

# Table of Contents

Introduction	1.1
前言	1.2
安装	1.3
安装 Command Line Developer Tools	1.3.1
安装 MacVim	1.3.2
清理当前 vim 配置	1.3.3
安装 SpaceVim	1.3.4
命令行	1.3.5
字体配置	1.3.6
基础使用	1.4
感受空格键	1.4.1
开始输入	1.4.2
打开文件	1.4.3
文件浏览器	1.4.4
缓冲区管理	1.4.5
窗口管理	1.4.6
Tab 管理	1.4.7
总结	1.4.8
简单配置	1.5
空格键延迟	1.5.1
换个主题	1.5.2
语言支持	1.5.3
字体配置	1.5.4
总结	1.5.5
工程管理	1.6
模糊查找	1.6.1
git 使用	1.6.2
初步进阶	1.7
命令行	1.7.1
Terminal	1.7.2

---

Python开发环境	1.7.3
小知识	1.8
1:相对行号	1.8.1
2:代码注释	1.8.2
3:录制宏	1.8.3
4:TagBar	1.8.4
5:设置文件语法类型	1.8.5
6:grep当前文件(缓冲区)	1.8.6
下一步	1.9

---

# SpaceVim入门教程

！！！本教程多数操作已经不再适合最新版本的*SpaceVim*，仅供参考

GitBook([请点击这里查看这本电子书](#))

The screenshot shows a Mac OS X desktop environment with a dark-themed window titled "Startify - (~) - VIM". Inside the window, there is a terminal-like interface for SpaceVim. The interface includes a large ASCII art logo at the top, followed by version information ("version : 0.7.0-dev by : spacevim.org"). Below this, a list of recently used files is displayed, starting with ".viminfo", "github/Mach0Explorer/moex-cli/src/impl/CommonDisplay.cpp", and ".SpaceVim.d/init.vim". A sidebar on the right lists various directory paths under "[in]: ~/". At the bottom of the window, there is a status bar with icons for Startify, vimfiler, and file system information (utf-8, 16:5, All).

```
[e] <empty buffer>
My most recently used files in the current directory:
[0] .viminfo
[1] github/Mach0Explorer/moex-cli/src/impl/CommonDisplay.cpp
[2] .SpaceVim.d/init.vim
My most recently used files:
[3] ~/.viminfo
[4] ~/github/Mach0Explorer/moex-cli/src/impl/CommonDisplay.cpp
[5] ~/.SpaceVim.d/init.vim
[q] <quit>
```

```
[in]: ~/
> Applications/
> Desktop/
> Documents/
> Downloads/
> exploit/
> github/
> go/
> Library/
> Movies/
> Music/
> mybin/
> onedrive/
> Papers/
> person/
> Pictures/
> projects/
> Public/
> qt/
> re/
> script/
> sec/
> software/
> topic/
> 2018/
```

# 前言

SpaceVim是作者受spacemacs启发而开发的一套VIM配置，对新手来说，与其他vim配置的重要不同是“存在快捷键提示”，因此上手十分容易，使用时也减轻了大脑的负担。

这篇教程

1. 假设用户已经具备vim的基本编辑能力。
2. 这篇教程目前仅面向macOS平台。
3. 教程每个章节会尽量简单，轻松阅读。

让我们开始吧！

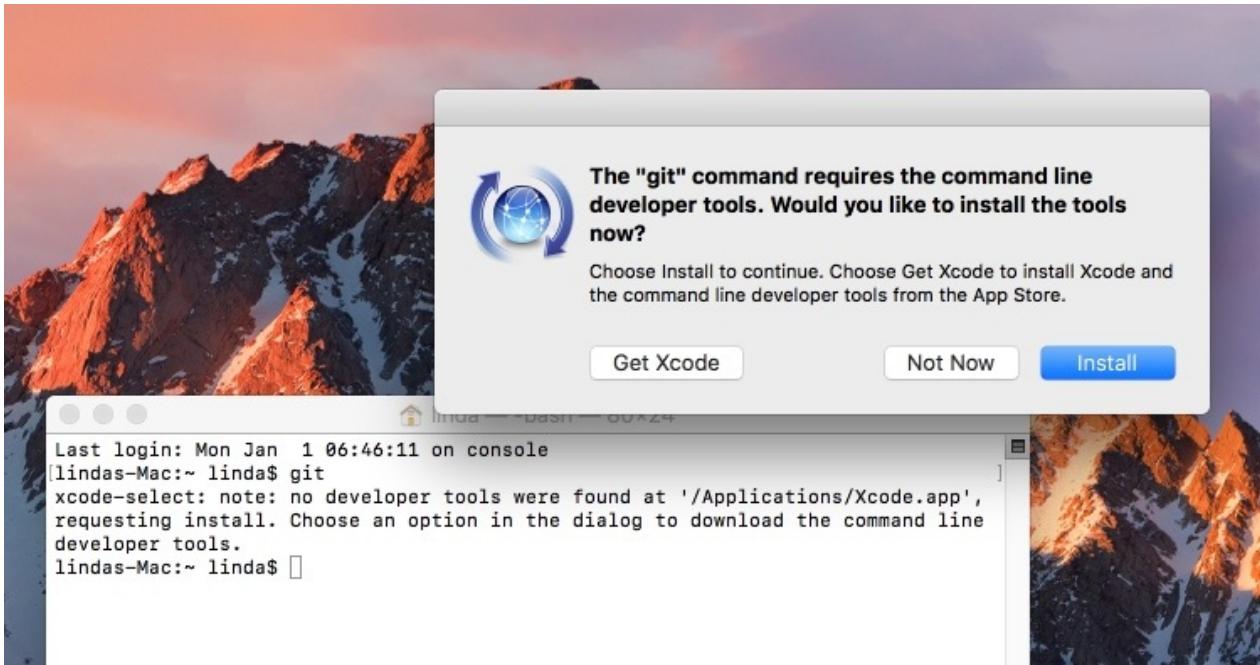
# 安装

这篇教程假设是在一个全新的macOS系统下进行安装，因此读者可以根据情况跳过某个章节。

# 安装Command Line Developer Tools

如果你安装了Xcode并打开使用过，或者你很明白git、curl这些命令行工具已经安装了，那可以跳过这一步。

如果没有安装过，那可以打开Terminal.app 然后输入git回车，系统检测到没有安装Command Line Developer Tools，会提示安装，如下图：



点击Install，然后Agree就可以了。

# 安装 MacVim

安装的方式有很多，这次我们使用Homebrew来安装。

先安装Homebrew，打开Terminal.app，输入：

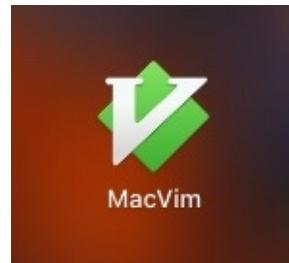
```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

回车后，会提示输入密码，输入密码后，等待安装完成。

然后，安装MacVim。

```
brew install macvim  
brew linkapps
```

(每行表示一次回车)



安装完成后，可以在LaunchPad看到MacVim了。

## 清理当前**vim**配置

打开Terminal.app

```
rm -rf ~/.vim  
rm ~/.vim*
```

# 安装SpaceVim

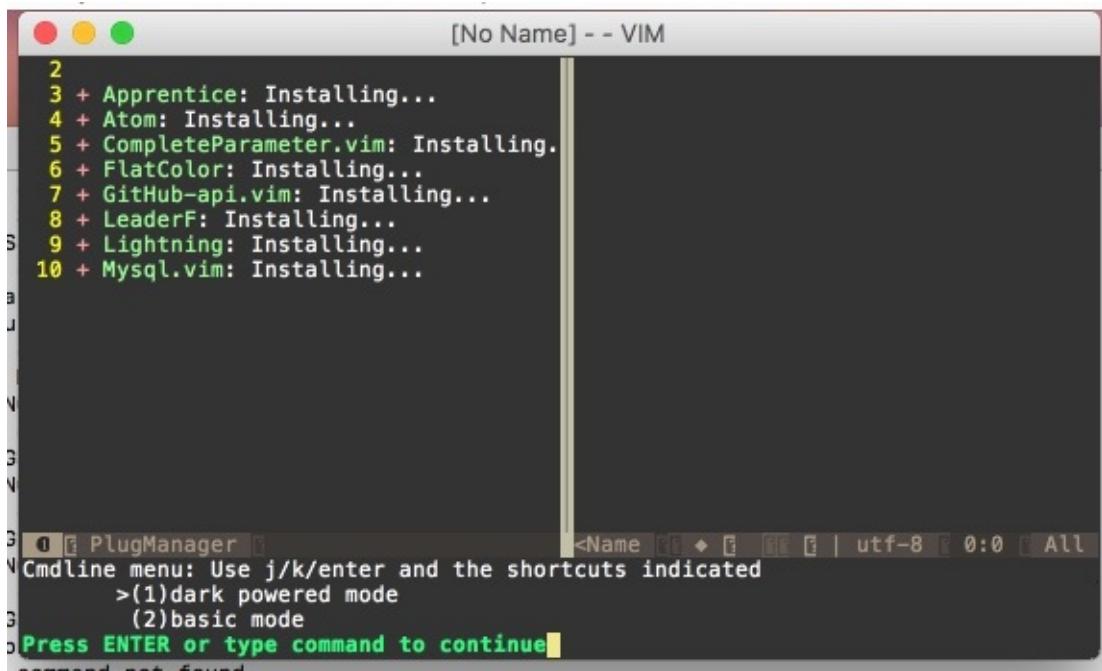
打开Terminal.app

```
curl -sLf https://spacevim.org/install.sh | bash
```

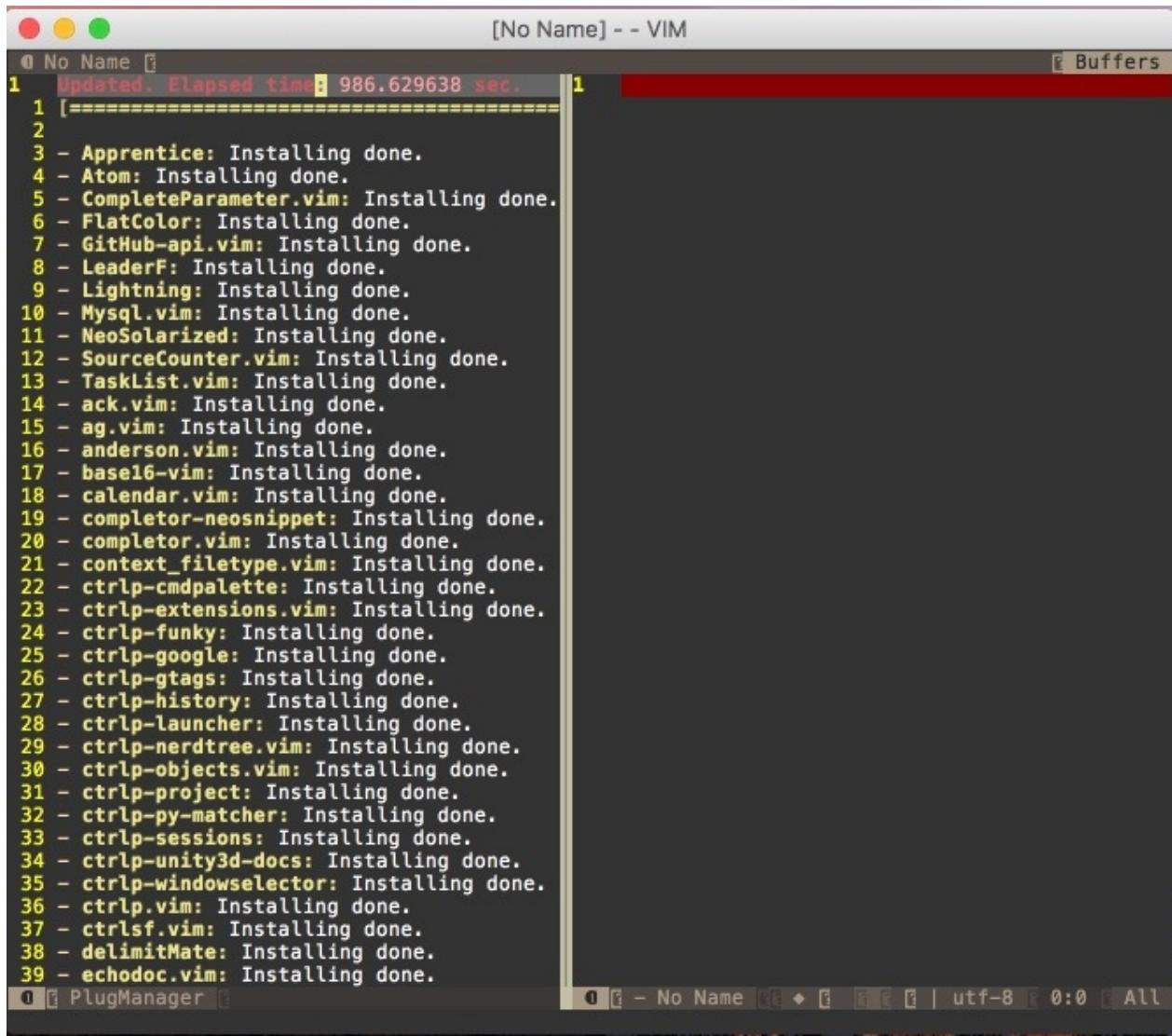


```
linda — git index-pack --stdin -v --fix-thin --keep=fetch-pack 512 on lindas-Mac....  
lindas-Mac:~ linda$ curl -sLf https://spacevim.org/install.sh | bash  
==> Trying to clone SpaceVim  
Cloning into '/Users/linda/.SpaceVim'...  
remote: Counting objects: 14245, done.  
Receiving objects: 38% (5414/14245), 1.64 MiB | 34.00 KiB/s
```

完成后运行MacVim.app



首次打开会提示选择一个模式，按键盘“1”选择第一个模式。然后，VIM会安装很多插件，等待安装完成。我这等了16分钟（986秒）安装完成。



The screenshot shows a MacVim window titled "[No Name] -- VIM". The buffer contains the output of a Vim command to update plugins. The output lists 39 plugins and their installation status. Most plugins are marked as "Installing done". The output ends with "echodoc.vim: Installing done." and the message "Updated. Elapsed time: 986.629638 sec.". The status bar at the bottom shows "0 - No Name" and "All".

```
1 [=====
2
3 - Apprentice: Installing done.
4 - Atom: Installing done.
5 - CompleteParameter.vim: Installing done.
6 - FlatColor: Installing done.
7 - GitHub-api.vim: Installing done.
8 - LeaderF: Installing done.
9 - Lightning: Installing done.
10 - Mysql.vim: Installing done.
11 - NeoSolarized: Installing done.
12 - SourceCounter.vim: Installing done.
13 - TaskList.vim: Installing done.
14 - ack.vim: Installing done.
15 - ag.vim: Installing done.
16 - anderson.vim: Installing done.
17 - base16-vim: Installing done.
18 - calendar.vim: Installing done.
19 - completor-neosnippet: Installing done.
20 - completor.vim: Installing done.
21 - context_filetype.vim: Installing done.
22 - ctrlp-cmdpalette: Installing done.
23 - ctrlp-extensions.vim: Installing done.
24 - ctrlp-funky: Installing done.
25 - ctrlp-google: Installing done.
26 - ctrlp-gtags: Installing done.
27 - ctrlp-history: Installing done.
28 - ctrlp-launcher: Installing done.
29 - ctrlp-nerdtree.vim: Installing done.
30 - ctrlp-objects.vim: Installing done.
31 - ctrlp-project: Installing done.
32 - ctrlp-py-matcher: Installing done.
33 - ctrlp-sessions: Installing done.
34 - ctrlp-unity3d-docs: Installing done.
35 - ctrlp-windowselector: Installing done.
36 - ctrlp.vim: Installing done.
37 - ctrlsf.vim: Installing done.
38 - delimitMate: Installing done.
39 - echodoc.vim: Installing done.

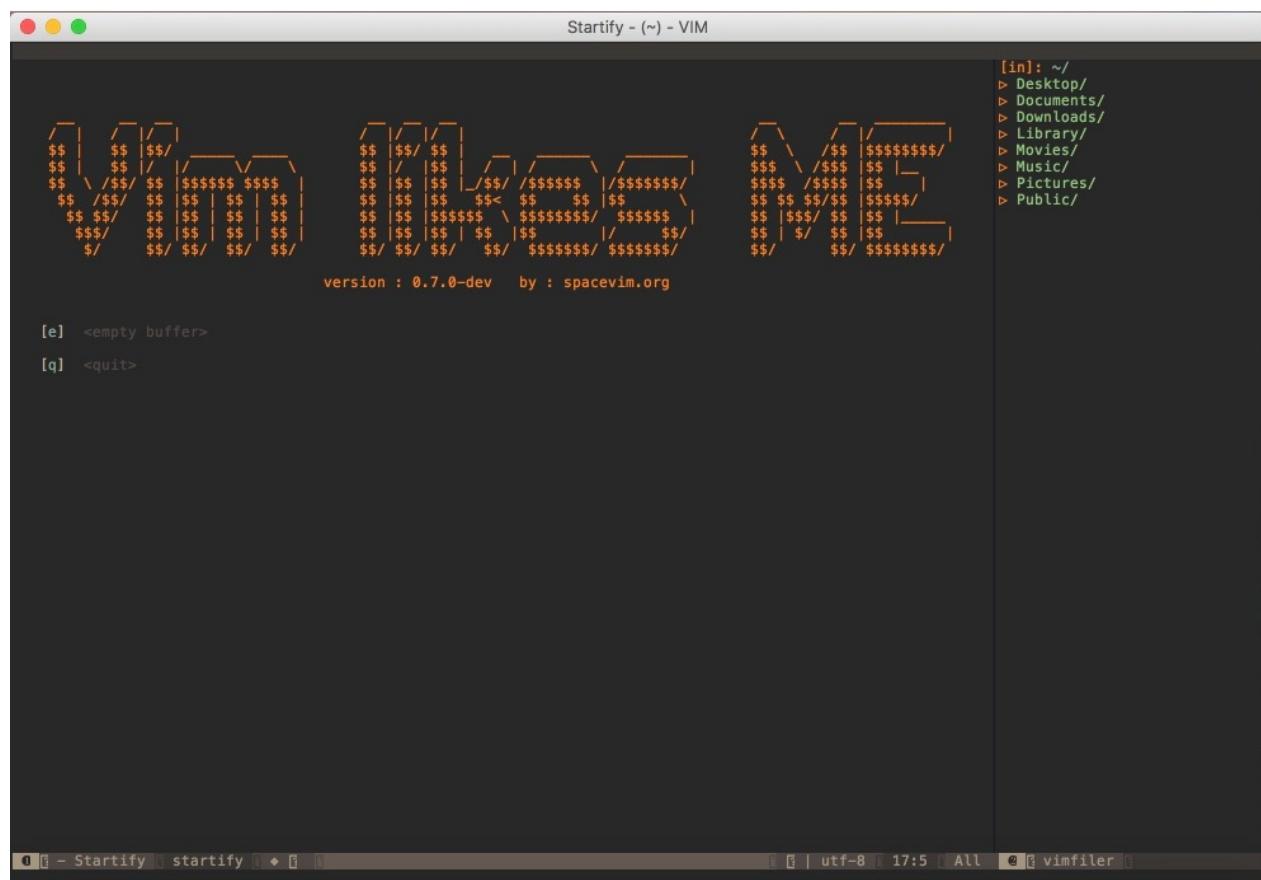
0 - No Name ◆ | utf-8 | 0:0 All
```

好了，退出MacVim，再次打开（再次打开可能有一些VIM插件更新），这次会更新插件，等待更新完成。（有些插件可能更新失败，目前先忽略，不影响当前教程的内容，如果需要的时候再去解决）

The screenshot shows a terminal window titled "[No Name] - - VIM". The window displays the progress of updating 50/52 plugins. Many plugins are listed as "Installing done.", while several others are marked with an 'x' and labeled "Building failed." These failing plugins include "phpcd.vim", "phpfold.vim", "tern\_for\_vim", "vim-import-js", and "vim-javascript". The status bar at the bottom indicates the current buffer is "No Name", the encoding is "utf-8", and the time is "0:0". A tab labeled "PlugManager" is visible on the left.

```
16 Updating plugins (50/52) 1
15 [=====
14
13 - JavaUnit.vim: Installing done.
12 - MatchTagAlways: Installing done.
11 - PHP-Indenting-for-VIM: Installing done.
10 - alchemist.vim: Installing done.
9 - clighter8: Installing done.
8 - es.next.syntax.vim: Installing done.
7 - exception.vim: Installing done.
6 - haskell-vim: Installing done.
5 - helpful.vim: Installing done.
4 - incsearch-easymotion.vim: Installing done
3 - incsearch-fuzzy.vim: Installing done.
2 - incsearch.vim: Installing done.
1 - java_getset.vim: Installing done.
17 - javascript-libraries-syntax.vim: Installi
1 * jedi-vim: Installing 0% (1/334)
2 - lua-support: Installing done.
3 - neco-vim: Installing done.
4 - perl-support: Installing done.
5 - perlomni.vim: Installing done.
6 - php.vim: Installing done.
7 x phpcd.vim: Building failed.
8 x phpfold.vim: Building failed.
9 - rust.vim: Installing done.
10 - screensaver.vim: Installing done.
11 x tern_for_vim: Building failed.
12 - vim-asterisk: Installing done.
13 - vim-autoflake: Installing done.
14 - vim-dict: Installing done.
15 - vim-elixir: Installing done.
16 x vim-import-js: Building failed.
17 - vim-java: Installing done.
18 - vim-javacomplete2: Installing done.
19 - vim-javascript: Installing done.
20 * vim-jsbeautify: Installing 0% (0/471)
21 - vim-jsx-pretty: Installing done.
22 - vim-lookup: Installing done.
23 - vim-lua: Installing done.
```

好了，完成。



# 命令行

现在如果在Terminal.app中输入vim的话，仍然会打开系统的vim（版本较低）。输入以下命令即可在命令行使用MacVim的Terminal模式。

```
cd ~;echo "alias vim=\"mvim -v\\"" >> .bash_profile; source .bash_profile
```

# 字体配置

如果仔细看，可能会发现安装后有些显示问号（?）的地方，这是因为还需要配置一些字体。

这里字体我们使用 <https://github.com/ryanoasis/nerd-fonts>

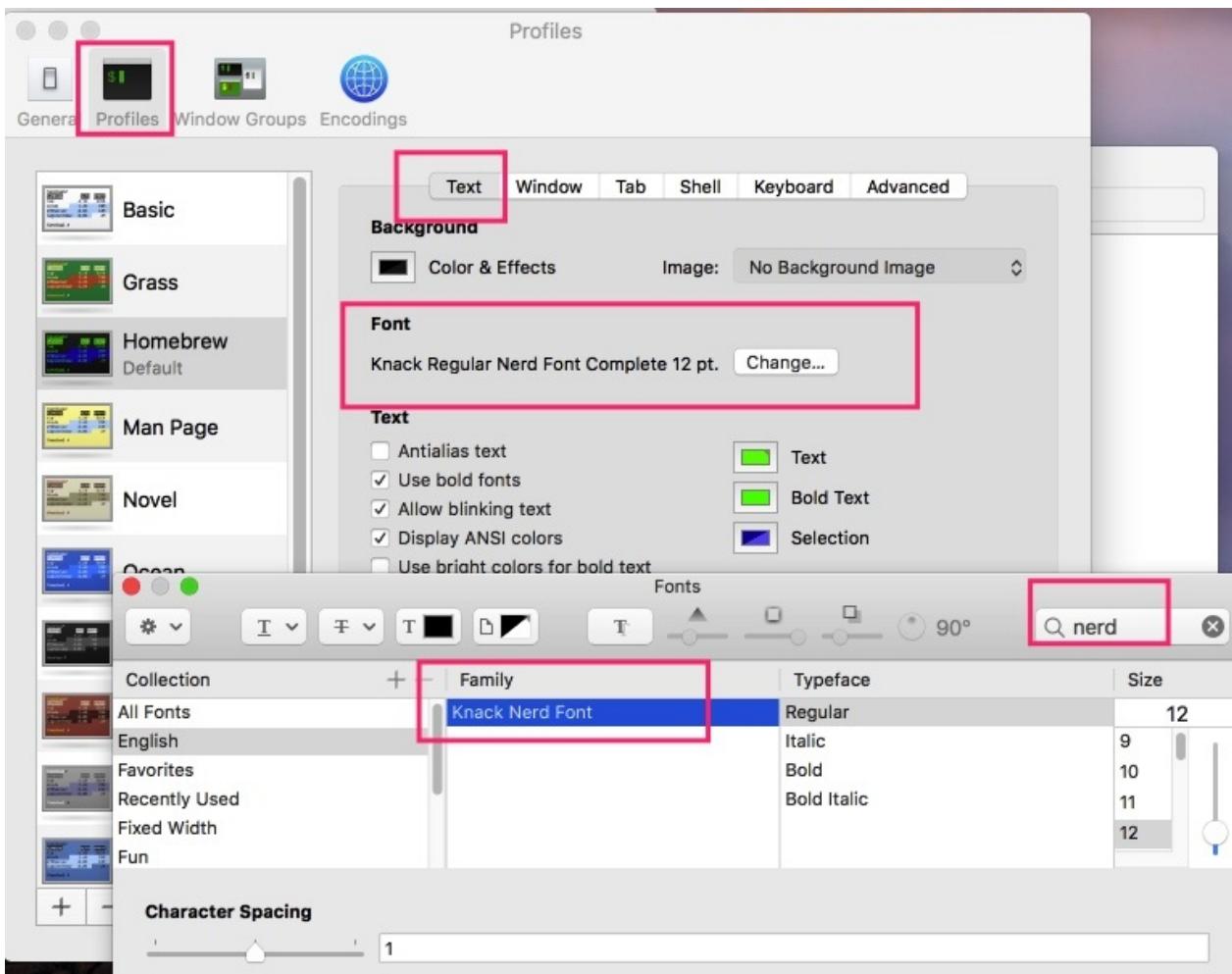
可以这样安装：

```
brew tap caskroom/fonts
brew cask install font-hack-nerd-font
```

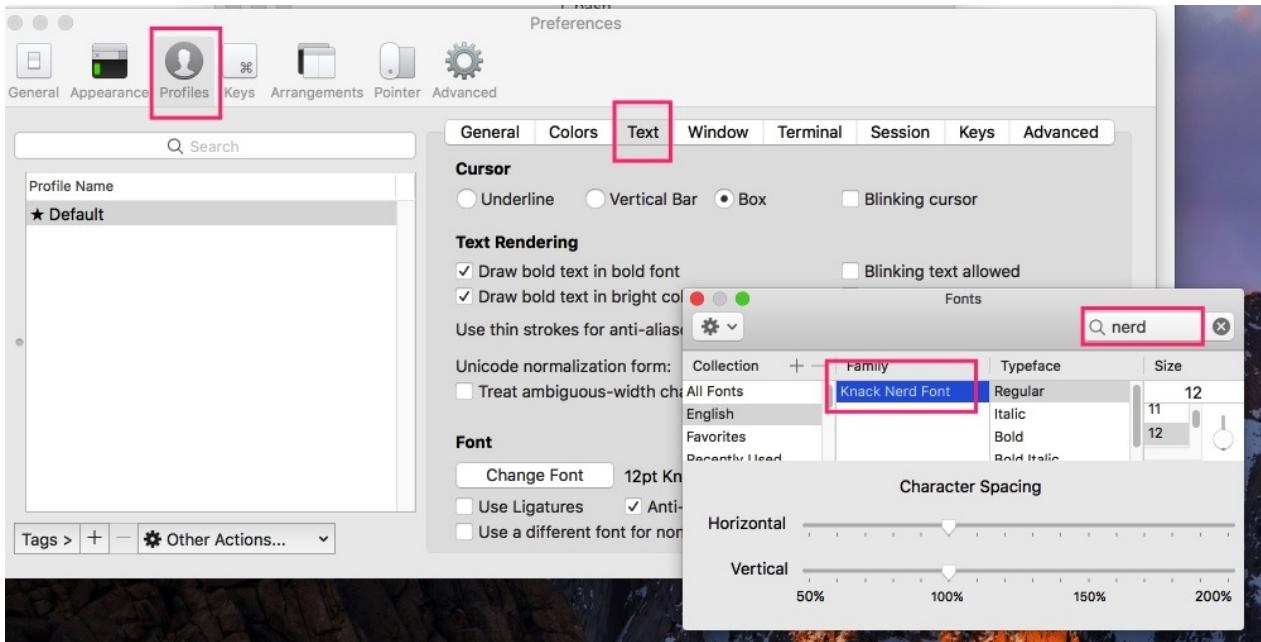
对于MacVim，安装后在配置文件中加入：

```
let g:spacevim_guifont='Knack\ Nerd\ Font:h12'
```

Terminal.app需要单独配置字体：



iTerm2.app也需要单独配置字体：

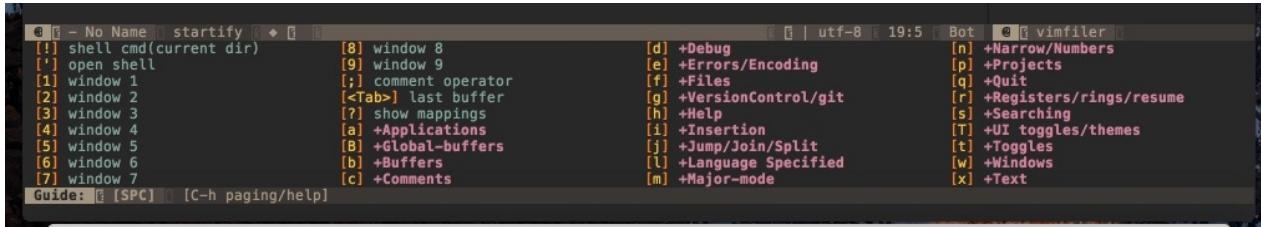


# 基本使用

之所以叫SpaceVim，就是因为很多快捷键是空格键作为第一个键。

# 感受空格键

打开MacVim，按下空格键，1秒后下方会出现一个提示窗口。



如果要关闭这个窗口，可以按 `ESC`。

现在试试输入 空格键 字母f 字母t（以后简称 `SPC ft`），可以关闭右侧的文件浏览器，再次输入 `SPC f t` 可以再打开。

备注：上面的截图中存在?的问题，是制作此教程时，没有配置字体的原因。如果安装后配置了字体，不会存在显示?的情况。配置字体请见“安装->字体配置”。

## 开始输入

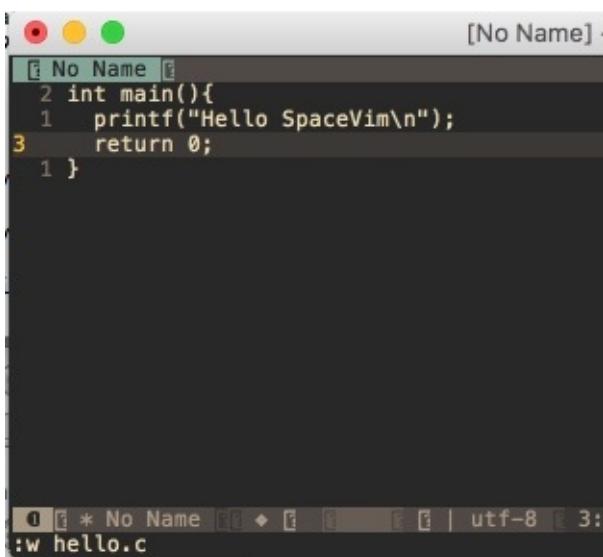


打开MacVim（以后简称vim）后，有两个提示选项 `e` 和 `q`。按下 `e` 会打开一个空的编辑器，按下 `q` 则退出vim。

现在输入 `e`，然后输入一段程序：

```
int main(){
    printf("Hello SpaceVim\n");
    return 0;
}
```

输入 `:w hello.c` 保存。



开始输入

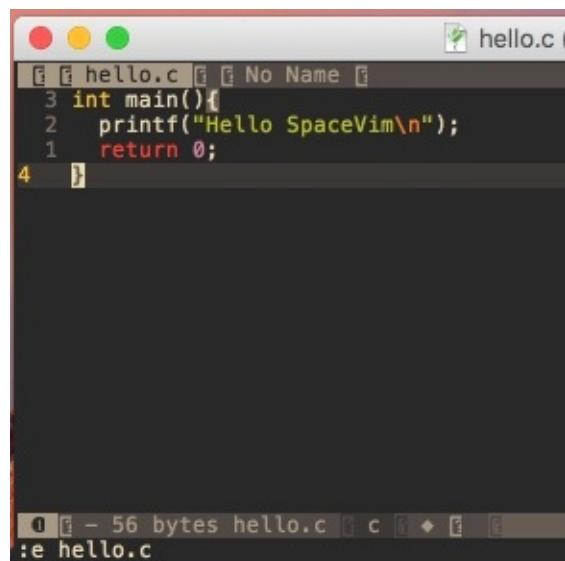
---

# 打开文件

重新打开vim，如果此时我们想打开一个文件，例如打开HOME目录下的hello.c文件，输入 `:e hello.c`。（这都是vim基本的使用方式）



打开后如图：

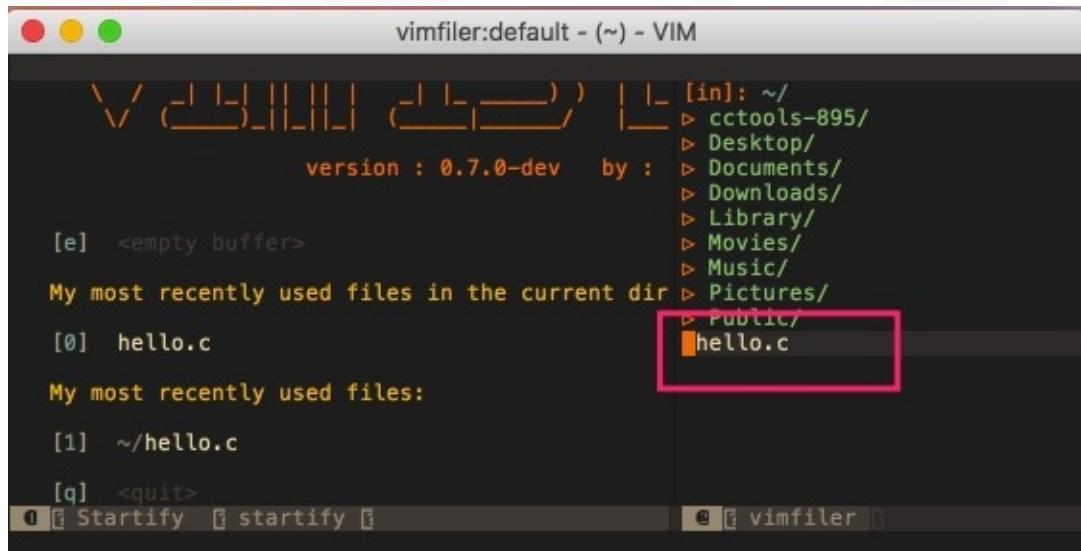


# 文件浏览器

现在再回到右侧的文件浏览器上，试试按下 `<F3>` 也可以打开关闭。也就是 `<F3>` 或者 `SPC f t` 都可以打开关闭文件浏览器。

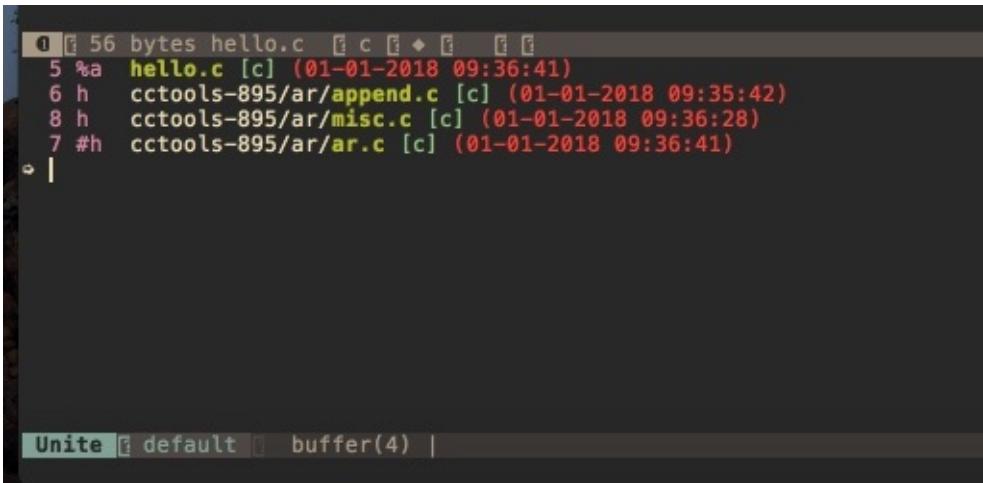
打开文件浏览器后，光标会进入文件浏览器中，这时可以输入 `j`、`k` 上下移动，`h` 和 `l` 可以打开关闭文件夹。

例如下图：按 `j` 移动到 `hello.c` 后，按回车或字母 `l` 都可以打开 `hello.c` 文件。



## 缓冲区管理

vim中打开的每个文件都可称为一个缓冲区（buffer）,如果我们打开了多个文件，可以 `SPC b`



The screenshot shows the Vim buffer list menu. It lists four buffers:

- 0 56 bytes hello.c [c] (01-01-2018 09:36:41)
- 5 %a hello.c [c] (01-01-2018 09:36:41)
- 6 h cctools-895/ar/append.c [c] (01-01-2018 09:35:42)
- 8 h cctools-895/ar/misc.c [c] (01-01-2018 09:36:28)
- 7 #h cctools-895/ar/ar.c [c] (01-01-2018 09:36:41)

At the bottom of the menu, it says "Unite default buffer(4) |".

b 显示列表：

此时可以输入文件名过滤，例如输入 `append`，然后回车，可以切换到 `append.c` 的文件。

此外，这个节目还可以通过 `ctrl+n` 或 `ctrl+p` 来上下移动选择文件。

如果要当前缓冲区，可以使用 `SPC b d`。

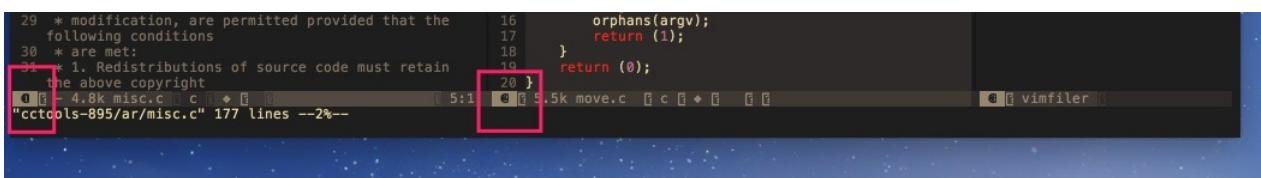
# 窗口管理

输入 `SPC w /` 可在右侧分割窗口，输入 `SPC w -` 可在下方分割窗口。分隔完窗口后，光标会落到文件浏览器，此时可以通过文件浏览器选择某个文件打开文件。在选中文件回车后，会提示将文件放入哪个窗口中。如下图：



此时输入 `a`（不需要大写），就可以把文件放入A窗口。

有了多个窗口，那如何切换呢？仔细看可以注意到 窗口左下角有编号，

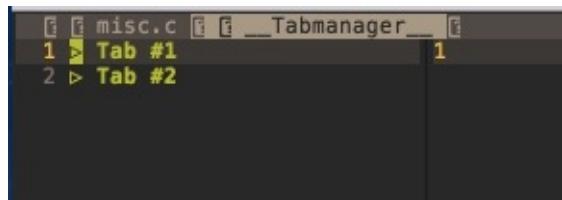


如果要切换到窗口1，则输入 `SPC 1` 即可切换过去。切换到窗口2则输入 `SPC 2`，以此类推。这点可以说是相当方便了。

还可以使用 `SPC w TAB` (`TAB`表示`Tab`键) 顺序切换窗口。

# Tab管理

打开的文件多了，还可以使用 Tab。现在输入 `SPC w F`（注意是大写 F），可以新建一个 Tab 页。然后输入 `SPC t t` 显示 Tab 列表：



在 Tab 中可以上下选择（`j`、`k` 移动光标）然后回车切换。

也可以使用 `SPC w o` 顺序切换 Tab。

## 总结

SpaceVim中的快捷键都容易记忆，窗口（window）就是w，文件（file）就是f，缓冲区（buffer）就是b，Tab管理就是t。再加上按SPC键后，每个功能都有字母提示，几乎不用再去找某个功能的文档。

## 简单配置

SpaceVim提供了我们可以自定义配置的文件，就是`~/.SpaceVim.d/init.vim`文件，所有自定义的内容都可以放到这个文件中。这个文件夹也就可以通过git管理起来。

以下配置可参考[我的配置文件](#)

## 空格键延迟

默认按下空格键是1秒后显示选项，由于刚刚上手，可以设置的短一些，例如200毫秒

```
set timeoutlen=200
```

## 换个主题

所有支持的主题 <https://github.com/rafi/awesome-vim-colorschemes>

```
let g:spacevim_colorscheme = 'onedark'
```

## 语言支持

有些语言我用不到，注释掉吧。（这里根据个人情况来，我主要是用vim来写Python和Go）



## 字体配置

如果仔细看，可能会发现安装后有些显示问号（?）的地方，这是因为还需要配置一些字体。

这里字体我们使用 <https://github.com/ryanoasis/nerd-fonts>

可以这样安装：

```
brew tap caskroom/fonts  
brew cask install font-hack-nerd-font
```

对于MacVim，安装后在配置文件中加入：

```
let g:spacevim_guifont='Knack\ Nerd\ Font:h12'
```

Terminal.app需要单独配置字体：



iTerm2.app也需要单独配置字体：



## 总结

然后，就MacVim像我这样啦。



## 空格键延迟

默认按下空格键是1秒后显示选项，由于刚刚上手，可以设置的短一些，例如200毫秒

```
set timeoutlen=200
```

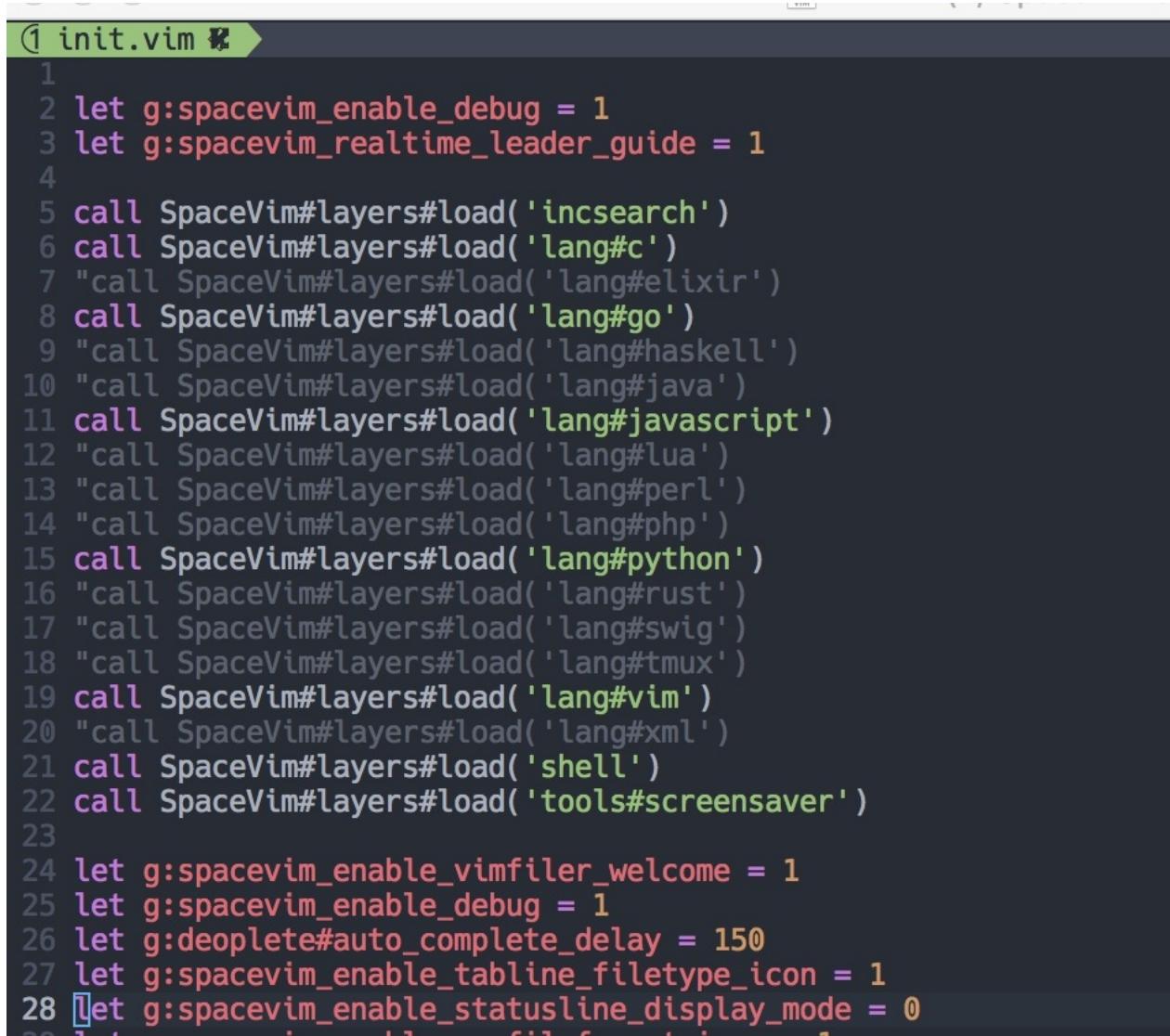
## 换个主题

所有支持的主题 <https://github.com/rafi/awesome-vim-colorschemes>

```
let g:spacevim_colorscheme = 'onedark'
```

# 语言支持

有些语言我用不到，注释掉吧。（这里根据个人情况来，我主要是用vim来写Python和Go）



```
① init.vim ➜
1
2 let g:spacevim_enable_debug = 1
3 let g:spacevim_realtime_leader_guide = 1
4
5 call SpaceVim#layers#load('incsearch')
6 call SpaceVim#layers#load('lang#c')
7 "call SpaceVim#layers#load('lang#elixir')
8 call SpaceVim#layers#load('lang#go')
9 "call SpaceVim#layers#load('lang#haskell')
10 "call SpaceVim#layers#load('lang#java')
11 call SpaceVim#layers#load('lang#javascript')
12 "call SpaceVim#layers#load('lang#lua')
13 "call SpaceVim#layers#load('lang#perl')
14 "call SpaceVim#layers#load('lang#php')
15 call SpaceVim#layers#load('lang#python')
16 "call SpaceVim#layers#load('lang#rust')
17 "call SpaceVim#layers#load('lang#swig')
18 "call SpaceVim#layers#load('lang#tmux')
19 call SpaceVim#layers#load('lang#vim')
20 "call SpaceVim#layers#load('lang#xml')
21 call SpaceVim#layers#load('shell')
22 call SpaceVim#layers#load('tools#screensaver')
23
24 let g:spacevim_enable_vimfiler_welcome = 1
25 let g:spacevim_enable_debug = 1
26 let g:deoplete#auto_complete_delay = 150
27 let g:spacevim_enable_tabline_filetype_icon = 1
28 let g:spacevim_enable_statusline_display_mode = 0
29 let g:spacevim_enable_statusline_filetype_icon = 1
```

## 字体配置

这个章节已经移动到了安装部分。

# 总结

然后，就MacVim像我这样啦。

The screenshot shows the MacVim interface. The main window displays the contents of the 'init.vim' file, which is located at `~/SpaceVim.d`. The file contains numerous Vimscript commands for initializing the SpaceVim environment, including calls to `SpaceVim#layers#load` for various languages like C, Elixir, Go, Haskell, Java, JavaScript, Lua, Perl, PHP, Python, Rust, Swig, and Tmux, as well as for vimlint and vint. It also sets configuration variables such as `g:spacevim_enable_debug`, `g:spacevim_enable_vimfiler_welcome`, and `g:spacevim_enable_statusline_display_mode`. The sidebar on the right lists the current buffers, which include various project directories like Applications, Desktop, Documents, Downloads, exploit, github, go, Library, Movies, Music, mybin, onedrive, Papers, person, Pictures, projects, Public, qt, re, script, sec, software, topic, and 2018.

```

1 let g:spacevim_enable_debug = 1
2 let g:spacevim_realtime_leader_guide = 1
3
4 call SpaceVim#layers#load('incsearch')
5 call SpaceVim#layers#load('lang#c')
6 "call SpaceVim#Layers#load('lang#elixir')
7 call SpaceVim#layers#load('lang#go')
8 "call SpaceVim#Layers#load('lang#haskell')
9 "call SpaceVim#Layers#load('lang#java')
10 "call SpaceVim#layers#load('lang#javascript')
11 call SpaceVim#layers#load('lang#lua')
12 "call SpaceVim#layers#load('lang#perl')
13 "call SpaceVim#layers#load('lang#php')
14 call SpaceVim#layers#load('lang#python')
15 call SpaceVim#layers#load('lang#rust')
16 "call SpaceVim#layers#load('lang#swig')
17 "call SpaceVim#layers#load('lang#tmux')
18 call SpaceVim#layers#load('lang#vim')
19 "call SpaceVim#layers#load('lang+xml')
20 "call SpaceVim#layers#load('lang+xml')
21 call SpaceVim#layers#load('shell')
22 call SpaceVim#layers#load('tools#screensaver')
23
24 let g:spacevim_enable_vimfiler_welcome = 1
25 let g:spacevim_enable_debug = 1
26 let g:deoplete#auto_complete_delay = 150
27 let g:spacevim_enable_tabline_filetype_icon = 1
28 let g:spacevim_enable_statusline_display_mode = 0
29 let g:spacevim_enable_os_fileformat_icon = 1
30 let g:spacevim_buffer_index_type = 1
31 let g:neomake_vim_enabled_makers = []
32
33 if executable('vimlint')
34 | call add(g:neomake_vim_enabled_makers, 'vimlint')
35 endif
36
37 if executable('vint')
38 | call add(g:neomake_vim_enabled_makers, 'vint')
39 endif
40
41 if has('python3')
42 | let g:crlp_map =

```

- 2.1k init.vim vim ◆ ⓘ ⌂ master

## 工程管理

工程的定义：SpaceVim会从当前文件自动向上查找 `.git` 目录或者 `.projections.json` 文件的目录作为根目录。

输入 `SPC p` 可以看到支持的功能。



# 模糊查找

最常用的功能就是模糊查找了，`SPC p /` 输入任意字符串后回车

A screenshot of the vimfiler interface showing a search results window. The title bar says '36.3k CommonDisplay.cpp > cpp > ⌘ ⌘ ⌘ develop >' and 'unix|utf-8 < 0%'. The main area displays 127 matches across various files like FunctionStartsViewNode.h, DynamicSymbolTable.cpp, and SegmentSplitInfoViewNode.h. The bottom status bar shows 'Found 127 matches.' and other vimfiler status indicators.

```
36.3k CommonDisplay.cpp > cpp > ⌘ ⌘ ⌘ develop >
1 libmoex/viewnode/FunctionStartsViewNode.h|14 col 5| MachHeaderPtr mh_;
2 libmoex/viewnode/FunctionStartsViewNode.h|16 col 15| void Init(MachHeaderPtr mh){mh_ = mh;};
3 libmoex/viewnode/DynamicSymbolTable.cpp|12 col 31| void DynamicSymbolTable::Init(MachHeaderPtr mh){
4 libmoex/viewnode/LoadCommandsViewNode.h|14 col 5| MachHeaderPtr mh_;
5 libmoex/viewnode/LoadCommandsViewNode.h|18 col 15| void Init(MachHeaderPtr mh);
6 libmoex/viewnode/TwoLevelHintsTableViewNode.h|14 col 5| MachHeaderPtr mh_;
7 libmoex/viewnode/TwoLevelHintsTableViewNode.h|16 col 15| void Init(MachHeaderPtr mh){mh_ = mh;};
8 libmoex/viewnode/ViewNode.h|120 col 5| MachHeader,
9 libmoex/viewnode/SegmentSplitInfoViewNode.h|14 col 5| MachHeaderPtr mh_;
10 libmoex/viewnode/SegmentSplitInfoViewNode.h|16 col 15| void Init(MachHeaderPtr mh){mh_ = mh;};
- No Name qf ⌘ ⌘ ⌘ develop
Found 127 matches.
```

在这个窗口中可以按 `jk` 上下选择，可以按 `q` 退出。

# git使用

SPC g s 查看状态

```
MachOExplorer:status - (gina:) - VIM
[CommonDisplay.cpp @ >]
On branch develop
Your branch is up-to-date with 'origin/develop'.
nothing to commit, working tree clean
```

```
[in]: ~/Applica/Desktop/Documen/Download/exploit/github/go/Library/Movies/Music/mybin/onedrive/Papers/person/Picture/project/Public/qt/re/script/sec/softwar/topic/2018/
```

```
1 - MachOExplorer:status gina-status ◆ ⓘ ⚡ develop ➔ | utf-8 1:1 All
6 #include <iostream>
7 #include "../util/Utility.h"
8 #include <libmoex/node/LoadCommand.h>
9 #include <string.h>
10 #include <libmoex/viewnode/ViewNodeManager.h>
11 #include <libmoex/node/MachHeader.h>
12
13 using namespace std;
14
15 bool CommonDisplay::Init(const std::string & filepath, bool is_csv){
16     print_ = BeautyTextPrinterFactory::CreatePrinter(is_csv);
17
18     try{
19         bin_ = std::make_shared<moex::Binary>(filepath);
20     }catch(std::exception & ex){
21         cout << "Parse failed : " << ex.what() << endl;
22         return false;
23     }
24
25     return true;
26 }
```

SPC g A 添加所有文件 SPC g S 添加当前文件

SPC g c 设置commit message，最后 :wq 表示完成并commit

```
MachOExplorer:commit + (gina:) - VIM
[CommonDisplay.cpp @ >]
fix bug
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch develop
# Your branch is up-to-date with 'origin/develop'.
#
# Changes to be committed:
#   modified:   moex-cli/src/impl/CommonDisplay.cpp
#
# * MachOExplorer:commit gina-commit ◆ ⓘ ⚡ develop ➔ | utf-8 1:7
7 #include "../util/Utility.h"
8 #include <libmoex/node/LoadCommand.h>
9 #include <string.h>
10 #include <libmoex/viewnode/ViewNodeManager.h>
11 #include <libmoex/node/MachHeader.h>
12
13 using namespace std;
14
15 bool CommonDisplay::Init(const std::string & filepath, bool is_csv){
16     print_ = BeautyTextPrinterFactory::CreatePrinter(is_csv);
17     try{
18         bin_ = std::make_shared<moex::Binary>(filepath);
19     }catch(std::exception & ex){
```

SPC g p 开始push

成功后会有提示：

```
[Location List] :setlocklist()  
[gina] To https://github.com/everettjf/Mach0Explorer.git  
[gina]   f0544a2..32a26b8  develop -> develop  
Press ENTER or type command to continue
```

## 初步进阶

玩点相对“高级”的。

# 命令行

SPC ! 然后输入 ls , 最后 :q 退出

# Terminal

```
:terminal
```

# Python开发环境

安装jedi后可支持自动补全。

```
pip install jedi
```

## 与pyenv配合

假设已经安装了pyenv和Oh my zsh，

1. .bash\_profile
2. .zprofile

在以上两个文件中加入：`export PATH="$HOME/.pyenv/shims:$PATH"`。

## 小知识

一些简单的使用或配置方法，都是我日常开发中使用的。

# 相对行号

SpaceVim默认使用了相对行号，我不习惯：

```
" Relative line number
let g:spacevim_relativenumber = 0
```

把这行加入~/.SpaceVim.d/init.vim中，就好了。

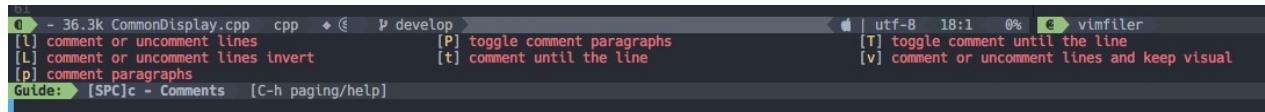
The screenshot shows the Vim interface with the CommonDisplay.cpp file open. On the right side, the vimfiler buffer list is visible, showing various project files like CommonDisplay.h, TestCode.cpp, and Utility.h. A red arrow points from the text above to the 'Buffers' section of the vimfiler list.

```
CommonDisplay.cpp (~/github/MachOExplorer/moex-cli/src/impl) - VIM
[in]: ~/
  Applications/
  Desktop/
  Documents/
  Downloads/
  exploit/
  archive/
  ARM-challenges/
  pwable/
  Stack1.sketch
  github/
  AppleTrace/
  everettjf.github.io/
  MachOExplorer/
  build-Mac..ng_64bit-Debug/
  image/
  libmoex/
  moex-cli/
  cmake-build-debug/
  cmake-build-release/
  libmoex/
  src/
  impl/
    CommonDisplay.cpp
    CommonDisplay.h
    TestCode.cpp
    TestCode.h
  util/
    ArgvParser.h
    BeautyTextPrint.h
    Utility.h
    AppHandler.cpp
    AppHandler.h
    main.cpp
    CMakeLists.txt
  moex-gui/
    libmoex/
    res/
    src/
    CMakeLists.txt
    MachOExplorer.icns
    MachOExplorer.pro
    MachOExplorer.pro.user
    MachOExplorer.qrc
  release/
  release-cli/
  release-gui/
  sample/
  script/
    dist.sh
    dist_cli.sh
    dist_dmg.sh
  LICENSE
  prepare.sh
  README.md
  Qt-Advanced-Docking-System/
  radare2/
  restore-symbol/
  SpaceVim/
  SpaceVimTutorial/
  supotato/
  XBookmark/
  xcode-theme/
  XViM2/
  Yolo/
```

## 注释

`<SPC> c l` 注释选择的行。一般我会 `v` 然后选择多行，然后按 `<SPC> c l` 注释掉选择的行。取消注释也是这样。`c l` 就是 `comment lines` 的意思。

当然按下 `<SPC> c` 后会有很多选择，我是感觉记不住也很少用哈。



## 补充

如果觉得 `SPC c l` 稍微麻烦（不同字母嘛），添加下面的代码到 `init.vim`。

```
call SpaceVim#custom#SPC('nmap', ['c', 'c'], '<Plug>NERDCommenterInvert', 'comment or
uncomment lines', 0)
```

就可以使用 `SPC c c` 作为注释快捷键了。

当然相同字母可能就 *hurt your fingers*。

## 录制宏

SpaceVim的录制宏操作，修改了vim的默认快捷键。个人感觉不太好，但或许大佬们有自己的理由。

但习惯了也就没什么问题了。

Normal mode 下按下 \ q r 开始或结束录制。

# TagBar

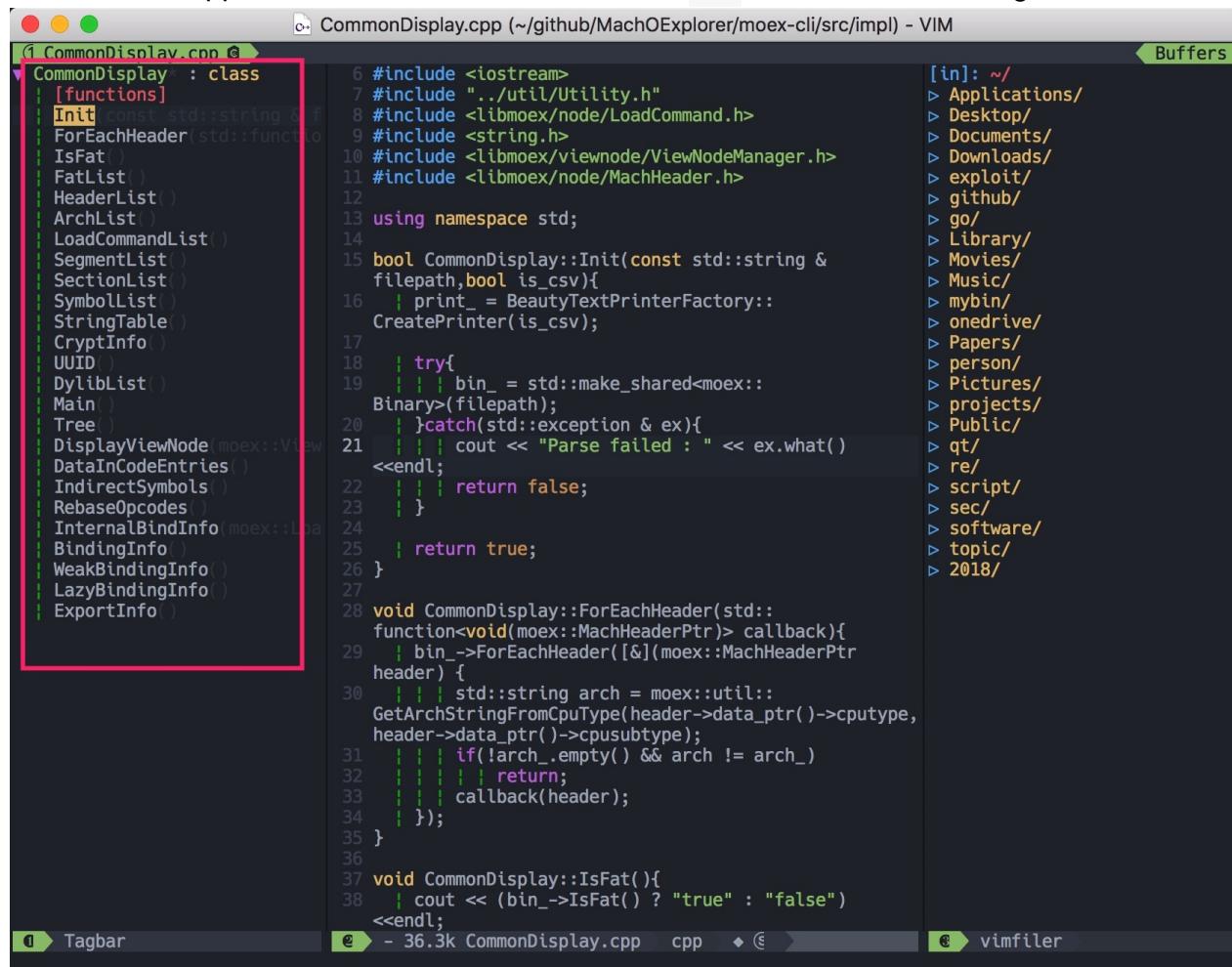
需要先安装ctags，macOS下可以如下安装：

```
brew install ctags
```

安装后可以如下看下是否安装成功：

```
[everettjf@e ~ ]$ ctags --version
Exuberant Ctags 5.8, Copyright (C) 1996-2009 Darren Hiebert
Compiled: Sep 17 2017, 20:48:43
Addresses: <dhiebert@users.sourceforge.net>, http://ctags.sourceforge.net
Optional compiled features: +wildcards, +regex
```

然后打开一个cpp文件（或者其他源代码文件），按 F2 就可以在左侧看到TagBar啦。



The screenshot shows a VIM session with the file 'CommonDisplay.cpp' open. The left margin displays the TagBar, which lists various functions and methods defined in the code. A red box highlights the 'CommonDisplay' class definition in the TagBar. The main editor area shows the source code, and the status bar at the bottom indicates the file name, size, and line count.

```
CommonDisplay : class
[functions]
Init(const std::string &f
ForEachHeader(std::function<
IsFat()
FatList()
HeaderList()
ArchList()
LoadCommandList()
SegmentList()
SectionList()
SymbolList()
StringTable()
CryptInfo()
UUID()
DylibList()
Main()
Tree()
DisplayViewNode(moex::View
DataInCodeEntries()
IndirectSymbols()
RebaseOpcodes()
InternalBindInfo(moex::ba
BindingInfo()
WeakBindingInfo()
LazyBindingInfo()
ExportInfo()
```

可以 SPC 1 移动到TagBar的窗口，选择后回车就可以跳转到对应的函数（方法）。



## 设置文件语法类型

作为一名主页iOS开发，打开Objective-C源文件(\*.m)时，SpaceVim会识别为matlab文件，添加如下代码可以修复过来：

```
" Set .m file type default to objc  
autocmd BufEnter *.m setlocal ft=objc
```

The screenshot shows a terminal window with the following content:

```
83  
84 static __attribute__((constructor)) void CHConstructor18()  
85 {  
86     | CHLoadClass_(&NSString$, [NSString class]);  
87     | $NSString_stringByDeletingLastPathComponent_register();  
88 }  
89  
90 int main(int argc, const char * argv[]) {  
91     | @autoreleasepool {  
92         |objc| ④ ② ◆ ⓘ ⌂ master
```

The word "objc" is highlighted with a red rectangle. The status bar at the bottom of the terminal shows the file name "main\_core.m", line count "3.1k", and two error counts "④ ②".

## grep当前文件（缓冲区）

SPC s s 可以在当前打开的文件（缓冲区）执行grep搜索。

例如下图grep brew 四个字符。

1 1.6k README.md > markdown > ◆ ⓘ > ⚡ develop >  
 57 brew install boost --c++11  
 56 brew install cmake  
 31 brew install moex  
 30 brew tap everettjf/tap  
 24 brew cask install machoexplorer  
 21 (2) or use brew cask  
 ↵ **brew**

Unite > default > line(6/93) | Target: README.md

输入完成 brew 后，可以按 esc ，然后就可以按 jk 来上下选择，在想定位的一行回车，就可以直接到目的行。如果不想回车，可以按 q 退出。

1 1.6k README.md > markdown > ◆ ⓘ > ⚡ develop >  
 57 brew install boost --c++11  
 56 brew install cmake  
 31 **brew** install moex  
 30 brew tap everettjf/tap  
 24 brew cask install machoexplorer  
 21 (2) or use brew cask  
 ↵ **brew**

## 下一步

1. 阅读一遍文档 <http://spacevim.org/documentation/>。
2. 加入QQ群 121056965 或 [Gitter](#)，和大家交流。