



Contents

1. install
 1. install font on window
 2. command for plugins
 1. vim-startify
 2. vim-easymotion
 3. nerdtree
 4. nerdcommenter
 5. DirDiff
 6. EasyGrep
 7. gundo
 8. ctrlp
 9. vim-bookmarks
 10. vim-multiple-cursors
 11. YouCompleteMe
 12. syntastic
 13. matrix
3. useful command
 1. buffer operation
 2. Windows command
 1. Modify window boundary
 2. add map in .vimrc as the following
 3. Navigate between windows
 4. locate currently edited file in NERDTree window
 3. jump in a file
 4. search in vim
 4. map for cscope

install

```
1. git clone https://github.com/tingezhang/vtconfig.git
2. cd vtconfig
3. cp .vimrc .tmux.conf .tmux.status.conf ~/
4. mkdir ~/.vim/bundle
5. git clone https://github.com/gmarik/Vundle.vim.git ~/.vim/bundle/Vundle.vim
6. vim +PluginInstall +qall
7. modify ~/.bashrc, add "alias tmux='tmux -2'" at the end of file
8. install font for status line:
   1. git clone https://github.com/runsisi/consolas-font-for-powerline
   1. put all font into system font directory
   1. config putty or terminal to use powerline-consola font to
```

install font on window

1. copy all powerline-consola font file to "Control Panel\All Control Panel Items\FONTs"
2. restart putty and set font to "powerline consola"

command for plugins

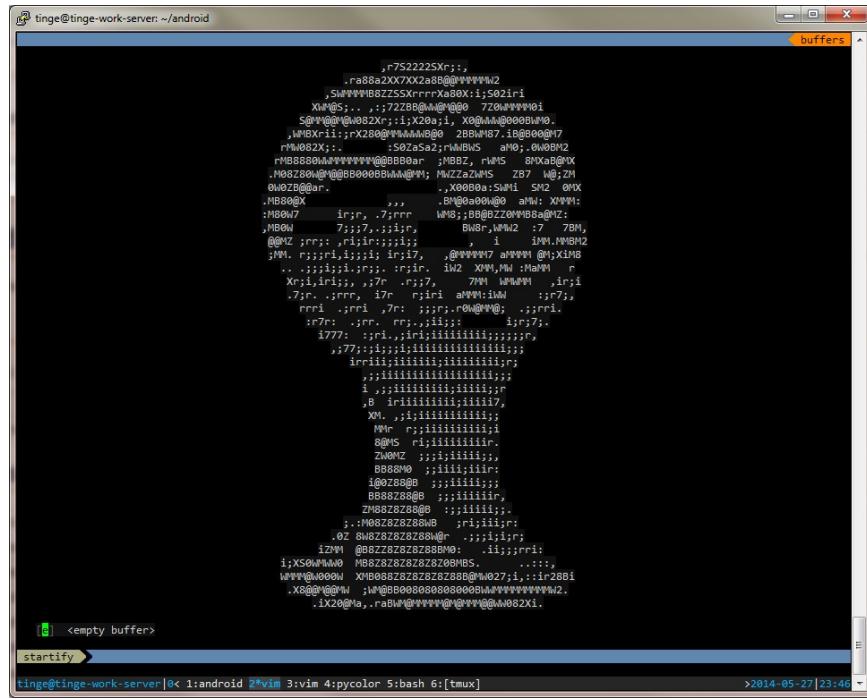
1. some tips for most often used commands/configs for plugins

vim-startify

1. <https://github.com/mhinz/vim-startify>

config	description

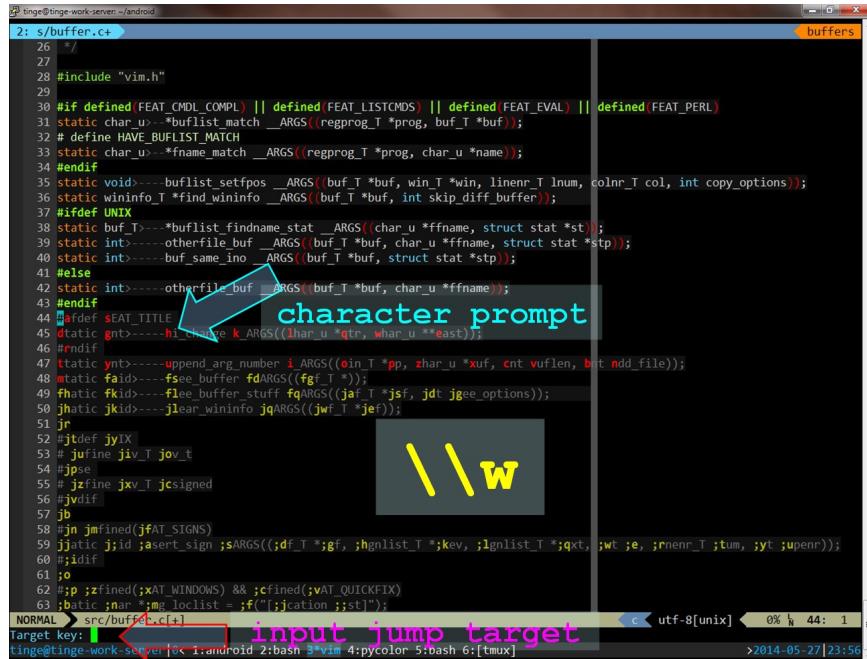
g:startify_custom_header | config the custom header



```
tинге@tinge-work-server:~/android
2: tinge@tinge-work-server:~/android ~*vim 3:vim 4:pycolor 5:bash 6:[tmux]
>2014-05-27[23:46]
```

vim-easymotion

Command	Description
\w	Beginning of word forward
\b	Beginning of word backward
\e	End of word forward
\ge	End of word backward
\j	Line downward
\k	Line upward



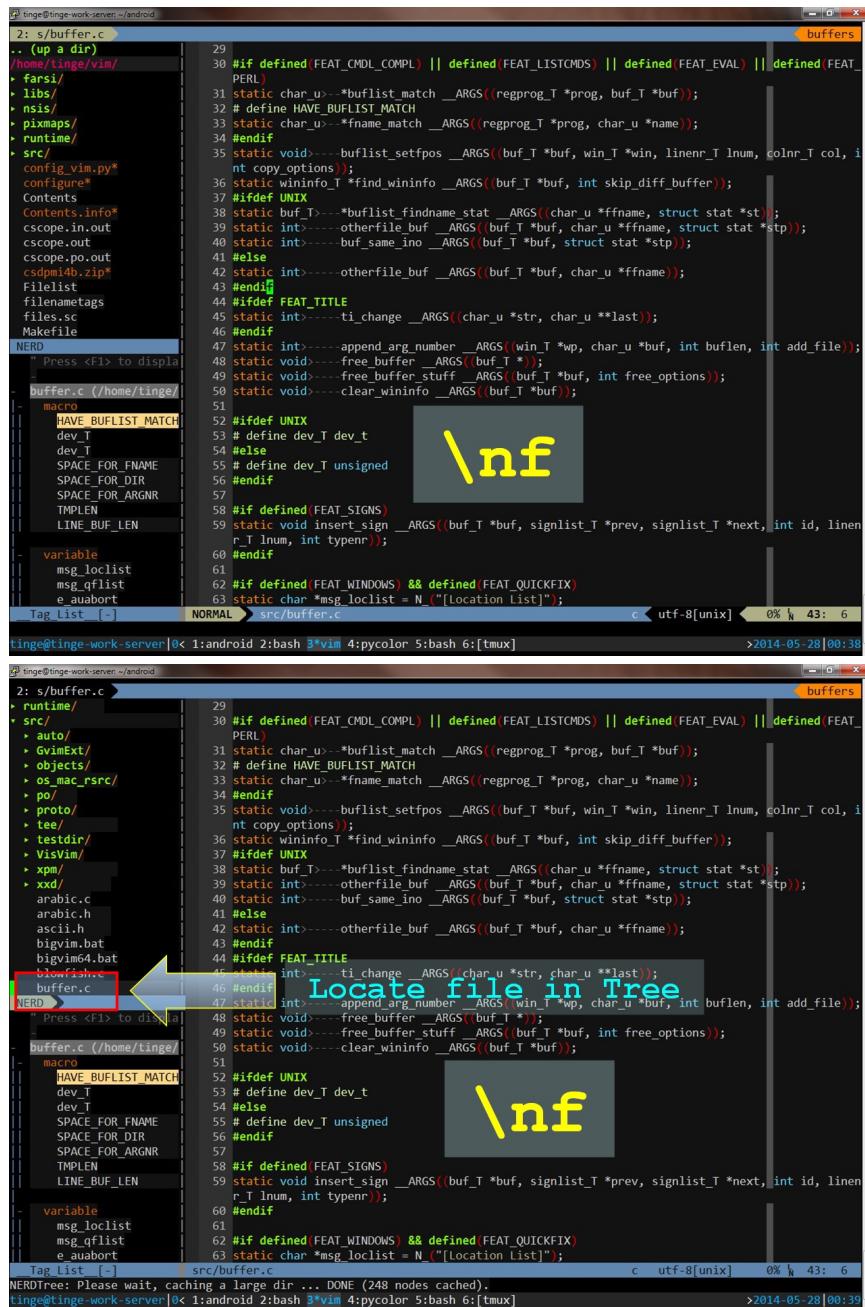
```
2: s/buffer.c+ buffers
26 /*
27
28 #include "vim.h"
29
30 #if defined(FEAT_CMDL_COMPL) || defined(FEAT_LISTCMD) || defined(FEAT_EVAL) || defined(FEAT_PERL)
31 static char_u *buflist_findname_stat__ARGS__(regprog_T *prog, buf_T *buf);
32 #define HAVE_BUFLIST_MATCH
33 static char_u *fname_match__ARGS__(regprog_T *prog, char_u *name);
34 #endif
35 static void---buflist_setfpos__ARGS__(buf_T *buf, win_T *win, linenr_T lnum, colnr_T col, int copy_options);
36 static wininfo_T *find_wininfo__ARGS__(buf_T *buf, int skip_diff_buffer);
37 #ifdef UNIX
38 static buf_T---*buflist_findname_stat__ARGS__(char_u *fname, struct stat *st);
39 static int----otherfile_buf__ARGS__(buf_T *buf, char_u *fname, struct stat *stp);
40 static int----buf_same_ino__ARGS__(buf_T *buf, struct stat *stp);
41 #else
42 static int----otherfile_buf__ARGS__((buf_T *buf, char_u *fname));
43 #endif
44 #define sEAT_TITLE
45 static qnt-----h_change_k__ARGS__(char_u *qtr, wchar_u **east);
46 #endif
47 ttadic ynt-----append_arg_number i__ARGS__(oin_T *po, wchar_u *xuf, cnt vuflen, b7 nnd_file));
48 #static fad-----fsee_buffer fdARGS((fif_T *));
49 #static fkid-----flee_buffer_stuff fqARGS((jaf_T *js, jd7 jgee_options));
50 #static jkdic jk_id-----jclear_wininfo jqARGS((jwf_T *jef));
51 #endif
52 #ifndef jyIX
53 # define jiv_T jov_t
54 #define jipse
55 # define jxfine jxv_T jcsigned
56 #define jydfid
57 #define jb
58 #define jmfinde(jfAT_SIGNS)
59 #define jid ;assert_sign ;SARGS((;df_T *gf, ;hgnlist_T *kev, ;lgnlist_T *qxt, ;wt ;e, ;rnenr_T ;tum, ;yt ;openr));
60 #endif;
61 #endif;
62 #if p ;zfinde(xAT_WINDOWS) && cfinde(;vAT_QUICKFIX)
63 #define jatc man *;ng loclist = ;f(";"jication ;;st");
NORMAL > src/buffer.c+1] input jump target
target key: [ ] 0% h 44: 1
tinge@tinge-work-server:~/android ~*1.android 2:bash 3:*vim 4:pycolor 5:bash 6:[tmux]
>2014-05-27[23:56]
```

nerdtree

1. Add the following mapping in `~/.vimrc`

```
:noremap <leader>nf :NERDTreeFind<CR>
```

Command	Description
\nf	find currently edited buffer in the NERDTree Window



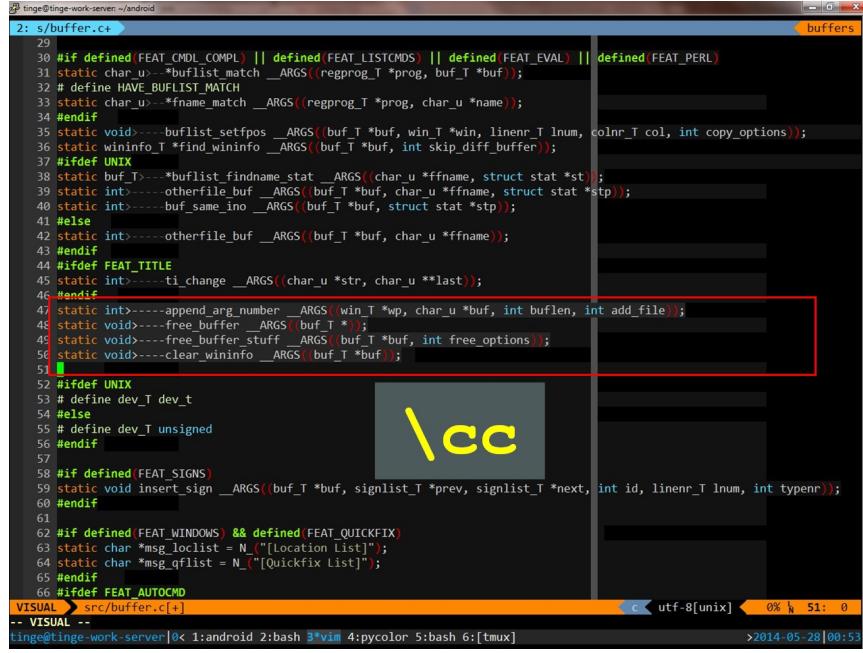
The screenshot shows a terminal session on a Linux system. The Vim editor is running in the foreground, displaying the source code for 'src/buffer.c'. A modal window titled 'Locate file in Tree' is overlaid on the Vim window, containing the command '\nf'. The Vim status bar at the bottom right indicates the date and time: '2014-05-28 00:38'. The terminal prompt shows the user is in a tmux session.

nerdcommenter

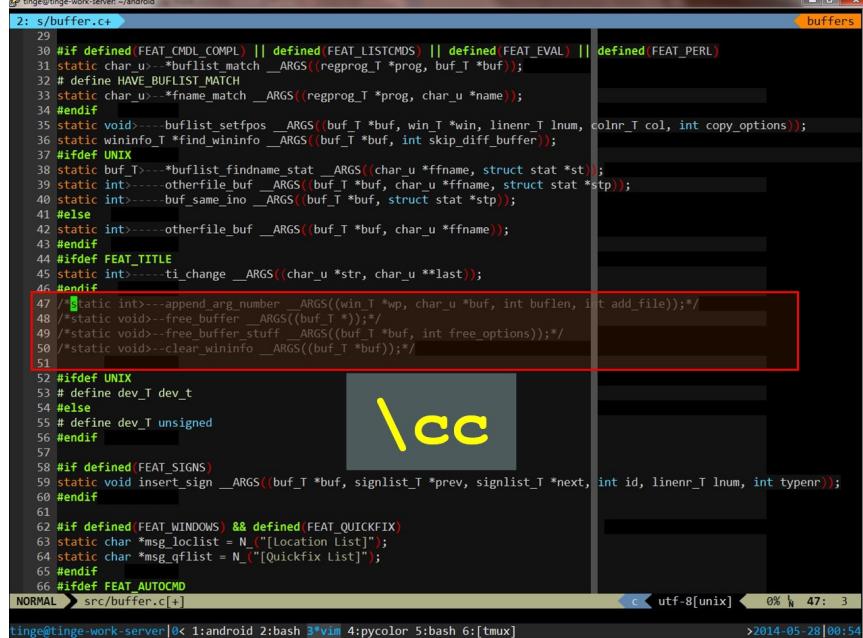
1. <https://github.com/scrooloose/nerdcommenter>
2. plugin to help comment/uncomment select code block

Command	Description

\cc	Comment out the current line or text selected in visual mode
\cu	Uncomments the selected line(s).
\cA	Adds comment delimiters to the end of line and goes into insert mode between them.
\c\$	Comments the current line from the cursor to the end of line.



The screenshot shows a Vim session with two windows. The left window displays a C source file named 'src/buffer.c'. A red box highlights a selection of code lines 47 through 50. The right window shows the character '\cc' in a large font. Below the windows, the status bar indicates 'utf-8[unix] 0% 51: 0'.



The second screenshot shows the same Vim session after the \cc command has been run. The lines highlighted in the first screenshot are now preceded by a double slash (//), indicating they are commented out. The right window still shows '\cc'. The status bar indicates 'utf-8[unix] 0% 47: 3'.

DirDiff

1. <https://github.com/vim-scripts/DirDiff.vim>
2. plugins to help compare diff between two directory
3. vim . -c ":DirDiff DIRECTORY_A DIRECTORY_B"

Command	Description
s	Synchronize the current diff. You can also select a range (through visual) and press 's' to synchronize differences across a range.

\dj	Diff next: (think j for down)
\dk	Diff previous: (think k for up)

```

3: /f/1
[9:vim ~/darkblue.vim]*[10:vim ~/darkblue.vim]*
buffers
MinibufExplorer
1 " Vim color file
2 .. Maintainer:--Bohdan Vlasuk <bohdan@stu.edu.ua>
3 " Last Change:> 2003 Jul 18
4
5 " darkblue -- for those who prefer dark background
6 " [note: looks bit uglier with some terminal palettes,
7 " but is fine on default linux console palette.]
8
9 set bg=dark
+ 10 +-- 19 lines: hi clear
29 hi WildMenu>>---- guifg=yellow guibg=black ctermfg=yell
30 hi ModeMsg>>---- guifg=#22cce2>>---- ctermfg=lightblue
31 hi MoreMsg>>---- ctermfg=darkgreen>>---- ctermfg=darkgreen
32 hi Question>>---- guifg=green gui=None ctermfg=green ct
33 hi NonText>>---- guifg=#0030ff>>---- ctermfg=darkblue
34
35 hi Statusline>>---- guifg=blue guibg=darkgray gui=None>>-
36 hi StatuslineNC>>---- guifg=black guibg=darkgray gui=None>
37 hi VertSplit>>---- guifg=black guibg=darkgray gui=None>>-
38
39 hi Folded>>---- guifg=#000080 guibg=#000040>>---- ctermfg=d
~/vim/runtime/colors/darkblue.vim vim vim 1% 1: 1 </>vim64/vim/runtime/colors/darkblue.vim vim vim 1% 1: 1
9 Files [A]/pixmaps/tb_print.xpm and [B]/pixmaps/tb_print.xpm differ
10 Files [A]/README.txt and [B]/README.txt differ
11 Files [A]/README_src.txt and [B]/README_src.txt differ
12 Files [A]/README_unix.txt and [B]/README_unix.txt differ
13 Files [A]/runtime/bugreport.vim and [B]/runtime/bugreport.vim differ
14 Files [A]/runtime/colors/blue.vim and [B]/runtime/colors/blue.vim differ
15 Files [A]/runtime/colors/darkblue.vim and [B]/runtime/colors/darkblue.vim differ
16 Files [A]/runtime/colors/delekv.vim and [B]/runtime/colors/delekv.vim differ
17 Files [A]/runtime/colors/desert.vim and [B]/runtime/colors/desert.vim differ
18 Files [A]/runtime/colors/evening.vim and [B]/runtime/colors/evening.vim differ
19 Files [A]/runtime/colors/koehler.vim and [B]/runtime/colors/koehler.vim differ
20 Files [A]/runtime/colors/morning.vim and [B]/runtime/colors/morning.vim differ
21 Files [A]/runtime/colors/README.txt and [B]/runtime/colors/README.txt differ
22 Files [A]/runtime/colors/ron.vim and [B]/runtime/colors/ron.vim differ
NORMAL > /tmp/v5izlw0/1[-] utf-8[unix] 0% 15: 1
tinge@tinge-work-server|@ 1:android 2:bash 3*vim 4:pycolor 5:bash 6:[tmux] >2014-05-28|01:09

```

EasyGrep

- <https://github.com/vim-scripts/EasyGrep>
- based on "vimgrep" command to find tag under cursor easily

Command	Description
<Leader>vv (double v)	Grep for the word under the cursor, match all occurrences

```

2: s/buffer.c
5861 /* restore curwin/curbuf and a few other things */
5862 aucmd_restbuf(&aco);
5863
5864 if (curbuf != newbuf)>----/* safety check */
5865 >-----wipe_buffer(newbuf, FALSE);
5866
5867 return differ;
5868 )
5869
5870 /* wipe out a buffer and decrement the last buffer number if it was used for
5871 * this buffer. Call this 'wipe' a temp buffer that does not contain any
5872 * marks.
5873 */
5874 void
5875 wipe_buffer(buf, aucmd)
5876 buf_t *buf;
5877 int>-----aucmd UNUSED;--- /* When TRUE trigger autocmds. */
5878 {
5879     if (buf->bnum == top_file_num - 1)
5880         --top_file_num;
5881
5882 #ifdef FEAT_AUTOCMD
5883     if (aucmd >>>----/* Don't trigger BufDelete autocmds here. */
5884 >-----block_autocmds();
5885
5886 #endif
5887     close_buffer(NULL, buf, DOBUF_WIPE, FALSE);
5888
NORMAL > src/buffer.c
2 tags[25991 col 1] wipe_buffer.../src/buffer.c/^wipe_buffer.buf, aucmd$/;">
3 src/buffer.c[5865 col 2] wipe_buffer.newbuf, FALSE;
4 src/buffer.c[5865 col 2] wipe_buffer.newbuf, FALSE;
5 src/buffer.c[5870 col 1] wipe_buffer.buf, aucmd
6 src/ex_cmds.c[5870 col 6] wipe_buffer.buf, TRUE;
7 src/fileio.c[7250 col 6] wipe_buffer.savebuf;
8 src/quickfix.c[3675 col 6] wipe_buffer.newbuf_to_wipe, FALSE;
9 src/quickfix.c[3717 col 2] wipe_buffer.buf, FALSE;
[Quickfix List][...]
tinge@tinge-work-server|@ 1:android 2:bash 3*vim 4:pycolor 5:bash 6:[tmux] >2014-05-28|01:21

```

gundo

- <https://github.com/sjl/gundo.vim>

2. Gundo.vim is Vim plugin to visualize your Vim undo tree.
3. Add the following mapping in `~/.vimrc`

```
nnoremap <F6> :GundoToggle<CR>
```

Command	Description
F6	Toggle undo window

A screenshot of a Vim session showing the Gundo plugin. The main buffer contains code for the `src/buffer.c` file. A modal window titled "undo history" is open, showing a list of undo steps:

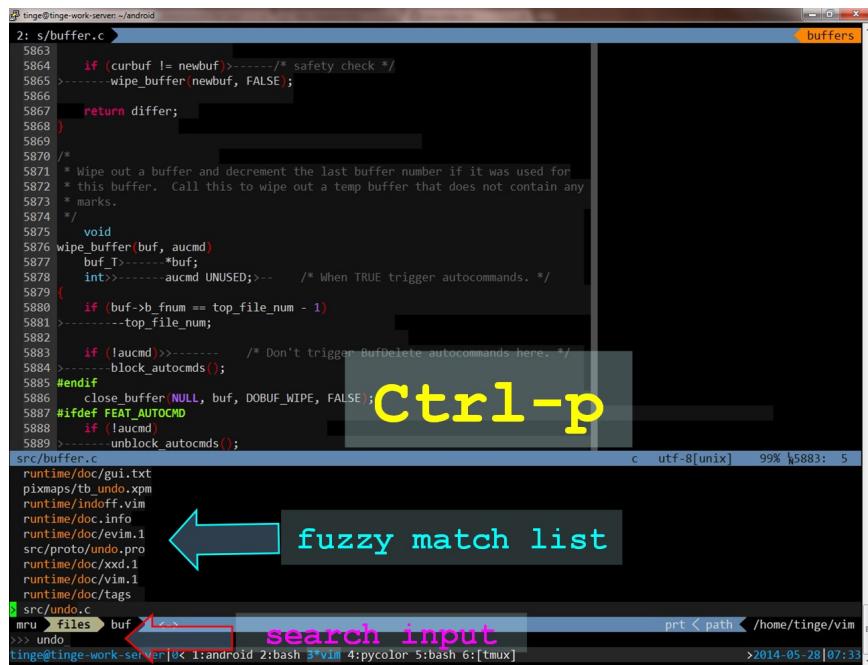
- [3] 20 seconds ago
- [2] 21 seconds ago
- [1] 24 seconds ago
- [0] Original

The status bar at the bottom shows keybindings: "history diff" and "F6".

ctrlp

1. <https://github.com/kien/ctrlp.vim>
2. Full path fuzzy file, buffer, mru, tag, ... finder for Vim.

Command	Description
<Ctrl-p>	invoke CtrlP in find file mode
<Ctrl-f> <Ctrl-b>	cycle between three modes. (Find files, Find Buffer, Find MRU)
<Ctrl-d>	switch to filename only search instead of full path.
<Ctrl-j> <Ctrl-k> narrow keys	to navigate the result list



vim-bookmarks

1. <https://github.com/MattesGroeger/vim-bookmarks>
2. bookmarks which works independently from vim marks
3. add the following config into .vimrc

```
let g:bookmark_sign = '♥'
let g:bookmark_highlight_lines = 1
highlight BookmarkSign ctermbg=166 ctermfg=89
highlight BookmarkAnnotationSign ctermbg=30 ctermfg=89
highlight BookmarkLine ctermbg=166 ctermfg=NONE
highlight BookmarkAnnotationLine ctermbg=30 ctermfg=NONE
```

Command	Description
mm	Add/remove bookmark at current line
mi	Add/edit/remove annotation at current line
ma	Show all bookmarks
mn	Jump to next bookmark in buffer
mp	Jump to previous bookmark in buffer
mc	Clear bookmarks in current buffer only
mx	Clear bookmarks in all buffers

A screenshot of a Vim session on a Linux terminal. The code being edited is from `src/buffer.c`. Several annotations are overlaid on the code:

- An orange arrow points to the word "bookmark" above the line `if (curbuf != newbuf) { /* safety check */`.
- An orange arrow points to the line `if (!aucmd) >----- Annotation: hello` with the text "annotation add by mi".
- An orange arrow points to the line `if (buf->b_fnum == top_file_num - 1)` with the text "mark listshown by ma".

The status bar at the bottom shows the file name `src/buffer.c`, line number `5865`, and the date `>2014-05-28|08:08`.

vim-multiple-cursors

1. <https://github.com/terryma/vim-multiple-cursors>
2. Porting of multiple selection feature from Sublime

Command	Description
<code><Ctrl-n></code>	select next key
<code><Ctrl-p></code>	in Visual mode will remove the current virtual cursor and go back to the previous virtual cursor location.
<code><Ctrl-x></code>	in Visual mode will remove the current virtual cursor and skip to the next virtual cursor location.

A screenshot of Vim showing a visual block selection. The text "Select 3 line in V mode" is overlaid on the screen. The status bar shows the file name `file.txt`, line number `144`, and the date `>2014-05-28|08:25`.

A screenshot of Vim showing the result of pressing `Ctrl-n`. The text "Ctrl-n" is overlaid on the screen. The status bar shows the file name `file.txt`, line number `142`, and the date `>2014-05-28|08:25`.

A screenshot of Vim showing the cursor moved to a target location. The text "move cursor to target location" is overlaid on the screen. The status bar shows the file name `file.txt`, line number `143`, and the date `>2014-05-28|08:26`.

A screenshot of Vim showing insert mode across multiple lines. The text "insert "demo" in all lines" is overlaid on the screen. The status bar shows the file name `file.txt`, line number `143`, and the date `>2014-05-28|08:27`.

YouCompleteMe

1. <https://github.com/Valloric/YouCompleteMe>
2. Vim Killer plugin for code completion engine
3. C/S architecture, make the completion suggestion very fast
4. Configure is a little complicated
5. Make sure Vim 7.3.584 with python2 support. some time need rebuild vim to the latest version(7.4)

1. <https://github.com/Valloric/YouCompleteMe/wiki/Building-Vim-from-source>

A screenshot of a terminal window titled "buffers" showing a Python script named "vim256_compare.py". The script contains code for generating color palettes. A red arrow points from the text "show suggestion very quickly" to the completion interface in the background. The status bar at the bottom shows "User defined completion" and the date "2014-05-28 09:00".

```

1: vim256_compare.py+
idx = color_cell.idx
mostlikecolor_str = org_color_str
elif local_distance < distance :
    distance = local_distance
    idx = color_cell.idx
    mostlikecolor_str = org_color_str
return idx, mostlikecolor_str

def main():
if len(sys.argv) < 4:
    _printUsage()
else:
    #xterm 256 colorset = xterm_256_gen.xterm256_colorset()
    xterm_256_colorset = xterm_256_gen
    input_xls_file = sys.argv[1]
    output_xls_file = sys.argv[2]
    output_svg_file = sys.argv[3]
    color_set = reader.load_model(input_xls_file)
    added_grp_list = []
    main
    model
    import: import model
    function: xterm_256_gen.main
    function: xterm_256_gen.xterm256_colorset
    printUsage
    writer
    import: import sys
    import: import model_writer as writer
for color_grp in color_set.grp:
    added_grp = model.ColorGrp(color_grp.name + '_vim')
    added_grp.width = color_grp.width
    added_grp.height = color_grp.height
    for color_row in color_grp.row:
        added_row = model.ColorRow(color_row.row)
        added_row.width = color_row.width
        added_row.height = color_row.height
        for color_cell in color_row.cell:
            added_cell = model.ColorCell(color_cell.col, color_cell.idx,
                                         color_cell.fill, color_cell.width, color_cell.height,
                                         color_cell.text, color_cell.textColor)
            added_cell.idx, added_cell.fill = findMostLikeColor(added_cell.fill, xterm_256_colorset)
INSERT >>> vim256_compare.py[4] -- User defined completion ("U'N'P") Back at original
tinge@tinge-work-server:~> 1:android 2:bash 3:>vim 4:pycolor 5:bash 6:[tmux]

```

syntastic

1. <https://github.com/scrooloose/syntastic>
2. Syntastic is a syntax checking plugin for Vim that runs files through external syntax checkers and displays any resulting errors to the user
3. following picture is from his github

A screenshot of GVIM showing a C++ file "main.cpp". The statusline indicates "Syntax: line:4 (3)". Several error markers are visible: a yellow sign at the top left, a red error balloon near the top right, a red location list in the statusline, a red command window at the bottom, and a red statusline flag. Red arrows point from numbered labels to these specific features.

matrix

1. <https://github.com/vim-scripts/matrix.vim-- Yang>
2. screen save for funny

useful command

buffer operation

No	Command	description	example
1	:buffers		show all buffers (the same as :ls)
2	:buffer 2(b2)		show buffer 2
3	:bf		bfirst: show first buffer
4	:bn		bnext: show next buffer
5	:bp		bprevious: show previous buffer
6	:bd 3		bdelete: delete buffer
7	:bufdo[!] {cmd}		Execute {cmd} in each buffer in the buffer list
8	:sb[n]		split window to show buffer [N] in horizontal mode
9	:e#		edit the last edit buffer

Windows command

1. Add the following definition in `~/.vimrc`

```

noremap <C-J>      <C-W>j
noremap <C-K>      <C-W>k
noremap <C-H>      <C-W>h
noremap <C-L>      <C-W>l

noremap <C-Down>   <C-W>j
noremap <C-Up>     <C-W>k
noremap <C-Left>   <C-W>h
noremap <C-Right>  <C-W>l

:set mouse=a
:set ttymouse=xterm2

```

Modify window boundary

Command	Description
<C-W> +, <C-W> -	adjust window row number
<C-W> >, <C-W> <	adjust window column number
20<C-W>[+ =]	adjust 20 unit one time
<C-W>	adjust window to maximum of column
<C-W> _	adjust window to maximum of row
'x' in Tlist window	toggle TList window full screen display

add map in .vimrc as the following

Navigate between windows

Command	Description
<C-J> <C-Down> <C-W> j	switch to the window below
<C-K> <C-UP> <C-W> k	switch to the window above
<C-H> <C-Left> <C-W> h	switch to the window left
<C-L> <C-Right> <C-W> l	switch to the window right

<C-W> <C-W>

switch between windows

locate currently edited file in NERDTree window

1. :NERDTreeFind ->Find the current file in the tree.

jump in a file

Command	Description
A	Cursor jump to end of line and enter 'insert' mode
CRTL+^	line head(first character of line)
CRTL+\$	line end
0(zero)	column zero of line
CRTL+o	go back to previous edit point
CRTL+i	go forward to next edit point
G	file end
gg	file head
CTRL+]	jump to define <within ctags and cscope>
CTRL+t	back to pre-jump tag<within ctags and cscope>
\w	show beginning of word forward and then select to jump <within easymotion plugin>
\b	show beginning of word backward and then select to jump <within easymotion plugin>
o(not zero)	enter a new line below current line and enter insert mode
O(not zero, big cap)	enter a new line above current line and enter insert mode
%	map 1) #if-#else-#endif 2){}
[#	go to end of #if-#else-#endif
]#	go to begin of #if-#else-#endif
[[jump to front of function body or front of previous function body
[]	jump to end of function body
]]	jump to the front of next function body

search in vim

command	usage	description
vimgrep		
	:vimgrep /{pattern}/{g}{j} {file} ...	
	:vimgrep /MarketId/ **/*.c **/*.h	and then :cw open quickfix window to watch the search list
grep		
	::grep -n --exclude=*.h -R MarketId . -n	and then :cw open quickfix window to watch grep result

quickfix

:cc	show the detail error info
:cp	jump to previous item
:cn	jump to next item
:cl	list all item
:cw	if any items, open quickfix window
:copen	open quickfix window, accept param for the height, for example: :copen 10
:cclose	close quickfix window
:col	previous old items list
:cnew	next new items list

EasyGrep

<Leader>vv	Grep for the word under the cursor, match all occurrences, like 'gstar'
<leader>vV	Grep for the word under the cursor, match whole word, like 'star'
<Leader>va	like vv, but add to existing list
<Leader>vA	like vV, but add to existing list
<Leader>vr	Perform a global search search on the word under the cursor and prompt for a pattern with which to replace it
<Leader>vo	Select the files to search in and set grep options

map for cscope

1. :nmap <C-f>s :cs find s <C-R><C-W><CR>
2. :nmap <C-f>g :cs find g <C-R><C-W><CR>
3. :nmap <C-f>c :cs find c <C-R><C-W><CR>
4. :nmap <C-f>t :cs find t <C-R><C-W><CR>

vim (last edited 2014-05-28 08:25:47 by tinge Zhang)