File Operations

- Editor UI

- Native functions

- Bookmarks management

- Fuzzy finder

- 交互

- 常规操作

- 以 g 为前缀的快捷键

- 以 z 开头的命令

- 搜索

- 编辑

- 错误处理

- 工程管理

EditorConfig

Vim Server

成就

致谢

当前文档《SapceVim中文文档》由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-07-13。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：<http://www.bookstack.cn/books/SpaceVim-zh>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

介绍

SpaceVim

A community-driven vim distribution

关于我们

版本

[SpaceVim](#) 是一个社区驱动的 vim 配置, 支持 vim 和 neovim。SpaceVim 使用模块的方式来组织插件, 对新手更加友好。

目标

- 提供可跨平台的优雅的用户体验。
- 为不同语言开发提供完整的解决方案。

标准

- 兼容 Vim 和 Neovim, 尽量确保 Vim下和 Neovim 下有相同的用户体验
- 兼容 Vim 7.4 或者无 +py/+py3/+lua等特性的Vim
- 不要从 origin 回归。
- 通过权衡成本和收益来决定结果。
- 收益与传统相比较, 当收益是压倒性的时候, 我们选择收益, 而非坚持传统。
- 给可用性一个机会。

鸣谢

- 开发者: [Wang Shidong](#)

contributors 106

- [vimdoc](#) 自动生成 vim 文档
- [Rafael Bodill](#) 的 vim 配置
- [Bailey Ling](#) 的 vim 配置

原文: <https://spacevim.org/cn/about/>

核心思想

核心思想

四大核心思想：记忆辅助，可视化交互，一致性，社区驱动。

如果违背了以上四大核心思想，我们将会尽力修复。

记忆辅助

所有快捷键，根据其功能的不同分为不同的组，以相应的按键作为前缀，例如 **b** 为 buffer 相关快捷键前缀，**p** 为 project 相关快捷键前缀，**s** 为 search 相关快捷键前缀，**h** 为 help 相关快捷键前缀。

可视化交互

创新的实时快捷键辅助系统，以及查询系统，方便快捷查询到可用的模块、插件以及其他更多信息。

一致性

相似的功能使用同样的快捷键，这在 SpaceVim 中随处可见。这得益于明确的约定。其他模块的文档都以此为基础。

社区驱动

社区驱动，保证了 bug 修复的速度，以及新特性更新的速度。

显著特性

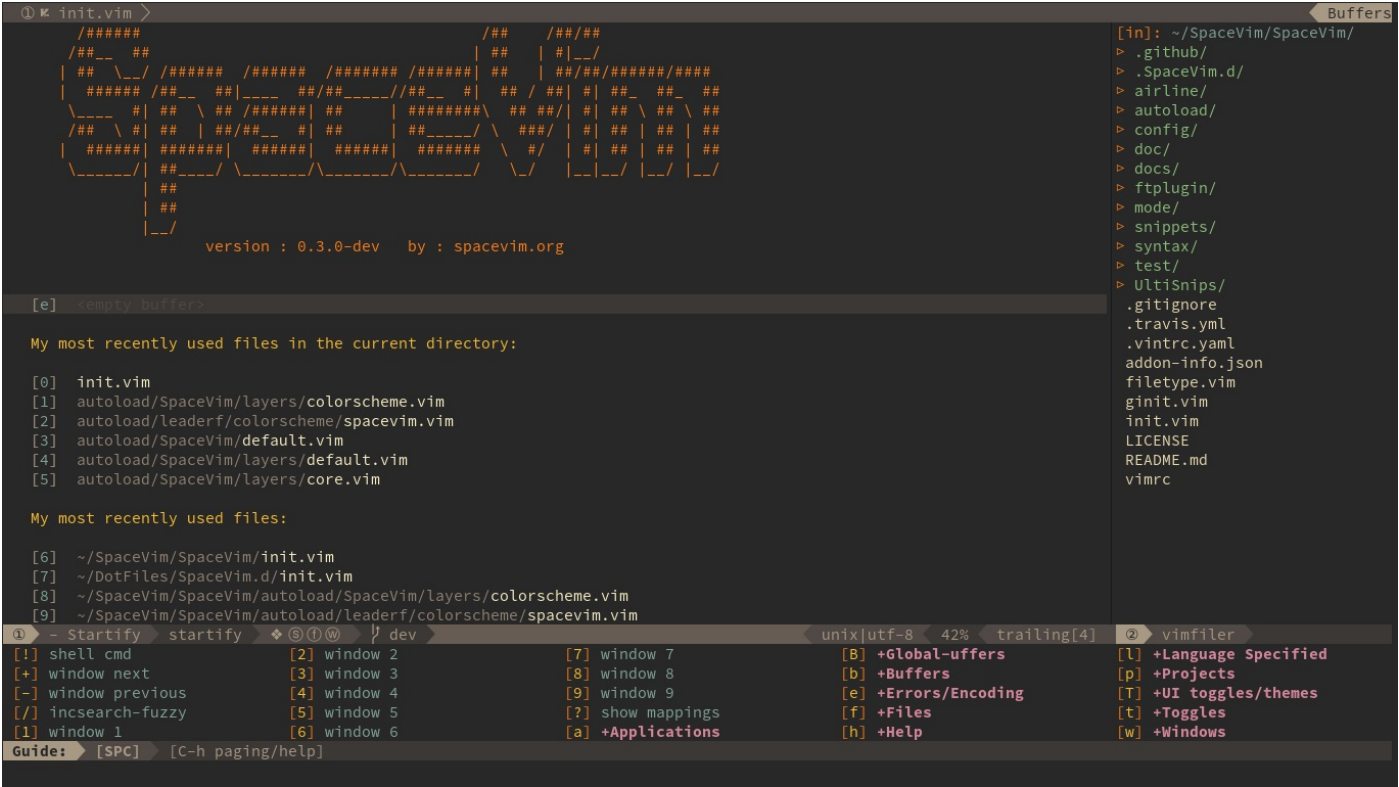
显著特性

- 详细的文档：在 SpaceVim 中通过 `:h SpaceVim` 来访问 SpaceVim 帮助文档。
- 优雅简洁的界面：你将会喜欢这样的优雅而实用的界面。
- 确保手指不离开主键盘区域：使用 Space 作为前缀键，合理组织快捷键，确保手指不离开主键盘区域。
- 快捷键辅助系统：SpaceVim 所有快捷键无需记忆，当输入出现停顿，会实时提示可用按键及其功能。
- 更快的启动时间：得益于 `dein.vim`，SpaceVim 中90% 的插件都是按需载入的。
- 更少的肌肉损伤：频繁使用空格键，取代 `ctrl`，`shift` 等按键，大大减少了手指的肌肉损伤。
- 更易扩展：依照一些[约定](#)，很容易将现有的插件集成到 SpaceVim 中来。
- 完美支持**Neovim**：依赖于 Neovim 的 `romote` 插件以及 异步 API，SpaceVim 运行在 Neovim 下将有更加完美的体验。

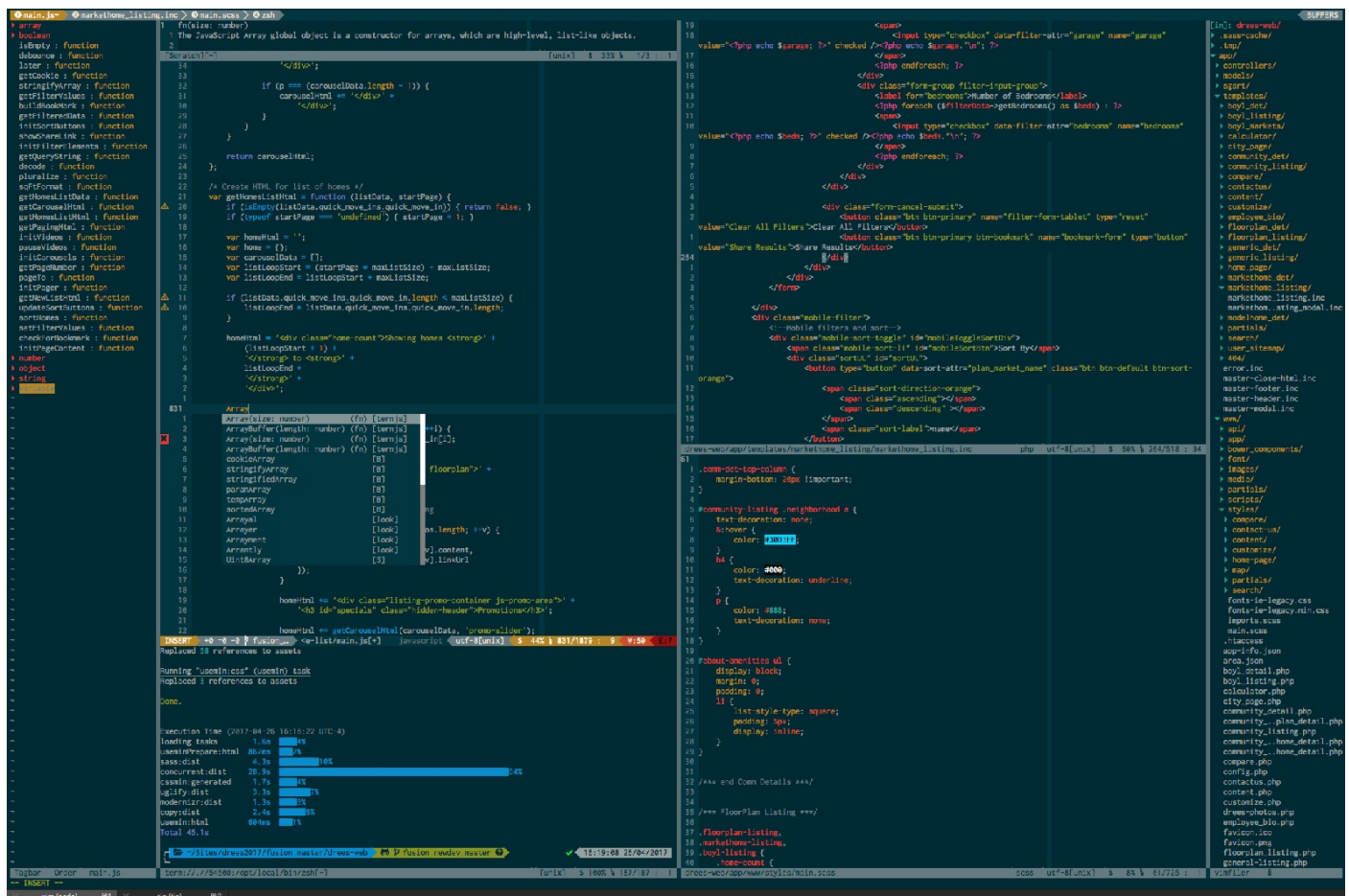
运行截图

运行截图

欢迎页面



工作界面



Neovim 运行在 iTerm2 上，采用 SpaceVim，配色为：base16-solarized-dark

展示了一个通用的前端开发界面，用于开发：JavaScript (jQuery)，SASS，and PHP buffers.

图中包含了一个 Neovim 的终端，一个语法树窗口，一个文件树窗口以及一个 TernJS 定义窗口

想要查阅更多截图，请阅读 [issue #415](#)

谁将从 SpaceVim 中获益？

谁将从 SpaceVim 中获益？

- 初级 Vim 用户。
- 追求优雅界面的 Vim 用户
- 追求更少肌肉损伤的 Vim 用户
- 想要学习一种不一样的编辑文件方式的 Vim 用户
- 追求简单但是可高度配置系统的 Vim 用户

更新和回滚

更新和回滚

SpaceVim 自身更新

可通过很多种方式来更新 SpaceVim 的核心文件。建议在更新 SpaceVim 之前，更新一下所有的插件。具体内容如下：

自动更新

注意：默认，这一特性是禁用的，因为自动更新将会增加 SpaceVim 的启动时间，影响用户体验。如果你需要这一特性，可以将如下加入到用户配置文件中：`let g:spacevim_automatic_update = 1`。

启用这一特性后，SpaceVim 将会在每次启动时候检测是否有新版本。更新后需重启 SpaceVim。

通过插件管理器更新

使用 `:SPUpdate SpaceVim` 这一命令，将会打开 SpaceVim 的插件管理器，更新 SpaceVim，具体进度会在插件管理器 buffer 中展示。

通过 git 进行更新

可通过在 SpaceVim 目录中手动执行 `git pull`，SpaceVim 在 windows 下默认目录为 `~/vimfilers`，但在 Linux 下则可使用如下命令：`git -C ~/.SpaceVim pull`。

更新插件

使用 `:SPUpdate` 这一命令将会更新所有插件，包括 SpaceVim 自身。当然这一命令也支持参数，参数为插件名称，可同时添加多个插件名称作为参数，同时可以使用 `Tab` 键来补全插件名称。

获取日志

使用 `:SPDebugInfo!` 这一命令可以获取 SpaceVim 运行时日志，同时，可以使用 `SPC h I` 使用打开问题模板。可在这个模板中编辑问题，并提交。

用户配置

用户配置

初次启动 SpaceVim 时，他将提供选择目录，用户需要选择合适自己的配置模板。此时，SpaceVim 将自动在 `HOME` 目录生成 `~/.SpaceVim.d/init.toml`。所有用户脚本可以存储在 `~/.SpaceVim.d/`，这一文件夹将被加入 Vim 的运行时路径 `&runtimepath`。详情清阅读 `:h rtp`。

当然，你也可以通过 `SPACEVIMDIR` 这一环境变量，执定用户配置目录。当然也可以通过软连接连改变目录位置，以便配置备份。

SpaceVim 同时还支持项目本地配置，配置初始文件为，当前目录下的 `.SpaceVim.d/init.toml` 文件。同时当前目录下的 `.SpaceVim.d/` 也将被加入到 Vim 运行时路径。

所有的 SpaceVim 选项可以使用 `:h SpaceVim-config` 来查看。选项名称未原先 Vim 脚本中使用的变量名称去处 `g:spacevim_` 前缀。

如果你需要添加自定义以 `SPC` 为前缀的快捷键，你需要使用 bootstrap function，在其中加入：

```
1. call SpaceVim#custom#SPCGroupName(['G'], '+TestGroup')
2. call SpaceVim#custom#SPC('nore', ['G', 't'], 'echom 1', 'echomessage 1', 1)
```

Vim 兼容模式

以下为 SpaceVim 中与 Vim 默认情况下的一些差异，而在兼容模式下，以下所有差异将不存在，可以通过设置 `vimcompatible = true` 来启用 Vim 兼容模式。

- Normal 模式下 `s` 按键不再删除光标下的字符，在 SpaceVim 中，它是 Windows 快捷键的前缀（可以在配置文件中设置成其他按键）。如果希望回复 `s` 按键原先的功能，可以通过 `windows_leader = ""` 使用一个空字符串来禁用这一功能。
- Normal 模式下，按键在 Vim 默认情况下是重复上一次的 `f`、`F`、`t` 和 `T` 按键，但在 SpaceVim 中默认被用作为语言专用的前缀键。如果需要禁用此选项，可设置 `enable_language_specific_leader = false`。
- Normal 模式下 `q` 按键在 SpaceVim 中被设置为了智能关闭窗口，即大多数情况下按下 `q` 键即可关闭当前窗口。可以通过 `windows_smartclose = ""` 使用一个空字符串来禁用这一功能，或修改为其他按键。
- 命令行模式下 按键在 SpaceVim 中被修改为了移动光标至命令行行首。

私有模块

这一部分简单介绍了模块的组成，更多关于新建模块的内容可以阅读SpaceVim 的[模块首页](#)。

目的

使用模块的方式来组织和管理插件，将相关功能的插件组织成一个模块，启用/禁用效率更加高。同时也节省了很多寻找插件和配置插件的时间。

结构

在 SpaceVim 中，一个模块是一个 Vim 文件，比如，`autocomplete` 模块存储在 `autoload/SpaceVim/layers/autocomplete.vim`，在这个文件内有以下几种公共函数：

- `SpaceVim#layers#autocomplete#plugins()`：返回该模块插件列表
- `SpaceVim#layers#autocomplete#config()`：模块相关设置
- `SpaceVim#layers#autocomplete#set_variable()`：模块选项设置函数

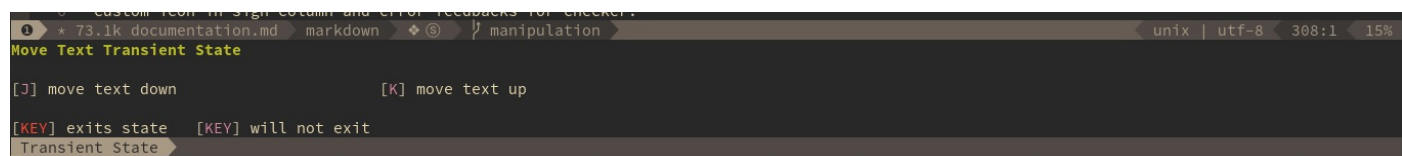
概念

概念

临时快捷键菜单

SpaceVim 根据需要定义了很多临时快捷键，这将避免需要重复某些操作时，过多按下 SPC 前置键。当临时快捷键启用时，会在窗口下方打开一个快捷键介绍窗口，提示每一临时快捷键的功能。此外一些格外的辅助信息也将会体现出来。

文本移动临时快捷键：



优雅的界面

优雅的界面

SpaceVim 集成了多种使用 UI 插件，如常用的文件树、语法树等插件，配色主题默认采用的是 gruvbox。

颜色主题

默认的颜色主题采用的是 gruvbox。这一主题有深色和浅色两种。关于这一主题一些详细的配置可以阅读 `:h gruvbox`。

如果需要修改 SpaceVim 的主题，可以在 `~/.SpaceVim.d/init.toml` 中修改 `colorscheme`。例如，使用 Vim 自带的内置主题 `desert`：

```
1. [options]
2.     colorscheme = "desert"
3.     colorscheme_bg = "dark"
```

快捷键	描述
<code>SPC T n</code>	切换至下一个随机主题
<code>SPC T s</code>	通过 Unite 选择主题

可以在[主题模块](#)中查看 SpaceVim 支持的所有主题。

注意：

SpaceVim 在终端下默认使用了真色，因此使用之前需要确认下你的终端是否支持真色。可以阅读 [Colours in terminal](#) 了解根多关于真色的信息。

如果你的终端不支持真色，可以在 SpaceVim 用户配置 `[options]` 中禁用真色支持：

```
1.     enable_guicolors = false
```

字体

在 SpaceVim 中默认的字体是 DejaVu Sans Mono for Powerline.如果你也喜欢这一字体，建议将这一字体安装到系统中。如果需要修改 SpaceVim 的字体，可以在用户配置文件中修改 `guifont`，默认值为：

```
1.     guifont = "DejaVu\ Sans\ Mono\ for\ Powerline\ 11"
```

如果指定的字体不存在，将会使用系统默认的字体，此外，这一选项在终端下是无效的，终端下修改字体，需要修改终端自身配置。

界面元素切换

大多数界面元素可以通过快捷键来隐藏或者显示（这一组快捷键以 `t` 和 `T` 开头）：

快捷键	描述
<code>SPC t 8</code>	高亮所有超过80列的字符
<code>SPC t f</code>	高亮临界列，默认 <code>max_column</code> 是第 120 列
<code>SPC t h h</code>	高亮当前行
<code>SPC t h i</code>	高亮代码对齐线
<code>SPC t h c</code>	高亮光标所在列
<code>SPC t h s</code>	启用/禁用语法高亮
<code>SPC t i</code>	切换显示当前对齐(TODO)
<code>SPC t n</code>	显示/隐藏行号
<code>SPC t b</code>	切换背景色
<code>SPC t t</code>	打开 Tab 管理器
<code>SPC T ~</code>	显示/隐藏 buffer 结尾空行行首的 <code>~</code>
<code>SPC T F</code>	切换全屏(TODO)
<code>SPC T f</code>	显示/隐藏 Vim 边框(GUI)
<code>SPC T m</code>	显示/隐藏菜单栏
<code>SPC T t</code>	显示/隐藏工具栏

状态栏

`core#statusline` 模块提供了一个高度定制的状态栏，提供如下特性，这一模块的灵感来自于 spacemacs 的状态栏。

- 展示窗口序列号
- 通过不同颜色展示当前模式
- 展示搜索结果序列号
- 显示/隐藏语法检查信息
- 显示/隐藏电池信息
- 显示/隐藏 SpaceVim 功能启用状态
- 显示版本控制信息（需要 git 和 VersionControl 模块）

快捷键	描述
<code>SPC [1-9]</code>	跳至制定序号的窗口

默认主题 gruvbox 的状态栏颜色和模式对照表：

模式	颜色
Normal	灰色
Insert	蓝色

优雅的界面

Visual	橙色
Replace	浅绿色

以上的这几种模式所对应的颜色取决于不同的主题模式。

一些状态栏元素可以进行动态的切换：

快捷键	描述
<code>SPC t m b</code>	显示/隐藏电池状态（需要安装 <code>acpi</code> ）
<code>SPC t m c</code>	toggle the org task clock (available in org layer)(TODO)
<code>SPC t m m</code>	显示/隐藏 SpaceVim 已启用功能
<code>SPC t m M</code>	显示/隐藏文件类型
<code>SPC t m n</code>	toggle the cat! (if colors layer is declared in your dotfile)(TODO)
<code>SPC t m p</code>	显示/隐藏鼠标位置信息
<code>SPC t m t</code>	显示/隐藏时间
<code>SPC t m d</code>	显示/隐藏日期
<code>SPC t m T</code>	显示/隐藏状态栏
<code>SPC t m v</code>	显示/隐藏版本控制信息

nerd 字体安装：

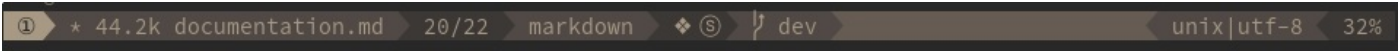
SpaceVim 默认使用 `nerd fonts`，可参阅其安装指南进行安装。

语法检查信息：

状态栏中语法检查信息元素如果被启用了，当语法检查结束后，会在状态栏中展示当前语法错误和警告的数量。

搜索结果信息：

当使用 `/` 或 `?` 进行搜索时，或当按下 `n` 或 `N` 后，搜索结果序号将被展示在状态栏中，类似于 `20/22` 显示搜索结果总数以及当前结果的序号。具体的效果图如下：



电池状态信息：

`acpi` 可展示电池电量剩余百分比。

使用不同颜色展示不同的电池状态：

电池状态	颜色
75% - 100%	绿色
30% - 75%	黄色
0 - 30%	红色

所有的颜色都取决于不同的主题。

优雅的界面

状态栏分割符：

可通过使用 `statusline_separator` 来定制状态栏分割符，例如使用非常常用的方向箭头作为状态栏分割符：

```
1. statusline_separator = 'arrow'
```

SpaceVim 所支持的分割符以及截图如下：

分割符	截图
arrow	
curve	
slant	
nil	
fire	

SpaceVim 功能模块：

功能模块可以通过 `SPC t m m` 快捷键显示或者隐藏。默认使用 Unicode 字符，可通过设置 `statusline_unicode_symbols = false` 来启用 ASCII 字符。（或许在终端中无法设置合适的字体时，可使用这一选项）。

状态栏中功能模块内的字符显示与否，同如下快捷键功能保持一致：

快捷键	Unicode	ASCII	功能
<code>SPC t 8</code>	Ⓔ	8	高亮指定列后所有字符
<code>SPC t f</code>	Ⓕ	f	高亮指定列字符
<code>SPC t s</code>	Ⓖ	s	语法检查
<code>SPC t S</code>	Ⓘ	S	拼写检查
<code>SPC t w</code>	Ⓜ	w	行尾空格检查

状态栏的颜色

当前版本的状态栏支持 `gruvbox` / `molokai` / `nord` / `one` / `onedark`，如果你需要使用其他主题，可以通过以下木板来设置：

```
1. " the theme colors should be
2. " [
3. "   \ [ a_gui_fg, a_gui_bg, a_cterm_fg, a_cterm_bg],
4. "   \ [ b_gui_fg, b_gui_bg, b_cterm_fg, b_cterm_bg],
5. "   \ [ c_gui_fg, c_gui_bg, c_cterm_fg, c_cterm_bg],
6. "   \ [ z_gui_bg, z_cterm_bg],
7. "   \ [ i_gui_fg, i_gui_bg, i_cterm_fg, i_cterm_bg],
8. "   \ [ v_gui_fg, v_gui_bg, v_cterm_fg, v_cterm_bg],
9. "   \ [ r_gui_fg, r_gui_bg, r_cterm_fg, r_cterm_bg],
```

```
10. " \ [ ii_guifg, ii_guibg, ii_ctermfg, ii_ctermbg],
11. " \ [ in_guifg, in_guibg, in_ctermfg, in_ctermbg],
12. " \ ]
13. " group_a: window id
14. " group_b/group_c: stausline sections
15. " group_z: empty area
16. " group_i: window id in insert mode
17. " group_v: window id in visual mode
18. " group_r: window id in select mode
19. " group_ii: window id in iedit-insert mode
20. " group_in: windows id in iedit-normal mode
21. function! SpaceVim#mapping#guide#theme#gruvbox#palette() abort
22.     return [
23.         \ ['#282828', '#a89984', 246, 235],
24.         \ ['#a89984', '#504945', 239, 246],
25.         \ ['#a89984', '#3c3836', 237, 246],
26.         \ ['#665c54', 241],
27.         \ ['#282828', '#83a598', 235, 109],
28.         \ ['#282828', '#fe8019', 235, 208],
29.         \ ['#282828', '#8ec07c', 235, 108],
30.         \ ['#282828', '#689d6a', 235, 72],
31.         \ ['#282828', '#8f3f71', 235, 132],
32.         \ ]
33. endfunction
```

这一模板是 gruvbox 主题的，如果你需要在切换主题是，状态栏都使用同一种颜色主题，可以设置

`custom_color_palette` :

```
1. custom_color_palette = [
2.     ["#282828", "#a89984", 246, 235],
3.     ["#a89984", "#504945", 239, 246],
4.     ["#a89984", "#3c3836", 237, 246],
5.     ["#665c54", 241],
6.     ["#282828", "#83a598", 235, 109],
7.     ["#282828", "#fe8019", 235, 208],
8.     ["#282828", "#8ec07c", 235, 108],
9.     ["#282828", "#689d6a", 235, 72],
10.    ["#282828", "#8f3f71", 235, 132],
11.    ]
```

标签栏

如果只有一个Tab，Buffers 将被罗列在标签栏上，每一个包含：序号、文件类型图标、文件名。如果有不止一个Tab，那么所有 Tab 将被罗列在标签栏上。标签栏上每一个 Tab 或者 Baffer 可通过快捷键 `<Leader> number` 进行快速访问，默认的 `<Leader>` 是 `\`。

快捷键	描述
<code><Leader> 1</code>	跳至标签栏序号 1
<code><Leader> 2</code>	跳至标签栏序号 2

<code><Leader> 3</code>	跳至标签栏序号 3
<code><Leader> 4</code>	跳至标签栏序号 4
<code><Leader> 5</code>	跳至标签栏序号 5
<code><Leader> 6</code>	跳至标签栏序号 6
<code><Leader> 7</code>	跳至标签栏序号 7
<code><Leader> 8</code>	跳至标签栏序号 8
<code><Leader> 9</code>	跳至标签栏序号 9

标签栏上也支持鼠标操作，左键可以快速切换至该序号，中键删除该标签。该特性只支持 neovim，并且需要 `has('tablineat')` 特性。

按键	描述
<code><Mouse-left></code>	掉至标签该序号标签
<code><Mouse-middle></code>	删除该序号标签

标签管理器

可使用 `SPC t t` 打开内置的标签管理器，标签管理器内的快捷键如下：

按键	描述
<code>o</code>	展开或关闭标签目录
<code>r</code>	重命名光标下的标签页
<code>n</code>	在光标位置下新建命名标签页
<code>N</code>	在光标位置下新建匿名标签页
<code>x</code>	删除光标下的标签页
<code><C-S-Up></code>	向上移动光标下的标签页
<code><C-S-Down></code>	向下移动光标下的标签页
<code><Enter></code>	跳至光标所对应的标签窗口

常规快捷键

- [窗口管理器](#)
- [File Operations](#)
- [Editor UI](#)
- [Native functions](#)
- [Bookmarks management](#)
- [Fuzzy finder](#)
- [交互](#)
- [常规操作](#)
- [以 g 为前缀的快捷键](#)
- [以 z 开头的命令](#)
- [搜索](#)
- [编辑](#)
- [错误处理](#)
- [工程管理](#)

窗口管理器

窗口管理器

窗口管理器快捷键只可以在 Normal 模式下使用，默认的前缀按键为 `s`，可以在配置文件中通过修改SpaceVim选项 `window_leader` 的值来设为其他按键：

按键	描述
<code>q</code>	Smart buffer close
<code>s + p</code>	Split nicely
<code>s + v</code>	<code>:split</code>
<code>s + g</code>	<code>:vsplit</code>
<code>s + t</code>	Open new tab (<code>:tabnew</code>)
<code>s + o</code>	Close other windows (<code>:only</code>)
<code>s + x</code>	Remove buffer, leave blank window
<code>s + q</code>	Remove current buffer, left buffer in the tabline will be displayed. If there is no buffer on the left, the right buffer will be displayed; if this is the last buffer in the tabline, then an empty buffer will be displayed.
<code>s + Q</code>	Close current buffer (<code>:close</code>)
<code>Tab</code>	Next window or tab
<code>Shift + Tab</code>	Previous window or tab
<code><leader> + sv</code>	Split with previous buffer
<code><leader> + sg</code>	Vertically split with previous buffer

SpaceVim has mapped normal `q` as smart buffer close, the normal func of `q` can be get by `<leader> q r`

Key	Mode	Action
<code><leader> + y</code>	visual	Copy selection to X11 clipboard (" <code>+y</code> ")
<code>Ctrl + c</code>	Normal	Copy full path of current buffer to X11 clipboard
<code><leader> + Ctrl + c</code>	Normal	Copy github.com url of current buffer to X11 clipboard(if it is a github repo)
<code><leader> + Ctrl + l</code>	Normal/visual	Copy github.com url of current lines to X11 clipboard(if it is a github repo)
<code><leader> + p</code>	Normal/visual	Paste selection from X11 clipboard (" <code>+p</code> ")
<code>Ctrl + f</code>	Normal	Smart page forward (<code>C-f/C-d</code>)
<code>Ctrl + b</code>	Normal	Smart page backwards (<code>C-b/C-u</code>)
<code>Ctrl + e</code>	Normal	Smart scroll down (<code>3C-e/j</code>)
<code>Ctrl + y</code>	Normal	Smart scroll up (<code>3C-y/k</code>)

Ctrl + q	Normal	Ctrl + w
Ctrl + x	Normal	Switch buffer and placement
Up, Down	Normal	Smart up and down
}	Normal	After paragraph motion go to first non-blank char (} [^])
<	Visual/Normal	Indent to left and re-select
>	Visual/Normal	Indent to right and re-select
Tab	Visual	Indent to right and re-select
Shift + Tab	Visual	Indent to left and re-select
gp	Normal	Select last paste
Q / gQ	Normal	Disable EX-mode ()
Ctrl + a	Command	Navigation in command line
Ctrl + b	Command	Move cursor backward in command line
Ctrl + f	Command	Move cursor forward in command line

File Operations

File Operations

Key	Mode	Action
<code><leader> + cd</code>	Normal	Switch to the directory of the open buffer
<code><leader> + w</code>	Normal/visual	Write (:w)
<code>Ctrl + s</code>	Normal/visual/Command	Write (:w)
<code>:w!!</code>	Command	Write as root (!sudo tee > /dev/null %)

Editor UI

Editor UI

Key	Mode	Action
F2	<i>All</i>	Toggle tagbar
F3	<i>All</i>	Toggle Vimfiler
<leader> + num	Normal	Jump to the buffer with the num index
<Alt> + num	Normal	Jump to the buffer with the num index, this only works in neovim
<Alt> + h / <Left>	Normal	Jump to left buffer in the tabline, this only works in neovim
<Alt> + l / <Right>	Normal	Jump to Right buffer in the tabline, this only works in neovim
<leader> + ts	Normal	Toggle spell-checker (:setlocal spell!)
<leader> + tn	Normal	Toggle line numbers (:setlocal nonumber!)
<leader> + tl	Normal	Toggle hidden characters (:setlocal nolist!)
<leader> + th	Normal	Toggle highlighted search (:set hlsearch!)
<leader> + tw	Normal	Toggle wrap (:setlocal wrap! breakindent!)
g0	Normal	Go to first tab (:tabfirst)
g\$	Normal	Go to last tab (:tablast)
gr	Normal	Go to previous tab (:tabprevious)
Ctrl + <Down>	Normal	Move to split below (j)
Ctrl + <Up>	Normal	Move to upper split (k)
Ctrl + <Left>	Normal	Move to left split (h)
Ctrl + <Right>	Normal	Move to right split (l)
*	Visual	Search selection forwards
#	Visual	Search selection backwards
, + Space	Normal	Remove all spaces at EOL
Ctrl + r	Visual	Replace selection
<leader> + lj	Normal	Next on location list
<leader> + lk	Normal	Previous on location list
<leader> + S	Normal/visual	Source selection

Native functions

Native functions

Key	Mode	Action
<leader> + qr	Normal	Same as native q
<leader> + qr/	Normal	Same as native q/, open cmdwin
<leader> + qr?	Normal	Same as native q?, open cmdwin
<leader> + qr:	Normal	Same as native q:, open cmdwin

Bookmarks management

Bookmarks management

Key	Mode	Action
<code>m</code> + <code>a</code>	Normal	Show list of all bookmarks
<code>m</code> + <code>m</code>	Normal	Toggle bookmark in current line
<code>m</code> + <code>n</code>	Normal	Jump to next bookmark
<code>m</code> + <code>p</code>	Normal	Jump to previous bookmark
<code>m</code> + <code>i</code>	Normal	Annotate bookmark

As SpaceVim use above bookmarks mappings, so you can not use `a` , `m` , `n` , `p` or `i` registers to mark current position, but other registers should works will. if you really need to use these registers, you can add `nnoremap <leader>m m` to your custom configuration, then you use use `a` registers via `\ma`

Fuzzy finder

Fuzzy finder

SpaceVim provides five kinds of fuzzy finder, each of them is configured in a layer(`unite` , `denite` , `leaderf` , `ctrlp` and `fzf` layer).These layers have the same key bindings and features. But they need different dependencies.

User only need to load one of these layers, then will be able to get these features.

Key bindings

Key bindings	Description
<code><Leader> f <space></code>	Fuzzy find menu:CustomKeyMaps
<code><Leader> f e</code>	Fuzzy find register
<code><Leader> f f</code>	Fuzzy find file
<code><Leader> f h</code>	Fuzzy find history/yank
<code><Leader> f j</code>	Fuzzy find jump, change
<code><Leader> f l</code>	Fuzzy find location list
<code><Leader> f m</code>	Fuzzy find output messages
<code><Leader> f o</code>	Fuzzy find outline
<code><Leader> f q</code>	Fuzzy find quick fix
<code><Leader> f r</code>	Resumes Unite window

But in current version of SpaceVim, leaderf/ctrlp and fzf layer has not be finished.

Feature	unite	denite	leaderf	ctrlp	fzf
menu: CustomKeyMaps	yes	yes	no	no	no
register	yes	yes	no	yes	yes
file	yes	yes	yes	yes	yes
yank history	yes	yes	no	no	yes
jump	yes	yes	no	yes	yes
location list	yes	yes	no	no	yes
outline	yes	yes	yes	yes	yes
message	yes	yes	no	no	yes
quickfix list	yes	yes	no	yes	yes
resume windows	yes	yes	no	no	no

Key bindings within fuzzy finder buffer

--	--	--

key bindings	Mode	description
<code>Tab</code> / <code><C-j></code>	-	Select next line
<code>Shift + Tab</code> / <code><C-k></code>	-	Select previous line
<code>jk</code>	Insert	Leave Insert mode (Only for denite/unite)
<code>Ctrl</code> + <code>w</code>	Insert	Delete backward path
<code>Enter</code>	-	Run default action
<code>Ctrl</code> + <code>s</code>	-	Open in a split
<code>Ctrl</code> + <code>v</code>	-	Open in a vertical split
<code>Ctrl</code> + <code>t</code>	-	Open in a new tab
<code>Ctrl</code> + <code>g</code>	-	Exit unite

Denite/Unite normal mode key bindings

key bindings	Mode	description
<code>Ctrl</code> + <code>h/k/l/r</code>	Normal	Un-map
<code>Ctrl</code> + <code>l</code>	Normal	Redraw
<code>Tab</code>	Normal	Select actions
<code>Space</code>	Normal	Toggle mark current candidate, up
<code>r</code>	Normal	Replace ('search' profile) or rename
<code>Ctrl</code> + <code>z</code>	Normal/insert	Toggle transpose window

The above key bindings only are part of fuzzy finder layers, please read the layer's documentation.

交互

交互

快捷键

快捷键导航

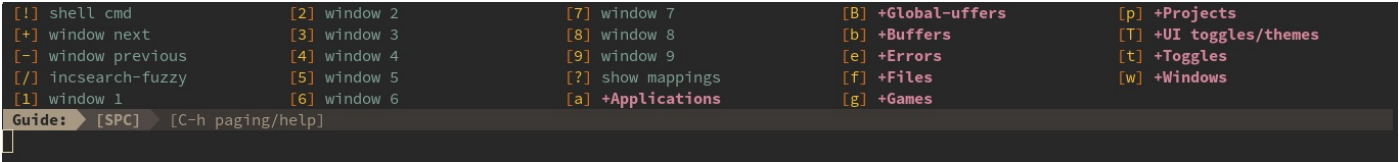
当 Normal 模式下按下前缀键后出现输入延迟，则会在屏幕下方打开一个快捷键导航窗口，提示当前可用的快捷键及其功能描述，目前支持的前缀键有：`[SPC]`、`[Window]`、`<Leader>`、`g`、`z`。

这些前缀的按键为：

前缀名称	用户选项以及默认值	描述
<code>[SPC]</code>	空格键	SpaceVim 默认前缀键
<code>[Window]</code>	<code>windows_leader</code> / <code>s</code>	SpaceVim 默认窗口前缀键
<code><leader></code>	默认的 Vim leader 键	Vim/neovim 默认前缀键

默认情况下，快捷键导航将在输入延迟超过 1000ms 后打开，你可以通过修改 vim 的 `'timeoutlen'` 选项来修改成适合自己的延迟时间长度。

例如，Normal 模式下按下空格键，你将会看到：



这一导航窗口将提示所有以空格键为前缀的快捷键，并且根据功能将这些快捷键进行了分组，例如 buffer 相关的快捷键都是 `b`，工程相关的快捷键都是 `p`。在代码导航窗口内，按下 `<C-h>` 键，可以获取一些帮助信息，这些信息将被显示在状态栏上，提示的是一些翻页和撤销按键的快捷键。

按键	描述
<code>u</code>	撤销按键
<code>n</code>	向下翻页
<code>p</code>	向上翻页

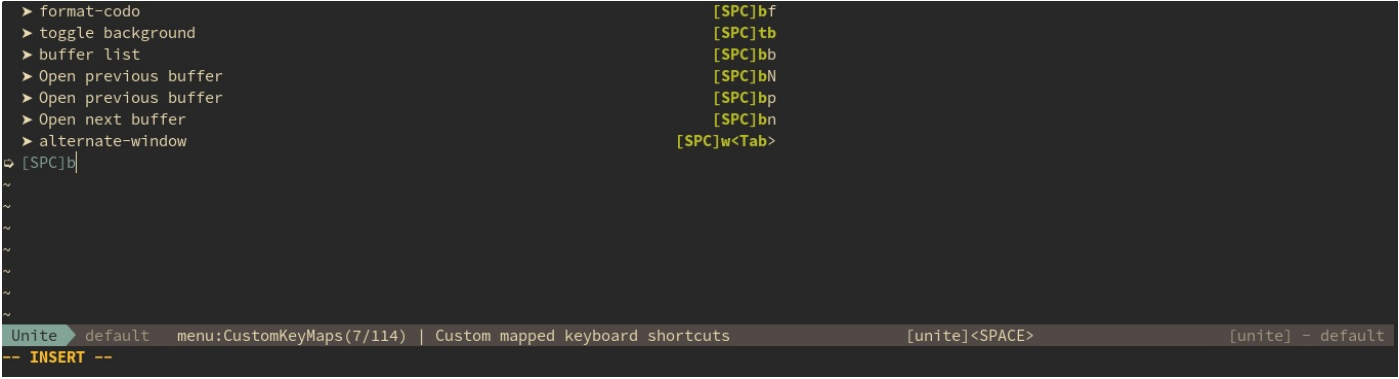
如果要自定义以 `[SPC]` 为前缀的快捷键，可以使用 `SpaceVim#custom#SPC()`，示例如下：

```
1. call SpaceVim#custom#SPC('nnoremap', ['f', 't'], 'echom "hello world"', 'test custom SPC', 1)
```

通过 Unite/Denite 浏览快捷键

可以通过 `SPC ?` 使用 Unite 将当前快捷键罗列出来。然后可以输入快捷键按键字母或者描述，Unite 可以通

过模糊匹配，并展示结果。



使用 `<Tab>` 键或者上下方向键选择你需要的快捷键，回车将执行这一快捷键。

获取帮助信息

Denite/Unite 是一个强大的信息筛选浏览器，这类似于 emacs 中的 Helm。以下这些快捷键将帮助你快速获取需要的帮助信息：

快捷键	描述
<code>SPC h SPC</code>	使用 Unite 展示 SpaceVim 帮助文档章节目录
<code>SPC h i</code>	获取光标下单词的帮助信息
<code>SPC h k</code>	使用快捷键导航，展示 SpaceVim 所支持的前缀键
<code>SPC h m</code>	使用 Unite 浏览所有 man 文档

报告一个问题：

快捷键	描述
<code>SPC h I</code>	根据模板展示 Issue 所必须的信息

可用模块

所有可用模块可以通过命令 `:SPLayer -l` 或者快捷键 `SPC h l` 来展示。

可用的插件



可通过快捷键 `<leader> l p` 列出所有已安装的插件，支持模糊搜索，回车将使用浏览器打开该插件的官网。

添加用户自定义插件

如果添加来自于 github.com 的插件，可以 `用户名/仓库名` 这一格式，将该插件添加到 `custom_plugins`，示例如下：

```
1. [[custom_plugins]]
2. name = 'lilydjwg/colorizer'
3. merged = 0
```

界面元素显示切换

所有的界面元素切换快捷键都是已  或者  开头的，你可以在快捷键导航中查阅所有快捷键。

常规操作

常规操作

光标移动

光标的移动默认采用 vi 的默认形式：`hjkl`。

快捷键	描述
<code>h</code>	向左移动光标（Vim 原生功能，无映射）
<code>j</code>	向下移动光标（Vim 原生功能，无映射）
<code>k</code>	向上移动光标（Vim 原生功能，无映射）
<code>l</code>	向右移动光标（Vim 原生功能，无映射）
<code>H</code>	光标移至屏幕最上方（Vim 原生功能，无映射）
<code>L</code>	光标移至屏幕最下方（Vim 原生功能，无映射）
<code>SPC j 0</code>	跳至行首（并且标记原始位置）
<code>SPC j \$</code>	跳至行尾（并且标记原始位置）
<code>SPC t -</code>	锁定光标在屏幕中间（TODO）

使用 vim-easymotion 快速跳转

快速跳到网址（TODO）

类似于 Firefox 的 vimperator 的 `f` 键的功能。

快捷键	描述
<code>SPC j u</code> / (<code>o</code> for help buffer)	快速跳到/打开url

常用的成对快捷键

快捷键	描述
<code>[SPC</code>	在当前行或已选区域上方添加空行
<code>] SPC</code>	在当前行或已选区域下方添加空行
<code>[b</code>	跳至前一 buffer
<code>] b</code>	跳至下一 buffer
<code>[f</code>	跳至文件夹中的前一个文件
<code>] f</code>	跳至文件夹中的下一个文件
<code>[l</code>	跳至前一个错误处
<code>] l</code>	跳至下一个错误处
<code>[c</code>	跳至前一个 vcs hunk（需要 VersionControl 模块）

<div>]</div> c	跳至下一个 vcs hunk (需要 VersionControl 模块)
<div>[</div> q	跳至前一个错误
<div>]</div> q	跳至下一个错误
<div>[</div> t	跳至前一个标签页
<div>]</div> t	跳至下一个标签页
<div>[</div> w	跳至前一个窗口
<div>]</div> w	跳至下一个窗口
<div>[</div> e	向上移动当前行或者已选择行
<div>]</div> e	向下移动当前行或者已选择行
<div>[</div> p	粘贴至当前行上方
<div>]</div> p	粘贴至当前行下方
g p	选择粘贴的区域

跳转，合并，拆分

以

SPC j

 为前缀的快捷键主要用作：跳转 (jumping)，合并 (joining)，拆分 (splitting)。

跳转

快捷键	描述
<div>SPC j o</div>	跳至行首，并且在原始位置留下标签，以便跳回
<div>SPC j \$</div>	跳至行尾，并且在原始位置留下标签，以便跳回
<div>SPC j b</div>	向后回跳
<div>SPC j f</div>	向前跳
<div>SPC j d</div>	跳至当前目录某个文件夹
<div>SPC j D</div>	跳至当前目录某个文件夹 (在另外窗口展示文件列表)
<div>SPC j i</div>	跳至当前文件的某个函数，使用 Denite 打开语法树
<div>SPC j I</div>	跳至所有 Buffer 的语法树 (TODO)
<div>SPC j j</div>	跳至当前窗口的某个字符 (easymotion)
<div>SPC j J</div>	跳至当前窗口的某两个字符的组合 (easymotion)
<div>SPC j k</div>	跳至下一行，并且对齐下一行
<div>SPC j l</div>	跳至某一行 (easymotion)
<div>SPC j q</div>	show the dumb-jump quick look tooltip (TODO)
<div>SPC j u</div>	跳至窗口某个 url (TODO)
<div>SPC j v</div>	跳至某个 vim 函数的定义处 (TODO)
<div>SPC j w</div>	跳至 Buffer 中某个单词 (easymotion)

合并，拆分

快捷键	描述

J	合并当前行和下一行
SPC j k	跳至下一行，并且对齐该行
SPC j n	从光标处断开当前行，并且插入空行以及进行对齐
SPC j o	从光标处拆分该行，光标留在当前行
SPC j s	从光标处进行拆分 String
SPC j S	从光标处进行拆分 String，并插入对齐的空行

窗口操作

窗口操作常用快捷键

每一个窗口，都有一个编号，该编号显示在状态栏的最前端，可通过 SPC 编号 进行快速窗口跳转。

快捷键	描述
SPC 1	跳至窗口 1
SPC 2	跳至窗口 2
SPC 3	跳至窗口 3
SPC 4	跳至窗口 4
SPC 5	跳至窗口 5
SPC 6	跳至窗口 6
SPC 7	跳至窗口 7
SPC 8	跳至窗口 8
SPC 9	跳至窗口 9

窗口操作相关快捷键（以 SPC w 为前缀）：

快捷键	描述
SPC w TAB / <Tab>	在统一标签内进行窗口切换
SPC w =	对齐分离的窗口
SPC w b	force the focus back to the minibuffer (TODO)
SPC w c	进入阅读模式，浏览当前窗口
SPC w C	选择某一个窗口，并且进入阅读模式
SPC w d	删除一个窗口
SPC u SPC w d	delete a window and its current buffer (does not delete the file) (TODO)
SPC w D	选择一个窗口，并且关闭
SPC u SPC w D	delete another window and its current buffer using vim-choosewin (TODO)
SPC w t	toggle window dedication (dedicated window cannot be reused by a mode) (TODO)
SPC w f	toggle follow mode (TODO)

<code>SPC w F</code>	新建一个新的标签页
<code>SPC w h</code>	移至左边窗口
<code>SPC w H</code>	将窗口向左移动
<code>SPC w j</code>	移至下方窗口
<code>SPC w J</code>	将窗口移至下方
<code>SPC w k</code>	移至上方窗口
<code>SPC w K</code>	将窗口移至上方
<code>SPC w l</code>	移至右方窗口
<code>SPC w L</code>	将窗口移至右方
<code>SPC w m</code>	最大化/最小化窗口（最大化相当于关闭其他窗口）(TODO, now only support maximize)
<code>SPC w M</code>	选择窗口进行替换
<code>SPC w o</code>	按序切换标签页
<code>SPC w p m</code>	open messages buffer in a popup window (TODO)
<code>SPC w p p</code>	close the current sticky popup window (TODO)
<code>SPC w r</code>	按序切换窗口
<code>SPC w R</code>	逆序切换窗口
<code>SPC w s or SPC w -</code>	水平分割窗口
<code>SPC w S</code>	水平分割窗口，并切换至新窗口
<code>SPC w u</code>	undo window layout (used to effectively undo a closed window) (TODO)
<code>SPC w U</code>	redo window layout (TODO)
<code>SPC w v or SPC w /</code>	垂直分离窗口
<code>SPC w V</code>	垂直分离窗口，并切换至新窗口
<code>SPC w w</code>	切换至前一窗口
<code>SPC w W</code>	选择一个窗口

文件和 Buffer 操作

Buffer 操作相关快捷键

Buffer 操作相关快捷键都是已 `SPC b` 为前缀的：

快捷键	描述
<code>SPC TAB</code>	切换至前一buffer，可用于两个 buffer 来回切换
<code>SPC b .</code>	启用 buffer 临时快捷键
<code>SPC b b</code>	切换至某一 buffer，通过 Unite/Denite 进行筛选
<code>SPC b d</code>	删除当前 buffer，但保留 Vim 窗口
<code>SPC u SPC b d</code>	kill the current buffer and window (does not delete the visited file) (TODO)

<div>SPC b D</div>	选择一个窗口，并删除其 buffer
<div><div>SPC u SPC b</div><div>D</div></div>	kill a visible buffer and its window using ace-window(TODO)
<div>SPC b C-d</div>	删除其他 buffer
<div>SPC b C-D</div>	kill buffers using a regular expression(TODO)
<div>SPC b e</div>	清除当前 buffer 内容，需要手动确认
<div>SPC b h</div>	打开 SpaceVim 欢迎界面
<div>SPC b n</div>	切换至下一个 buffer，排除特殊插件的 buffer
<div>SPC b m</div>	打开 Messages buffer
<div><div>SPC u SPC b</div><div>m</div></div>	kill all buffers and windows except the current one(TODO)
<div>SPC b p</div>	切换至前一个 buffer，排除特殊插件的 buffer
<div>SPC b P</div>	使用剪切板内容替换当前 buffer
<div>SPC b R</div>	从磁盘重新读取当前 buffer 所对应的文件
<div>SPC b s</div>	switch to the <i>scratch</i> buffer (create it if needed) (TODO)
<div>SPC b w</div>	切换只读权限
<div>SPC b Y</div>	将整个 buffer 复制到剪切板
<div>z f</div>	Make current function or comments visible in buffer as much as possible (TODO)

新建空白 buffer

快捷键	描述
<div>SPC b N h</div>	在左侧新建一个窗口，并在其中新建空白 buffer
<div>SPC b N j</div>	在下方新建一个窗口，并在其中新建空白 buffer
<div>SPC b N k</div>	在上方新建一个窗口，并在其中新建空白 buffer
<div>SPC b N l</div>	在右侧新建一个窗口，并在其中新建空白 buffer
<div>SPC b N n</div>	在当前窗口新建一个空白 buffer

特殊 buffer

在 SpaceVim 中，有很多特殊的 buffer，这些 buffer 是由插件或者 SpaceVim 自身新建的，并不会被列出。

文件操作相关快捷键

文件操作相关的快捷键都是以

SPC f

 为前缀的：

快捷键	描述
<div>SPC f b</div>	跳至文件书签
<div>SPC f c</div>	copy current file to a different location(TODO)
<div>SPC f C d</div>	修改文件编码 unix -> dos
<div>SPC f C u</div>	修改文件编码 dos -> unix

SPC f D	删除文件以及 buffer，需要手动确认
SPC f E	open a file with elevated privileges (sudo edit)(TODO)
SPC f f	打开文件
SPC f F	打开光标下的文件
SPC f o	open a file using the default external program(TODO)
SPC f R	rename the current file(TODO)
SPC f s	保存文件
SPC f S	保存所有文件
SPC f r	打开文件历史
SPC f t	切换侧栏文件树
SPC f T	打开文件树侧栏
SPC f y	复制当前文件，并且显示当前文件路径

Vim 和 SpaceVim 相关文件

SpaceVim 相关的快捷键均以 `SPC f v` 为前缀，这便于快速访问 SpaceVim 的配置文件：

快捷键	描述
SPC f v v	复制并显示当前 SpaceVim 的版本
SPC f v d	打开 SpaceVim 的用户配置文件

文件树

SpaceVim 使用 vimfiler 作为默认的文件树插件，默认的快捷键是 `F3`，SpaceVim 也提供了另外一组快捷键 `SPC f t` 和 `SPC f T` 来打开文件树，如果需要使用 nerdtree 作为默认文件树，需要设置：

```
1. # 默认值为 vimfiler
2. filemanager = "nerdtree"
```

SpaceVim 的文件树提供了版本控制信息的借口，但是这一特性需要分析文件夹内容，会使得文件树插件比较慢，因此默认没有打开，如果需要使用这一特性，可向配置文件中加入 `enable_vimfiler_gitstatus = true`，启用后的截图如下：



文件树中的常用操作

文件树中主要以 `h j k l` 为核心，这类似于 `vim` 中常用的快捷键：

快捷键	描述
<code><F3></code> or <code>SPC f t</code>	切换文件树
文件树内的快捷键	
<code><Left></code> or <code>h</code>	移至父目录，并关闭文件夹
<code><Down></code> or <code>j</code>	向下移动光标
<code><Up></code> or <code>k</code>	向上移动光标
<code><Right></code> or <code>l</code>	展开目录，或打开文件
<code>Ctrl + j</code>	未使用
<code>Ctrl + l</code>	未使用
<code>E</code>	未使用
<code>.</code>	切换显示隐藏文件
<code>sv</code>	分屏编辑该文件
<code>sg</code>	垂直分屏编辑该文件
<code>p</code>	预览文件
<code>i</code>	切换至文件夹历史
<code>v</code>	快速查看
<code>gx</code>	使用相关程序执行该文件 (TODO)
<code>'</code>	切换标签
<code>V</code>	标记该文件

Ctrl + r

刷新页面

文件树中打开文件

如果只有一个可编辑窗口，则在该窗口中打开选择的文件，否则则需要制定窗口来打开文件：

快捷键	描述
<div>l</div> or <div>Enter</div>	打开文件
<div>sg</div>	分屏打开文件
<div>sv</div>	垂直分屏打开文件

以 g 为前缀的快捷键

以 g 为前缀的快捷键

在 Normal 模式下按下 `g` 之后，如果你不记得快捷键出现按键延迟，那么快捷键导航将会提示你所有以 `g` 为前缀的快捷键。

快捷键	描述
<code>g#</code>	反向搜索光标下的词
<code>g\$</code>	跳向本行最右侧字符
<code>g&</code>	针对所有行重复执行上一次 “:s” 命令
<code>g'</code>	跳至标签
<code>g*</code>	正向搜索光标下的词
<code>g+</code>	newer text state
<code>g,</code>	newer position in change list
<code>g-</code>	older text state
<code>g/</code>	stay incsearch
<code>g0</code>	go to leftmost character
<code>g;</code>	older position in change list
<code>g<</code>	last page of previous command output
<code>g<Home></code>	go to leftmost character
<code>gE</code>	end of previous word
<code>gF</code>	edit file under cursor(jump to line after name)
<code>gH</code>	select line mode
<code>gI</code>	insert text in column 1
<code>gJ</code>	join lines without space
<code>gN</code>	visually select previous match
<code>gQ</code>	switch to Ex mode
<code>gR</code>	enter VREPLACE mode
<code>gT</code>	previous tag page
<code>gU</code>	make motion text uppercase
<code>g]</code>	tselect cursor tag
<code>g^</code>	go to leftmost no-white character
<code>g_</code>	go to last char
<code>g`</code>	跳至标签，等同于 <code>g'</code>
<code>ga</code>	打印光标字符的 ascii 值
<code>gd</code>	跳至定义处

<code>ge</code>	go to end of previous word
<code>gf</code>	edit file under cursor
<code>gg</code>	go to line N
<code>gh</code>	select mode
<code>gi</code>	insert text after '^ mark
<code>gj</code>	move cursor down screen line
<code>gk</code>	move cursor up screen line
<code>gm</code>	go to middle of screenline
<code>gn</code>	visually select next match
<code>go</code>	goto byte N in the buffer
<code>gs</code>	sleep N seconds
<code>gt</code>	next tag page
<code>gu</code>	make motion text lowercase
<code>g~</code>	swap case for Nmove text
<code>g<End></code>	跳至本行最右侧字符，等同于 <code>g\$</code>
<code>g<C-G></code>	显示光标信息

以 z 开头的命令

以 z 开头的命令

当你不记得按键映射时，你可以在普通模式下输入前缀 `z`，然后你会看到所有以 `z` 为前缀的函数映射。

Key Binding	Description
<code>z<Right></code>	scroll screen N characters to left
<code>z+</code>	cursor to screen top line N
<code>z-</code>	cursor to screen bottom line N
<code>z.</code>	cursor line to center
<code>z<CR></code>	cursor line to top
<code>z=</code>	spelling suggestions
<code>zA</code>	toggle folds recursively
<code>zC</code>	close folds recursively
<code>zD</code>	delete folds recursively
<code>zE</code>	eliminate all folds
<code>zF</code>	create a fold for N lines
<code>zG</code>	mark good spelled(update internal-wordlist)
<code>zH</code>	scroll half a screenwidth to the right
<code>zL</code>	scroll half a screenwidth to the left
<code>zM</code>	set <code>foldlevel</code> to zero
<code>zN</code>	set <code>foldenable</code>
<code>zO</code>	open folds recursively
<code>zR</code>	set <code>foldlevel</code> to deepest fold
<code>zW</code>	mark wrong spelled
<code>zX</code>	re-apply <code>foldlevel</code>
<code>z^</code>	cursor to screen bottom line N
<code>za</code>	toggle a fold
<code>zb</code>	redraw, cursor line at bottom
<code>zc</code>	close a fold
<code>zd</code>	delete a fold
<code>ze</code>	right scroll horizontally to cursor position
<code>zf</code>	create a fold for motion
<code>zg</code>	mark good spelled
<code>zh</code>	scroll screen N characters to right
<code>zi</code>	toggle foldenable

<code>zj</code>	mode to start of next fold
<code>zk</code>	mode to end of previous fold
<code>zl</code>	scroll screen N characters to left
<code>zm</code>	subtract one from <code>foldlevel</code>
<code>zn</code>	reset <code>foldenable</code>
<code>zo</code>	open fold
<code>zr</code>	add one to <code>foldlevel</code>
<code>zs</code>	left scroll horizontally to cursor position
<code>zt</code>	cursor line at top of window
<code>zv</code>	open enough folds to view cursor line
<code>zx</code>	re-apply foldlevel and do “zV”
<code>zz</code>	smart scroll
<code>z<Left></code>	scroll screen N characters to right

搜索

搜索

使用额外工具

SpaceVim 像下面那样调用不同搜索工具的搜索接口：

- `rg` - ripgrep
- `ag` - the silver searcher
- `pt` - the platinum searcher
- `ack`
- `grep`

SpaceVim 中的搜索命令是以 `SPC s` 为前缀的，前一个键是使用的工具，后一个键是范围。例如 `SPC s a b` 将使用 `ag` 在当前所有已经打开的缓冲区中进行搜索。

如果最后一个键(决定范围)是大写字母，那么就会对当前光标下的单词进行搜索。举个例子 `SPC s a B` 将会搜索当前光标下的单词。

如果工具键被省略了，那么会用默认的搜索工具进行搜索。默认的搜索工具对应在 `g:spacevim_search_tools` 列表中的第一个工具。列表中的工具默认的顺序为：`rg`，`ag`，`pt`，`ack` then `grep`。举个例子 如果 `rg` 和 `ag` 没有在系统中找到，那么 `SPC s b` 会使用 `pt` 进行搜索。

下表是全部的工具键：

Tool	Key
ag	a
grep	g
ack	k
rg	r
pt	t

应当避免的范围和对应按键为：

范围	键
opened buffers	b
files in a given directory	f
current project	p

可以双击按键序列中的第二个键来在当前文件中进行搜索。举个例子：`SPC s a a` 会使用 `ag` 在当前文件中进行搜索。

Notes：

- 如果使用源代码管理的话 `rg`, `ag` 和 `pt` 都会被忽略掉, 但是他们可以在任意目录中正常运行.
- 也可以通过将它们标记在联合缓冲区来一次搜索多个目录. 注意 如果你使用 `pt`, [TCL parser tools](#) 同时也需要安装一个名叫 `pt` 的命令行工具.

常用按键绑定

Key Binding	Description
<code>SPC r l</code>	resume the last completion buffer
<code>SPC s `</code>	go back to the previous place before jump
Prefix argument	will ask for file extensions

在当前文件中进行搜索

Key Binding	Description
<code>SPC s s</code>	search with the first found tool
<code>SPC s S</code>	search with the first found tool with default input
<code>SPC s a a</code>	<code>ag</code>
<code>SPC s a A</code>	<code>ag</code> with default input
<code>SPC s g g</code>	<code>grep</code>
<code>SPC s g G</code>	<code>grep</code> with default input
<code>SPC s r r</code>	<code>rg</code>
<code>SPC s r R</code>	<code>rg</code> with default input

搜索当前文件所在的文件夹

Key Binding	Description
<code>SPC s d</code>	searching in buffer directory with default tool
<code>SPC s D</code>	searching in buffer directory cursor word with default tool
<code>SPC s a d</code>	searching in buffer directory with <code>ag</code>
<code>SPC s a D</code>	searching in buffer directory cursor word with <code>ag</code>
<code>SPC s g d</code>	searching in buffer directory with <code>grep</code>
<code>SPC s g D</code>	searching in buffer directory cursor word with <code>grep</code>
<code>SPC s k d</code>	searching in buffer directory with <code>ack</code>
<code>SPC s k D</code>	searching in buffer directory cursor word with <code>ack</code>
<code>SPC s r d</code>	searching in buffer directory with <code>rg</code>
<code>SPC s r D</code>	searching in buffer directory cursor word with <code>rg</code>
<code>SPC s t d</code>	searching in buffer directory with <code>pt</code>
<code>SPC s t D</code>	searching in buffer directory cursor word with <code>pt</code>

在所有打开的缓冲区中进行搜索

Key Binding	Description

SPC s b	search with the first found tool
SPC s B	search with the first found tool with default input
SPC s a b	ag
SPC s a B	ag with default input
SPC s g b	grep
SPC s g B	grep with default input
SPC s k b	ack
SPC s k B	ack with default input
SPC s r b	rg
SPC s r B	rg with default input
SPC s t b	pt
SPC s t B	pt with default input

在任意目录中进行搜索

Key Binding	Description
SPC s f	search with the first found tool
SPC s F	search with the first found tool with default input
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 r f	rg
SPC s r F	rg with default text
SPC s t f	pt
SPC s t F	pt with default text

在工程中进行搜索

Key Binding	Description
SPC / or SPC s p	search with the first found tool
SPC * or SPC s P	search with the first found tool with default input
SPC s a p	ag
SPC s a P	ag with default text
SPC s g p	grep
SPC s g P	grep with default text
SPC s k p	ack

<code>SPC s k P</code>	ack with default text
<code>SPC s t p</code>	pt
<code>SPC s t P</code>	pt with default text
<code>SPC s r p</code>	rg
<code>SPC s r P</code>	rg with default text

提示：在工程中进行搜索的话，无需提前打开文件。在工程保存目录中使用 `SPC p p` 和 `C-s`， 就比如 `SPC s p` .(TODO)

后台进行工程搜索

在工程中进行后台搜索时，当搜索完成时，会在状态栏上进行显示。

Key Binding	Description
<code>SPC s j</code>	searching input expr background with the first found tool
<code>SPC s J</code>	searching cursor word background with the first found tool
<code>SPC s l</code>	List all searching result in quickfix buffer
<code>SPC s a j</code>	ag
<code>SPC s a J</code>	ag with default text
<code>SPC s g j</code>	grep
<code>SPC s g J</code>	grep with default text
<code>SPC s k j</code>	ack
<code>SPC s k J</code>	ack with default text
<code>SPC s t j</code>	pt
<code>SPC s t J</code>	pt with default text
<code>SPC s r j</code>	rg
<code>SPC s r J</code>	rg with default text

在网上进行搜索

Key Binding	Description
<code>SPC s w g</code>	Get Google suggestions in vim. Opens Google results in Browser.
<code>SPC s w w</code>	Get Wikipedia suggestions in vim. Opens Wikipedia page in Browser. (TODO)

注意：为了在 vim 中使用谷歌 suggestions， 你需要在你的默认配置文件中加入 `let g:spacevim_enable_googlesuggest = 1` .

实时代码检索

Key Binding	Description
<code>SPC s g G</code>	Searching in project on the fly with default tools

FlyGrep 缓冲区的按键绑定：

Key Binding	Description
<code><Esc></code>	close FlyGrep buffer
<code><Enter></code>	open file at the cursor line
<code><Tab></code>	move cursor line down
<code><S-Tab></code>	move cursor line up
<code><Bs></code>	remove last character
<code><C-w></code>	remove the Word before the cursor
<code><C-u></code>	remove the Line before the cursor
<code><C-k></code>	remove the Line after the cursor
<code><C-a></code> / <code><Home></code>	Go to the beginning of the line
<code><C-e></code> / <code><End></code>	Go to the end of the line

保持高亮

SPaceVim 使用 `g:spacevim_search_highlight_persist` 保持当前搜索结果的高亮状态到下一次搜索。同样可以通过 `SPC s c` 或者 运行 ex 命令 `:noh` 来取消搜索结果的高亮表示。

Highlight current symbol

SpaceVim supports highlighting of the current symbol on demand and add a transient state to easily navigate and rename these symbol.

It is also possible to change the range of the navigation on the fly to:











- buffer
- function
- visible area

To Highlight the current symbol under point press `SPC s h`.

Navigation between the highlighted symbols can be done with the commands:

Key Binding	Description
<code>*</code>	initiate navigation transient state on current symbol and jump forwards
<code>#</code>	initiate navigation transient state on current symbol and jump backwards
<code>SPC s e</code>	edit all occurrences of the current symbol
<code>SPC s h</code>	highlight the current symbol and all its occurrence within the current range
<code>SPC s H</code>	go to the last searched occurrence of the last highlighted symbol

In highlight symbol transient state:

Key Binding	Description
 e	edit occurrences ( *)
 n	go to next occurrence
 N /  p	go to previous occurrence
 b	search occurrence in all buffers
 /	search occurrence in whole project
 Tab	toggle highlight current 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

编辑

编辑

粘贴文本

粘贴文本自动缩进

文本操作命令

文本相关的命令（以 `x` 开头）:

Key Binding	Description	
<code>SPC x a &</code>	align region at &	
<code>SPC x a (</code>	align region at (
<code>SPC x a)</code>	align region at)	
<code>SPC x a [</code>	align region at [
<code>SPC x a]</code>	align region at]	
<code>SPC x a {</code>	align region at {	
<code>SPC x a }</code>	align region at }	
<code>SPC x a ,</code>	align region at ,	
<code>SPC x a .</code>	align region at . (for numeric tables)	
<code>SPC x a :</code>	align region at :	
<code>SPC x a ;</code>	align region at ;	
<code>SPC x a =</code>	align region at =	
<code>SPC x a </code>	align region at	
<code>SPC x a</code>		align region at
<code>SPC x a a</code>	align region (or guessed section) using default rules (TODO)	
<code>SPC x a c</code>	align current indentation region using default rules (TODO)	
<code>SPC x a l</code>	left-align with evil-lion (TODO)	
<code>SPC x a L</code>	right-align with evil-lion (TODO)	
<code>SPC x a r</code>	align region using user-specified regexp (TODO)	
<code>SPC x a m</code>	align region at arithmetic operators <code>(+ - * /)</code> (TODO)	
<code>SPC x c</code>	count the number of chars/words/lines in the selection region	
<code>SPC x d w</code>	delete trailing whitespaces	

<code>SPC x d</code> <code>SPC</code>	Delete all spaces and tabs around point, leaving one space
<code>SPC x g l</code>	set lanuages used by translate commands (TODO)
<code>SPC x g t</code>	translate current word using Google Translate
<code>SPC x g T</code>	reverse source and target languages (TODO)
<code>SPC x i c</code>	change symbol style to <code>lowerCamelCase</code>
<code>SPC x i C</code>	change symbol style to <code>UpperCamelCase</code>
<code>SPC x i i</code>	cycle symbol naming styles (i to keep cycling)
<code>SPC x i -</code>	change symbol style to <code>kebab-case</code>
<code>SPC x i k</code>	change symbol style to <code>kebab-case</code>
<code>SPC x i _</code>	change symbol style to <code>under_score</code>
<code>SPC x i u</code>	change symbol style to <code>under_score</code>
<code>SPC x i U</code>	change symbol style to <code>UP_CASE</code>
<code>SPC x j c</code>	set the justification to center (TODO)
<code>SPC x j f</code>	set the justification to full (TODO)
<code>SPC x j l</code>	set the justification to left (TODO)
<code>SPC x j n</code>	set the justification to none (TODO)
<code>SPC x j r</code>	set the justification to right (TODO)
<code>SPC x J</code>	move down a line of text (enter transient state)
<code>SPC x K</code>	move up a line of text (enter transient state)
<code>SPC x l d</code>	duplicate line or region (TODO)
<code>SPC x l s</code>	sort lines (TODO)
<code>SPC x l u</code>	uniquify lines (TODO)
<code>SPC x o</code>	use avy to select a link in the frame and open it (TODO)
<code>SPC x O</code>	use avy to select multiple links in the frame and open them (TODO)
<code>SPC x t c</code>	swap (transpose) the current character with the previous one
<code>SPC x t w</code>	swap (transpose) the current word with the previous one
<code>SPC x t l</code>	swap (transpose) the current line with the previous one
<code>SPC x u</code>	set the selected text to lower case (TODO)
<code>SPC x U</code>	set the selected text to upper case (TODO)
<code>SPC x w c</code>	count the number of occurrences per word in the select region (TODO)
<code>SPC x w d</code>	show dictionary entry of word from wordnik.com (TODO)
<code>SPC x TAB</code>	indent or dedent a region rigidly (TODO)

文本插入命令

文本插入相关命令(以 `i` 开头):

Key binding	Description
<code>SPC i l l</code>	insert lorem-ipsum list
<code>SPC i l p</code>	insert lorem-ipsum paragraph
<code>SPC i l s</code>	insert lorem-ipsum sentence
<code>SPC i p 1</code>	insert simple password
<code>SPC i p 2</code>	insert stronger password
<code>SPC i p 3</code>	insert password for paranoids
<code>SPC i p p</code>	insert a phonetically easy password
<code>SPC i p n</code>	insert a numerical password
<code>SPC i u</code>	Search for Unicode characters and insert them into the active buffer.
<code>SPC i u 1</code>	insert UUIDv1 (use universal argument to insert with CID format)
<code>SPC i u 4</code>	insert UUIDv4 (use universal argument to insert with CID format)
<code>SPC i u U</code>	insert UUIDv4 (use universal argument to insert with CID format)

Increase/Decrease numbers

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

In transient state:

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

Tips: you can increase or decrease a value by more than once by using a prefix argument (i.e. `10 SPC n +` will add 10 to the number under point).

Replace text with iedit

SpaceVim uses powerful iedit mode to quickly edit multiple occurrences of a symbol or selection.

Two new modes: `iedit-Normal` / `iedit-Insert`

The default color for iedit is `red` / `green` which is based on the current colorscheme.

iedit states key bindings

State transitions:

Key Binding	From	to
<code>SPC s e</code>	normal or visual	iedit-Normal

In iedit-Normal mode:

`iedit-Normal` mode inherits from `Normal` mode, the following key bindings are specific to `iedit-Normal` mode.

Key Binding	Description
<code>Esc</code>	go back to <code>Normal</code> mode
<code>i</code>	switch to <code>iedit-Insert</code> mode, same as <code>i</code>
<code>a</code>	switch to <code>iedit-Insert</code> mode, same as <code>a</code>
<code>I</code>	go to the beginning of the current occurrence and switch to <code>iedit-Insert</code> mode
<code>A</code>	go to the end of the current occurrence and switch to <code>iedit-Insert</code> mode
<code><Left></code> / <code>h</code>	Move cursor to left
<code><Right></code> / <code>l</code>	Move cursor to right
<code>0</code> / <code><Home></code>	go to the beginning of the current occurrence
<code>\$</code> / <code><End></code>	go to the end of the current occurrence
<code>D</code>	delete the occurrences
<code>S</code>	delete the occurrences and switch to <code>iedit-Insert</code> mode
<code>gg</code>	go to first occurrence
<code>G</code>	go to last occurrence
<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><Tab></code>	toggle current occurrence

In iedit-Insert mode:

Key Binding	Description
<code>Esc</code>	go back to <code>iedit-Normal</code> mode
<code><Left></code>	Move cursor to left
<code><Right></code>	Move cursor to right
<code><C-w></code>	delete words before cursor
<code><C-K></code>	delete words after cursor

Examples

注释(Commentings)

注释(comment)通过下面的工具来处理 [nerdcommenter](#)，它用下面的按键来界定范围。

Key Binding	Description
<code>SPC ;</code>	comment operator
<code>SPC c h</code>	hide/show comments
<code>SPC c l</code>	comment lines
<code>SPC c L</code>	invert comment lines
<code>SPC c p</code>	comment paragraphs
<code>SPC c P</code>	invert comment paragraphs
<code>SPC c t</code>	comment to line
<code>SPC c T</code>	invert comment to line
<code>SPC c y</code>	comment and yank
<code>SPC c Y</code>	invert comment and yank

小提示：用 `SPC ; SPC j l` 组合键高效的注释一个文本块的所有内容。

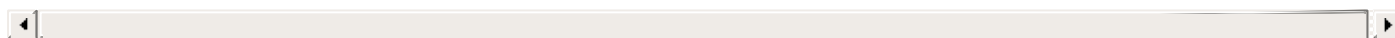
多方式编码

SpaceVim 默认使用 utf-8 码进行编码。下面是 utf-8 编码的四个设置：

- fileencodings (fencs): ucs-bom,utf-8,default,latin1
- fileencoding (fenc): utf-8
- encoding (enc): utf-8
- termencoding (tenc): utf-8 (only supported in vim)

修复混乱的显示：`SPC e a` 是自动选择文件编码的按键映射。在选择好文件编码方式后，你可以运行下面的代码来修复编码：

```
1. set enc=utf-8
2. write
```



错误处理

错误处理

SpaceVim 通过 `neomake` `fly` 工具来进行错误反馈。默认在操作保存时进行错误检查。

错误管理导航键 (以 `e` 开头):

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

下一个/上一个 错误导航键 和 错误暂态(error transinet state) 可用于浏览语法检查器和位置列表缓冲区的错误，甚至可检查vim位置列表的所有错误。这包括下面的例子：在已被保存的位置列表缓冲区进行搜索。默认提示符：

Symbol	Description	Custom option
<code>*</code>	Error	<code>g:spacevim_error_symbol</code>
<code>></code>	warning	<code>g:spacevim_warning_symbol</code>
<code>?</code>	Info	<code>g:spacevim_info_symbol</code>

工程管理

工程管理

SpaceVim 中的工程通过 vim-projectionisst 和 vim-rooter 进行管理。当发现一个 `.git` 目录 或在文件树中发现 `.projections.json` 文件后 vim-rooter 会自动找到项目的根目录。

工程管理的命令以 `p` 开头：

Key Binding	Description
<code>SPC p '</code>	open a shell in project's root (with the shell layer)

Searching files in project

Key Binding	Description
<code>SPC p f</code>	find files in current project
<code>SPC p /</code>	fuzzy search for text in current project
<code>SPC p k</code>	kill all buffers of current project
<code>SPC p t</code>	find project root
<code>SPC p p</code>	list all projects

EditorConfig

EditorConfig

SpaceVim has support for [EditorConfig](#), a configuration file to “define and maintain consistent coding styles between different editors and IDEs.”

To customize your editorconfig experience, read the [editorconfig-vim package's documentation](#).

Vim Server

Vim Server

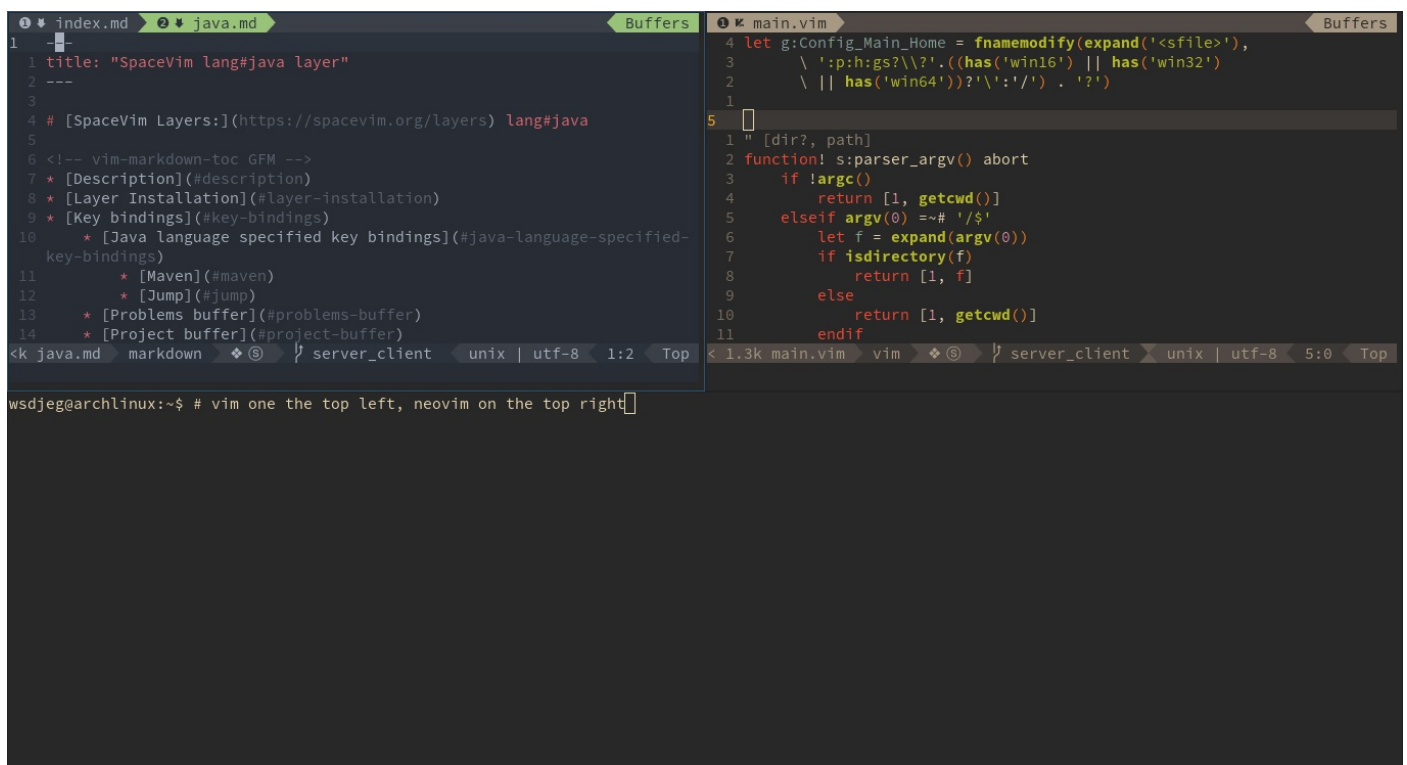
SpaceVim starts a server at launch. This server is killed whenever you close your Vim windows.

Connecting to the Vim server

If you are using neovim, you need to install [neovim-remote](#), then add this to your `bashrc`.

```
1. export PATH=$PATH:$HOME/.SpaceVim/bin
```

Use `svc` to open a file in the existing Vim server, or using `nsvc` to open a file in the existing neovim server.



成就

成就

错误

Achievements	Account
100th issue(issue)	BenBergman

Stars, forks and watchers

Achievements	Account
First stargazers	monkeydterry
100th stargazers	naraj
1000th stargazers	icecity96
2000th stargazers	frowhy
3000th stargazers	purkylin