
Wiki

发布 1.0

欧阳松

2019 年 12 月 22 日

1	开发工具	1
1.1	搜索 RG	1
1.2	分屏 Tmux	1
1.3	编辑器 Vim	3

1.1 搜索 RG

最快的搜索工具

Header row, column 1 (header rows optional)	Header 2	Header 3	Header 4
body row 1, column 1	column 2	column 3	column 4
body row 2	

1.2 分屏 Tmux

1.2.1 安装

一般都是安装在服务端，所以主要记录 Linux 版本的安装过程。这里主要记录没有 ROOT 的时候安装最新版 Tmux，因为有 ROOT 的话有很多种办法安装到最新版的软件。主要需要注意的地方就是 configure 和 make 的时候要添加寻找头文件的地方。

```
$ mkdir $HOME/.local
$ curl -LO https://github.com/tmux/tmux/releases/download/3.0a/tmux-3.0a.tar.gz
$ curl -LO https://github.com/libevent/libevent/releases/download/release-2.1.11-
↪stable/libevent-2.1.11-stable.tar.gz
$ curl -LO https://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz
```

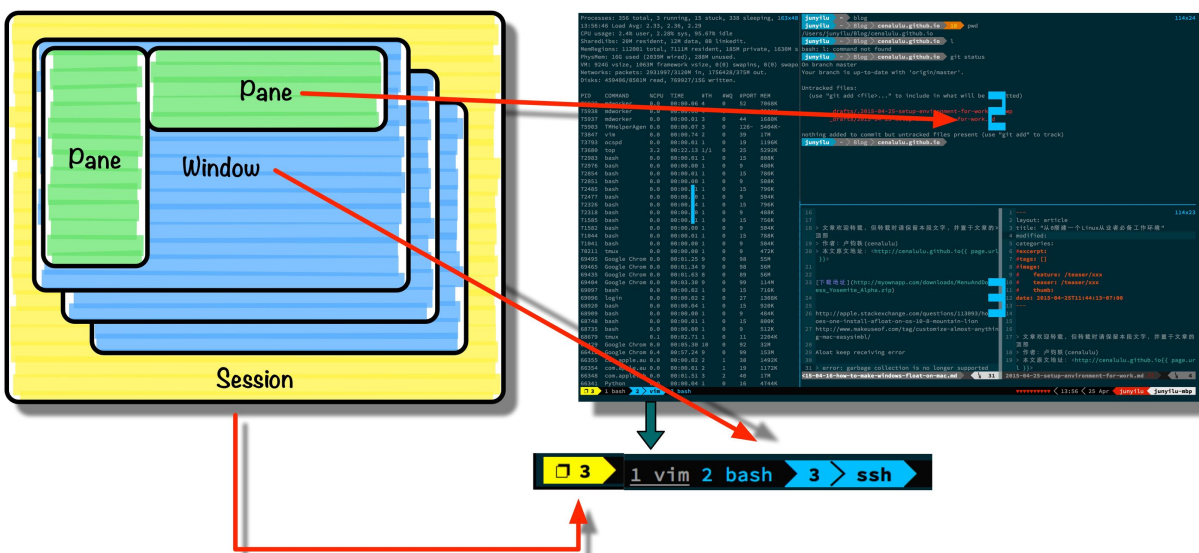
(下页继续)

(续上页)

```

$ tar zxvf libevent-2.1.11-stable.tar.gz && cd libevent-2.1.11-stable
$ ./configure --prefix=$HOME/.local --disable-shared
$ make && make install && cd ..
$ tar zxvf ncurses-5.9.tar.gz && cd ncurses-5.9/
$ ./configure --prefix=$HOME/.local
$ make && make install && cd ..
$ tar zxvf tmux-3.0a.tar.gz && cd tmux-3.0a/
$ ./configure CFLAGS="-I$HOME/.local/include -I$HOME/.local/include/ncurses" LDFLAGS=
  ↳ "-L$HOME/.local/lib -L$HOME/.local/include/ncurses -L$HOME/.local/include" --prefix=
  ↳ $HOME/.local
$ CPPFLAGS="-I$HOME/.local/include -I$HOME/.local/include/ncurses" LDFLAGS="-static -L
  ↳ $HOME/.local/include -L$HOME/.local/include/ncurses -L$HOME/.local/lib" make
$ make install && cd ..
$ export PATH=$HOME/.local/bin:$PATH
$ tmux -V
tmux 3.0a

```



1.2.2 配置

基于网上很热门的配置 [gpakosz/.tmux](#) 进行修改。主要修改的地方是：

```

# 使用 vi 键位
set -g status-keys vi
set -g mode-keys vi

# PREFIX-Q 显示编号保留时间, 单位 ms

```

(下页继续)

(续上页)

```
set -g display-panes-time 5000

# 判断终端是否支持真彩色决定 Tmux 开启真彩色
tmux_conf_theme_24b_colour=true
```

1.2.3 Tmux Plugin Manager

```
$ git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
$ bash ~/.tmux/plugins/tpm/bin/install_plugins
```

然后修改 `.tmux.conf.local` 文件。

```
setenv -g TMUX_PLUGIN_MANAGER_PATH '~/.tmux/plugins'
# List of plugins
set -g @tpm_plugins '
tmux-plugins/tpm
tmux-plugins/tmux-resurrect
tmux-plugins/tmux-continuum
'
# tmux-resurrect
set -g @resurrect-dir '~/.tmux/resurrect'

# 初始化 TPM 插件管理器 (放在配置文件的最后)
run '~/.tmux/plugins/tpm/tpm'
```

1.2.4 参考链接

<https://gist.github.com/ryin/3106801>

1.3 编辑器 Vim

为了使用更加现代化的 Vim 插件以及异步等特性，推荐使用 Vim8。

1.3.1 安装

macOS

```
$ brew install vim
```

Linux

```
$ curl -LO https://github.com/vim/vim/archive/v8.1.2418.tar.gz
$ tar zvxf v8.1.2418.tar.gz && cd vim-8.1.2418
$ LDFLAGS=-L$HOME/.local/lib ./configure --with-features=huge \
> --enable-multibyte \
> --enable-cscope \
> --prefix=$HOME/.local \
> --with-compiledby="ouyangsong" \
> --enable-pythoninterp=yes \
> --with-python-config-dir=/usr/lib/python2.7/config-x86_64-linux-gnu \
> --enable-fail-if-missing
```

如果想直接通过 APT 安装带有 Python 支持的 Vim，可以参考下面的命令。

```
$ sudo apt-get remove --purge vim vim-runtime vim-gnome vim-tiny vim-common vim-gui-
↪common
$ sudo apt install vim-nox
```

如果使用 Linuxbrew 管理软件的话，参考上面 macOS 的安装方法。

1.3.2 依赖

必须的依赖

- Node.js & npm
- ccls
- universal-ctags

可选的依赖

- ripgrep 强大的搜索软件
- protoc-gen-lint proto 文件的检测工具
- flake8 Python 代码风格检测

1.3.3 插件

网上很多 Vim 配置由于年久失修，其中推荐的插件很多没有使用 Vim8 的新特性或者性能存在问题。下面介绍日常编程中使用频率高的插件。

外观

- [gruvbox-community/gruvbox](#)

gruvbox 项目之前的作者没有继续维护，社区版本还有热心志愿者继续维护该项目。所以如果想用 gruvbox 主题的话，推荐使用社区版。

235 bg #282828	0	124 red #cc241d	1	106 green #98971a	2	172 yellow #d79921	3	66 blue #458588	4	132 purple #b16286	5	72 aqua #689d6a	6	246 gray #a89984	7
245 gray #928374	8	167 red #fb4934	9	142 green #b8bb26	10	214 yellow #fabd2f	11	109 blue #83a598	12	175 purple #d3869b	13	108 aqua #8ec07c	14	223 fg #ebdbb2	15
234 bg0_h #1d2021	0*	235 bg0 #282828	0	237 bg1 #3c3836	-	239 bg2 #504945	-	241 bg3 #665c54	-	243 bg4 #7c6f64	-	245 gray #928374	8	166 orange #d65d0e	-
		236 bg0_s #32302f	0*	246 fg4 #a89984	7	248 fg3 #bdae93	-	250 fg2 #d5c4a1	-	223 fg1 #ebdbb2	15	229 fg0 #fbf1c7	-	208 orange #fe8019	-

0* - contrast alternative

- [luochen1990/rainbow](#)

通过将不同层次的括号高亮为不同的颜色, 帮助你阅读世界上最复杂的代码。

```

14 # -*- coding: utf-8 -*-
13 import heapq
12
11 def merge(*sequences):
10     heap = [(seq[0], seq) for seq in sequences]
9     heapq.heapify(heap)
8     while heap:
7         x, seq = heapq.heappop(heap)
6         yield seq.pop(0)
5         if seq:
4             heapq.heappush(heap, (seq[0], seq))
3
2 if __name__ == '__main__':
1     A = [[1, 2, 3, 4], [5, 6, 8], [1, (2+1), 5, (3*(3-2)), 7, 9]]
15     #B = ([1, 2, 3, 4], [5, 6, 8], [1, (2+1), 5, (3*(3-2)), 7, 9])
1     print list(merge(*A))
2     print list(merge(*B))

```

- [Yggdroot/indentLine](#)

显示代码缩进的对齐线, 对于写 Python 这类依靠缩进判断语法块的语言还是很有用的。建议只针对需要的文件类型开启, 否则在部分文件类型, 比如 JSON 上会导致显示有问题。

```
1 function indentLineGuide () {
2     // Show off the more subtle glyph for indentLine in this codeblock
3     while (indentLine !== shownOff) {
4         if (!indentedEnough) {
5             indentSomeMore();
6         }
7         else {
8             switch(indentAmount) {
9                 case 'A lot':
10                    indentLess();
11                    break;
12
13                 case 'Not much':
14                    indentMoreStill();
15                    break;
16
17                 case 'Just right':
18                 default:
19                    stopIndenting();
20            }
21        }
22    }
23 }
```

- vim-gitgutter

在左边显示 Git 中文件中每一行被修改的状态。

```

172 #### Hunks
173
174 You can jump between hunks:
175
176 * jump to next hunk (change): `]c`
177 * jump to previous hunk (change): `[c`.
178
179 Both of those take a preceding count.
180
181 To set your own mappings for these, for example `]h` and `[h`:
182
+ 183 In the lines below I have deleted brackets around GitGutter{Next,Prev}Hunk:
+ 184
185 ```viml
~ 186 nmap ]h <Plug>GitGutterNextHunk
~ 187 nmap [h <Plug>GitGutterPrevHunk
188 ```
189
190 You can load all your hunks into the quickfix list with `:GitGutterQuickFix`. Note this i
    ↪ gnore any unsaved changes in your buffers.
191
192 You can stage or undo an individual hunk when your cursor is in it:
193
194 * stage the hunk with `<Leader>hs` or
195 * undo it with `<Leader>hu`.
196
./README.mkd 671 lines 1
1 -nmap ]h <Plug>GitGutterNextHunk
2 -nmap [h <Plug>GitGutterPrevHunk
3 +nmap ]h <Plug>GitGutterNextHunk
4 +nmap [h <Plug>GitGutterPrevHunk
gitgutter:/hunk-preview 4 lines 2

```

- [itchyny/lightline.vim](#)

轻量级但是功能强大的状态栏插件，可以配合显示其他插件的一些插件。

NORMAL	[No Name]		unix	utf-8	no ft	100%	0:1
INSERT	[No Name]		unix	utf-8	no ft	100%	0:1
VISUAL	[No Name]		unix	utf-8	no ft	100%	0:1
REPLACE	[No Name]		unix	utf-8	no ft	100%	0:1

增强

- [dense-analysis/ale](#)

使用 Language Server Protocol 对代码进行错误检查，以及支持对应的修复。需要安装对应的代码检测软件，比如对 C/C++ 代码检查可以安装 `ccls`。

- [scroolouse/nerdcommenter](#)

针对不同的代码，自动选择相对应的注释符号来实现快速注释与取消注释某些代码。

- [tpope/vim-surround](#)

对 Vim 中本身就有的范围词性进行扩充，支持更多范围选择。

- [tpope/vim-repeat](#)

将 Vim 中某些本身只能重复一次的语句扩充至可以无限重复。

- [jiangmiao/auto-pairs](#)

自动补全括号等其他需要闭合的符号。

- [easymotion/vim-easymotion](#)

实现快速移动，启动快速移动后，可以通过两个字符就可以移动到任何位置。

```

559 " -- Find Motion Helper -----
560 function! s:findMotion(num_strokes, direction) "{{{
561   " Find Motion: S,F,T
562   let s:current.is_operator = mode(1) ==# 'no' ? 1 : 0
563   " store cursor pos because 'n' key find motion could be jump to offscreen
564   let s:current.original_position = [line('.'), col('.')]
565   let s:flag.regexp = a:num_strokes == -1 ? 1 : 0
566
567   if g:EasyMotion_add_search_history && a:num_strokes == -1
568     let s:previous['input'] = @/
569   else
570     let s:previous['input'] = get(s:previous, 'input', '')
571   endif
572   let input = EasyMotion#command_line#GetInput(
573     \ a:num_strokes, s:previous.input, a:direction)
574   let s:previous['input'] = input
575
576   " Check that we have an input char
577   if empty(input)
578     redraw | return ''
579   endif
580
581   let re = s:convertRegep(input)
582
583   if g:EasyMotion_add_search_history && a:num_strokes == -1
584     let @/ = re "For textobject: 'gn'
585     call histadd('search',
586       \ substitute(re, '\\c\\|\\C', '', ''))
587
588   " EasyMotion.vim | adjust/lokalto
vim 34% 572:17

```

- [ap/vim-buftabline](#)

在顶部用 `tab` 的方式显示已经存在的 `buffer`。这个功能其实 `vim-airline` 已经自带，但是如果使用 `lightline.vim` 就需要安装本插件。

```

LICENSE  README.md  buftabline.txt  buftabline.vim
15 =====
16 1. Intro                                          buftabline-intro
17
18 This plugin takes over the 'tabline' and renders the buffer list in it instead
19 of a tab list. It is designed with the ideal that it should Just Work: drop it
20 into your setup and you're done. There is only minimal configurable behavior.
doc/buftabline.txt 15,1 11%
77 Buftabline vs. X
78 -----
79
80 As of Nov 15, 2014, here is how Buftabline compares with some other plugins
81 of which I am aware that they offer related functionality, roughly in order
82 of their age.
83
84
85 * [MiniBufExpl](http://www.vim.org/scripts/script.php?script_id=159)
86
87 Obviously no rundown can be complete without the veteran of buffer list
88 plugins, Mini Buffer Explorer. There are two major differences:
89
90 1. Buftabline uses the tabline while MiniBufExpl renders to a special buffer
91 in a split. The tabline is newer than MiniBufExpl, and unlike a buffer, it
92 is guaranteed to stick to the top of the screen within Vim, unaffected by
93 any splits.
94
README.md 83,0-1 57%
```

- [bronson/vim-trailing-whitespace](#)

自动移除多余的空格，包括尾部的空格和含有空格的行。

- [rhysd/accelerated-jk](#)

当长按 J 或者 K 键进行上下移动时会不断加速移动。

补全

- [neoclide/coc.nvim](#)

异步的代码补全引擎，可以管理和安装各个语言的插件。强力推荐！

- [honza/vim-snippets](#)

提供一些常用的代码片段。

1.3.4 参考链接

<https://askubuntu.com/a/602249>

<https://github.com/yycm-core/YouCompleteMe/wiki/Building-Vim-from-source>

<https://www.cnblogs.com/freeweb/p/9321520.html>