[**经典vim插件功能说明、安装方法和使用方法介绍（已更新）**](http://blog.csdn.net/tge7618291/article/details/4216977)

分类： [linux](http://blog.csdn.net/tge7618291/article/category/536024) 2009-05-26 13:11 34838人阅读 [评论](http://blog.csdn.net/tge7618291/article/details/4216977#comments)(16) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/tge7618291/article/details/4216977#report)

  1 #=============================================================================  
  2  转载请注明出处: http://blog.csdn.net/tge7618291

[http://nuoerlz.35free.net](http://nuoerlz.35free.net/index.html)

  8 #=============================================================================  
  9 **1. 查看 key 相关信息说明的命令  :help keycodes**  
 10   
 11 #=============================================================================  
 12 **2. ctags**  
 13 (1). 帮助手册查看  
 14     :help usr\_29  
 15   
 16 (2). 功能  
 17     ctags的功能, 只要在unix/linux环境编程过的人都知道它的历害! 我在这也作个小  
 18     小介绍吧ndow=0: 对浏览代码非常的方便, 可以在函数, 变量之间跳来跳去等等.(注  
 19             意: 我这里说到的仅是小小的一部分!).  
 20   
 21 (3). 安装  
 22     首先, 下载ctags安装包, 然后解压并安装, 命令如下:  
 23     $ tar -xzvf ctags-5.6.tar.gz  
 24     $ cd ctags-5.6  
 25     $ make  
 26     # make install   // 需要root权限  
 27         或:  
 28         $ tar -xzvf ctags-5.7.tar.gz  
 29         $ cd ctags-5.7  
 30         $ ./configure  
 31         # make  
 32         # make install  
 33   
 34   
 35 (4). 使用方法  
 36     然后去你的源码目录, 如果你的源码是多层的目录, 就去最上层的目录, 在该目录下  
 37     运行命令: ctags -R  
 38     我现在以 vim71 的源码目录做演示  
 39     $ cd /home/nuoerll/vim71  
 40     $ ctags -R  
 41       
 42     此时在/home/nuoerll/vim71目录下会生成一个 tags 文件, 现在用vim打开  
 43     /home/nuoerll/vim71/src/main.c  
 44     $ vim /home/nuoerll/vim71/src/main.c  
 45       
 46     再在vim中运行命令:  
 47     :set tags=/home/nuoerll/vim71/tags  
 48     该命令将tags文件加入到vim中来, 你也可以将这句话放到~/.vimrc中去, 如果你经  
 49     常在这个工程编程的话.  
 50     对于经常在不同工程间编程, 可以在.vimrc中设置:  
 51         set tags=tags;    // ; 不能没有  
 52         set autochdir  
 53   
 54 (5). 使用例子  
 55     把光标定位到某一函数名上, 按下 Ctar + ], vim就可以自动切换到该函数定义处!  
 56     要返回只需要按下Ctrl + t .  
 57     更多用法, 在vim命令模式输入 :help usr\_29 查看即可.  
 58   
 59   
 60 #=============================================================================  
 61 **3. TagList 插件**  
 62 (1). 帮助手册查看  
 63     :help taglist.txt  
 64   
 65 (2). 功能  
 66     高效地浏览源码, 其功能就像vc中的workpace, 那里面列出了当前文件中的所有宏,  
 67     全局变量, 函数名等.  
 68   
 69 (3). 安装  
 70     下载taglist压缩包, 然后把解压的两个文件taglist.vim 和 taglist.txt 分别放到  
 71     $HOME/.vim/plugin 和 $HOME/.vim/doc 目录中.  
 72   
 73 (4). 使用方法  
 74     首先请先在你的~/.vimrc文件中添加下面语句:  
 75         let Tlist\_Ctags\_Cmd='**/bin/ctags**'   // 若在windows中应写成: let/  
 76                             Tlist\_Ctags\_Cmd='**ctags.exe**'  
 77         let Tlist\_Show\_One\_File=1  
 78         let Tlist\_OnlyWindow=1  
 79         let Tlist\_Use\_Right\_Window=0  
 80         let Tlist\_Sort\_Type='**name**'  
 81         let Tlist\_Exit\_OnlyWindow=1  
 82         let Tlist\_Show\_Menu=1  
 83         let Tlist\_Max\_Submenu\_Items=10  
 84         let Tlist\_Max\_Tag\_length=20  
 85         let Tlist\_Use\_SingleClick=0  
 86         let Tlist\_Auto\_Open=0  
 87         let Tlist\_Close\_On\_Select=0  
 88         let Tlist\_File\_Fold\_Auto\_Close=1  
 89         let Tlist\_GainFocus\_On\_ToggleOpen=0  
 90         let Tlist\_Process\_File\_Always=1  
 91         let Tlist\_WinHeight=10  
 92         let Tlist\_WinWidth=18  
 93         let Tlist\_Use\_Horiz\_Window=0  
 94   
 95       
 96     此时用vim打开一个c源文件试试:  
 97     $ vim ~/vim/src/main.c  
 98       
 99     进入vim后用下面的命令打开taglist窗口.  
100     :Tlist  
101   
102     为了更方便地使用, 可以在.vimrc文件中加入:  
103         map <silent> <leader>tl :TlistToggle<CR>  
104     这样就可以用 "**,tl**" 命令进行taglist窗口的打开和关闭之间方便切换了. // 这里  
105     的"**,**"是我.vimrc设置的leader, 你也可以设置成别的, 在.vimrc中修改即可, 如我  
106     的: let mapleader="**,**"  
107           
108       
109 #=============================================================================  
110 **4. WinManager 插件**  
111 (1). 帮助手册查看  
112     :help winmanager  
113   
114 (2). 功能  
115     管理各个窗口, 或者说整合各个窗口.  
116   
117 (3). 安装  
118     下载WinManager.zip压缩包, 解压后把\*.vim文件放到 $HOME/.vim/plugin 目录中,  
119     把\*.txt文件放到 $HOME/.vim/doc 目录中.  
120   
121 (4). 使用方法  
122     在.vimrc中加入如下设置:  
123     let g:winManagerWindowLayout='**FileExplorer|BufExplorer**'  // 这里可以设置/  
124     为多个窗口, 如'**FileExplorer|BufExplorer|TagList**'  
125     let g:persistentBehaviour=0             // 只剩一个窗口时, 退出vim.  
126     let g:winManagerWidth=20  
127     let g:defaultExplorer=1  
128     nmap <silent> <leader>fir :FirstExplorerWindow<cr>  
129     nmap <silent> <leader>bot :BottomExplorerWindow<cr>  
130     nmap <silent> <leader>wm :WMToggle<cr>  
131   
132 (5). 使用例子  
133     在终端输入vim启动vim:  
134     $vim  
135     在正常模式下, 敲入 "**,wm**" 即可看到, vim的左侧新建了两个窗口:FileExplorer和  
136     BufExplorer, 这样我们即可在FileExplorer窗口很方便地对目录进行查看, 编辑等  
137     操作; 在BufExplorer窗口中查看当前vim已经打开那些文件.  
138   
139   
140 #=============================================================================  
141 **5. cscope**  
142 (1). 帮助手册查看  
143     :help if\_cscop.txt  
144   
145 (2). 功能  
146     用Cscope自己的话说 - "**你可以把它当做是超过频的ctags**", 其功能和强大程度可见  
147     一斑吧, 关于它的介绍我就不详细说了, 如果你安装好了前文介绍的帮助手册.  
148   
149 (3). 安装  
150     如果是在linux环境中, cscope一般都会随系统一起安装了; 在windows环境中, 则需  
151     要下载windows版本的(cscope.exe), 然后把它放到path环境变量所设  
152     置的目录中(如: C:/Program Files/Vim/vim72).  
153   
154 (4). 使用方法  
155     在.vimrc中增加如下设置, 就可以利用相应的快捷键进行不同的查找了.  
156         if has("cscope")  
157             set cscopetag   "**使支持用 Ctrl+]  和 Ctrl+t 快捷键在代码间跳来跳去**  
158 "**check cscope for definition of a symbol before checking ctags:**  
159 " set to 1 if you want the reverse search order.  
160             set csto=1  
161   
162             "**add any cscope database in current directory**  
163 **if filereadable(**"cscope.out")  
164                 cs add cscope.out  
165             "**else add the database pointed to by environment variable**  
166 **elseif $CSCOPE\_DB !=**""  
167                 cs add $CSCOPE\_DB  
168             endif  
169   
170             " show msg when any other cscope db added  
171             set cscopeverbose  
172   
173             nmap <C-/>s :cs find s <C-R>=expand("<cword>")<CR><CR>  
174             nmap <C-/>g :cs find g <C-R>=expand("<cword>")<CR><CR>  
175             nmap <C-/>c :cs find c <C-R>=expand("<cword>")<CR><CR>  
176             nmap <C-/>t :cs find t <C-R>=expand("<cword>")<CR><CR>  
177             nmap <C-/>e :cs find e <C-R>=expand("<cword>")<CR><CR>  
178             nmap <C-/>f :cs find f <C-R>=expand("<cfile>")<CR><CR>  
179             nmap <C-/>i :cs find i ^<C-R>=expand("<cfile>")<CR>$<CR>  
180             nmap <C-/>d :cs find d <C-R>=expand("<cword>")<CR><CR>  
181         endif  
182   
183 (5). 使用例子  
184     首先进入源码目录, 在linux终端中输入以下命令以创建cscope数据库:  
185         $ find ~/work/..Project/ -name "**\*.h**" -o -name "**\*.cpp**" > cscope.files  
186         $ cscope -bkq -i cscope.files  
187     如果是在windows环境中, 则换成如下命令:  
188         dir /s /b \*.cpp \*.h > cscope.files  
189         cscope -b -q -k -i cscope.files  
190   
191     然后, 用vim打开一个源文件(如: vim main.cpp),  
192     打开后, 第一件事就是导入cscope数据库了:  
193         :cs add /home/yourname/workpace/cscope.out /home/yourname/workpace  
194   
195     cscope数据库导入成功后, 就可以利用"**步骤(4)**"中定义的快捷键进行相关的查找,  
196     浏览等操作了(当然也可以直接利用相关命令, 嘿嘿).  
197   
198 #=============================================================================  
199 **5. c.vim 插件**  
200 (1). 帮助手册查看  
201     help csupport  
202   
203 (2). 功能  
204     C/C++-IDE for Vim/gVim. 简单的说, 就是如果安装配置好后, vim/gvim就是一个  
205     c/c++编程的一个IDE, 其功能堪比windows里常用的vc.  
206   
207 (3). 安装  
208     下载cvim.zip压缩包后, 把压缩包copy到 $HOME/.vim 目录(windows下, copy 到  
209             C:/Program Files/Vim/vimfiles)下解压, 即可.  
210         unzip cvim.zip    // 解压命令  
211           
212 (4). 使用方法  
213     安装好后就可以直接用了, 具体例子看"**使用例子**".  
214       
215 (5). 使用例子  
216     在终端用vim打开一个c文件:  
217         $vim hello.c  
218     进入vim, 敲入 "**/im**" 即可发现一个main函数框架就这样快捷简单完美地写出.      
219   
220     我比效常用的的操作有(第一列为命令, 第二列为说明, 第三列为该命令所支持的模  
221             式(n:普通模式, v:可视模式, i:插入模式):  
222           -- Help -----------------------------------------------  
223           /hm       show manual for word under the cursor(n,i)  
224           /hp       show plugin help                      (n,i)  
225   
226           -- Comments -------------------------------------------  
227           /cl       end-of-line comment                 (n,v,i)  
228           /cj       adjust end-of-line comment(s)      (n,v,i)  
229           /cs       set end-of-line comment column      (n)  
230           /c\*       code -> comment /\* \*/               (n,v)  
231           /cc       code -> comment //                  (n,v)  
232           /co       comment -> code                     (n,v)  
233           /cfr      frame comment                       (n,i)  
234           /cfu      function comment                    (n,i)  
235           /cme      method description                  (n,i)  
236           /ccl      class description                  (n,i)  
237           /cd       date                                (n,v,i)  
238           /ct       date /& time                        (n,v,i)  
239   
240           -- Statements ------------------------------------------  
241           /sd       do { } while                        (n,v,i)  
242           /sfo      for { }                            (n,v,i)  
243           /sife     if { } else { }                    (n,v,i)  
244           /se       else { }                            (n,v,i)  
245           /swh      while { }                          (n,v,i)  
246           /ss       switch                              (n,v,i)  
247   
248           -- Preprocessor -----------------------------------------  
249           /p<       #include <>                        (n,i)  
250           /p''      #include ""                        (n,i)  
251           /pd       #define                            (n,i)  
252           /pu       #undef                              (n,i)  
253           /pie      #if  #else #endif                  (n,v,i)  
254           /pid      #ifdef #else #endif                (n,v,i)  
255           /pin      #ifndef #else #endif                (n,v,i)  
256           /pind     #ifndef #def #endif                (n,v,i)  
257           /pi0      #if 0 #endif                        (n,v,i)  
258           /pr0      remove #if 0 #endif                (n,i)  
259           /pe       #error                              (n,i)  
260           /pl       #line                              (n,i)  
261           /pp       #pragma                            (n,i)  
262   
263           -- Idioms ------------------------------------------------  
264           /if       function                            (n,v,i)  
265           /isf      static function                    (n,v,i)  
266           /im       main()                              (n,v,i)  
267           /i0       for( x=0; x<n; x+=1 )              (n,v,i)  
268           /in       for( x=n-1; x>=0; x-=1 )            (n,v,i)  
269           /ie       enum   + typedef                    (n,i)  
270           /is       struct + typedef                    (n,i)  
271           /iu       union  + typedef                    (n,i)  
272           /ip       printf()                            (n,i)  
273           /isc      scanf()                            (n,i)  
274           /ica      p=calloc()                          (n,i)  
275           /ima      p=malloc()                          (n,i)  
276           /isi      sizeof()                            (n,v,i)  
277           /ias      assert()                            (n,v)  
278           /ii       open input file                    (n,i)  
279           /io       open output file                    (n,i)  
280   
281           -- Snippets ------------------------------------------------  
282           /nr       read code snippet                  (n,i)  
283           /nw       write code snippet                  (n,v,i)  
284           /ne       edit code snippet                  (n,i)  
285           /np       pick up prototype                  (n,v,i)  
286           /ni       insert prototype(s)                (n,i)  
287           /nc       clear  prototype(s)                (n,i)  
288           /ns       show   prototype(s)                (n,i)  
289           /ntl      edit local templates                (n,i)  
290           /ntg      edit global templates              (n,i)  
291           /ntr      rebuild templates                  (n,i)  
292   
293           -- C++ ----------------------------------------------------  
294           /+co      cout  <<  << endl;                  (n,i)  
295           /+c       class                              (n,i)  
296           /+cn      class (using new)                   (n,i)  
297           /+ci      class implementation                (n,i)  
298           /+cni     class (using new) implementation    (n,i)  
299           /+mi      method implementation              (n,i)  
300           /+ai      accessor implementation            (n,i)  
301   
302           /+tc      template class                      (n,i)  
303           /+tcn     template class (using new)          (n,i)  
304           /+tci     template class implementation      (n,i)  
305           /+tcni    template class (using new) impl.    (n,i)  
306           /+tmi     template method implementation      (n,i)  
307           /+tai     template accessor implementation    (n,i)  
308   
309           /+tf      template function                  (n,i)  
310           /+ec      error class                        (n,i)  
311           /+tr      try ... catch                      (n,v,i)  
312           /+ca      catch                              (n,v,i)  
313           /+c.      catch(...)                          (n,v,i)  
314   
315           -- Run ---------------------------------------------------  
316           /rc       save and compile                    (n,i)  
317           /rl       link                                (n,i)  
318           /rr       run                                (n,i)  
319           /ra       set comand line arguments          (n,i)  
320           /rm       run make                            (n,i)  
321           /rg       cmd. line arg. for make            (n,i)  
322           /rp       run splint                          (n,i)  
323           /ri       cmd. line arg. for splint          (n,i)  
324           /rk       run CodeCheck (TM)                  (n,i)  
325           /re       cmd. line arg. for CodeCheck(TM)   (n,i)  
326           /rd       run indent                          (n,v,i)  
327           /rh       hardcopy buffer                    (n,v,i)  
328           /rs       show plugin settings                (n,i)  
329           /rx       set xterm size                      (n, only Linux/UNIX & GUI)  
330           /ro       change output destination          (n,i)  
331                   
332     关于此插件的更多功能和各种说明, 请查看帮助手册, help csupport.  
333   
334               
335 #=============================================================================  
336 **6. omnicppcoplete 插件**  
337 (1). 帮助手册查看  
338     :help omnicppcoplete  
339 (2). 功能  
340     实现像vc那样的代码自动补全功能, 比如 this-><Ctrl+X><Ctrl+O> 后, 将出现一个  
341     提示框, 其中包含了this指针所有可以接收的函数或数据成员等.  
342 (3). 安装  
343     把下载下来的 omnicppcoplete-0.41.zip 压缩包copy到 $HOME/.vim/(windows 复  
344             制到 C:/Program Files/Vim/vimfiles ), 然后解压, 即可.  
345 (4). 使用方法  
346     在.vimrc中添加以下两条语句:  
347         set nocp   "**不兼容vi**  
348 **filetype plugin on**"开启文件类型识别功能  
349     进入c++源码目录, 在终端执行命令 ctags -R --c++-kinds=+p --fields=+iaS  
350     --extra=+q .  
351 (5). 使用例子  
352     编写c++代码时, 如要自动补全, 敲入 Ctrl+X Ctrl+O, 即可在出现的提示框中用  
353     Ctrl+N 选择符合要求的.  
354   
355   
356 #=============================================================================  
357 **7. a.vim插件**  
358 (1). 帮助手册查看  
359     这个插件没有帮助手册, 不过大可放心使用, 其提供的功能也不是很多, 就几条命令  
360     , 但是用起来真的是很方便.  
361 (2). 功能  
362     在 .h 和 .c/.cpp 文件中切换. (英文原句 "A few of quick commands to swtich  
363             between source files and header files quickly.")  
364 (3). 安装  
365     把下载到的a.vim插件放到 $HOME/.vim/plugin 目录下, 即可.  
366 (4). 使用方法  
367     只要在vim中输入以下命令即可完成相应的功能:  
368         :A switches to the header file corresponding to the current file being  
369             edited (or vise versa)  
370         :AS splits and switches  
371         :AV vertical splits and switches  
372         :AT new tab and switches  
373         :AN cycles through matches  
374         :IH switches to file under cursor  
375         :IHS splits and switches  
376         :IHV vertical splits and switches  
377         :IHT new tab and switches  
378         :IHN cycles through matches  
379         <Leader>ih switches to file under cursor  
380         <Leader>is switches to the alternate file of file under cursor(e.g.  
381                 on  <foo.h> switches to foo.cpp)  
382         <Leader>ihn cycles through matches  
383   
384 #=============================================================================  
385 **8. VisualMark.vim插件**  
386 (1). 帮助手册查看  
387     这个插件没有帮助手册, 不过大可放心使用, 其提供的功能也不是很多, 就几条命令  
388     , 但是用起来真的是很方便.  
389 (2). 功能  
390     高亮书签.  
391 (3). 安装  
392     把下载好的VisualMark.vim插件放到 $HOME/.vim/plugin 目录下, 即可.  
393 (4). 使用方法  
394     只要在vim中执行以下命令即可完成相应的功能:  
395          1.  For gvim, use "**Ctrl + F2**" to toggle a visual mark.  
396              For both vim and gvim, use "**mm**" to toggle a visual mark.  
397          2.  Use "**F2**" to navigate through the visual marks forward in the  
398          file.  
399          3.  Use "**Shift + F2**" to navigate backwards.  
400   
401 #=============================================================================  
402 **9. Mark.vim插件**  
403 (1). 帮助手册查看  
404      这个插件没有帮助手册, 不过大可放心使用, 其提供的功能也不是很多, 就几条命  
405      令, 但是用起来真的是很方便.  
406   
407 (2). 功能  
408      这个插件与vim中自带的'**\***'与'**#**'非常相像. 不同之处是: vim中的'**\***'与'**#**'命令只  
409      能同时高亮一个同类词组(或正则表达式的搜索结果), 而Mark.vim插件可以同时高  
410      亮多个.  
411   
412 (3). 安装  
413     把下载好的Mark.vim插件放到 $HOME/.vim/plugin 目录中, 即可.  
414   
415 (4). 使用方法  
416     /m      mark or unmark the word under (or before) the cursor  
417     /r      manually input a regular expression. 用于搜索.  
418     /n      clear this mark (i.e. the mark under the cursor), or clear all  
419             highlighted marks .  
420     /\*      把光标向前切换到当前被Mark的MarkWords中的下一个MarkWord.  
421     /#      把光标向后切换到当前被Mark的MarkWords中的上一个MarkWord.  
422     //      把光标向前切换到所有被Mark的MarkWords中的下一个MarkWord.  
423     /?      把光标向后切换到所有被Mark的MarkWords中的上一个MarkWord.  
424   
425         说明: 这些命令中的 '/' 是 vim 中的 mapleader, 你也可以设置为别的: 如,  
426         若要设置为 '**,**', 把下面这条语句加到.vimrc文件中, 即可,  
427             let mapleader="**,**"  
428   
429   
430 #=============================================================================  
431 **10. code\_complete.vim插件**  
432 (1). 帮助手册查看  
433 (2). 功能  
434      函数参数提示.  
435   
436 (3). 安装  
437      下载code\_complete.vim插件放到 C:/Program Files/Vim/vimfiles 目录中, 即可.  
438   
439 (4). 使用方法  
440      进入源码目录, 执行如下命令:  
441      ctags -R --c-kinds=+p --fields=+S .  
442   
443 (5). 使用例子  
444 "**Usage:**  
445 "           hotkey:  
446 ""<tab>"**(default value of g:completekey)**  
447 "               Do all the jobs with this key, see  
448 "**example:**  
449 "               press <**tab> after function name and (**  
450 "                 foo ( <**tab>**  
451 "               becomes:  
452 "**foo ( /<first param>`,/<second param>` )**  
453 "               press <**tab> after code template**  
454 "                 if <**tab>**  
455 "               becomes:  
456 "**if( /<...>` )**  
457 "                 **{**  
458 "                     /<**...>`**  
459 "                 }  
460   
461   
462 #=============================================================================  
463 **11. autoproto.vim**  
464 Using this script, typing ``(`` will result in (|), where | is the cursor  
465         position and the double backticks are just marking input. Typing a  
466         ``)`` will move the cursor outside the parens. This moving outside  
467 works even in nested scenarios. Typing ``if(my\_array['key`` results in  
468         if(my\_array['key|']) and ``)`` gets you if(my\_array['key'])|.  
469   
470 The paired characters are: [, (, ｛, "**, ';   //**"  
471   
472 If you like this script, you should also check out surround.vim  
473   
474   
475   
476   
477 #=============================================================================  
478 **12. pyclewn**  
479 pyclewn在unix, windows下的安装方法:  
480 <http://pyclewn.wiki.sourceforge.net/install>+  
481   
482 下载安装python  
483 <http://www.python.org/download/>  
484   
485 python补丁(pywin32-212.win32-py2.6.txt)(对pyclewn)下载安装  
486 <http://sourceforge.net/project/platformdownload.php?group_id=78018>  
487   
488 下载安装MimGW或Cywin  
489   
490 下载安装pyclewn  
491 <http://sourceforge.net/project/showfiles.php?group_id=212808>  
492   
493 (1). 帮助手册查看  
494 (2). 功能  
495 (3). 安装  
496 (4). 使用方法  
497 (5). 使用例子  
498   
499   
500 #=============================================================================  
501 **13. project.vim**  
502 (1). 帮助手册查看  
503     :help project.txt  
504   
505 (2). 功能  
506     组织管理工程, 方便于浏览, 打开, 查找文件等.  
507   
508 (3). 安装  
509     下载project.vim压缩包(如: project.gz), 然后把解压的两个文件project.vim 和  
510     project.txt 分别放到 $HOME/.vim/plugin 和 $HOME/.vim/doc 目录中.  
511   
512 (4). 使用方法  
513     在.vimrc中加入以下设置:  
514         // 切换打开和关闭project窗口  
515         nmap <silent> <Leader>P <Plug>ToggleProject  
516         //插件项目窗口宽度.    默认值: 24  
517         let g:proj\_window\_width=20 //当按空格键 <space> 或者单击鼠标左键/  
518                                 <LeftMouse>时项目窗口宽度增加量,默认值:100  
519         let g:proj\_window\_increment=90  
520         let g:proj\_flags='**i**'    //当选择打开一个文件时会在命令行显示文件名和当  
521                                 前工作路径.  
522         let g:proj\_flags='**m**'    //在常规模式下开启 |CTRL-W\_o| 和  
523                                 |CTRL-W\_CTRL\_O| 映射, 使得当前缓冲区成为唯一可  
524                                 见的缓冲区, 但是项目窗口仍然可见.  
525         let g:proj\_flags='**s**'    //开启语法高亮.  
526         let g:proj\_flags='**t**'    //用按 <space> 进行窗口加宽.  
527         let g:proj\_flags='**c**'    //设置后, 在项目窗口中打开文件后会自动关闭项目  
528                                 窗口.  
529         //let g:proj\_flags='**F**'   //显示浮动项目窗口. 关闭窗口的自动调整大小和窗  
530                                 口替换.  
531         let g:proj\_flags='**L**'    //自动根据CD设置切换目录.  
532         //let g:proj\_flags='**n**'    //显示行号.  
533         let g:proj\_flags='**S**'    //启用排序.  
534         let g:proj\_flags='**T**'    //子项目的折叠在更新时会紧跟在当前折叠下方显示(  
535                                 而不是其底部).  
536         let g:proj\_flags='**v**'    //设置后将, 按 /G 搜索时用 :vimgrep 取代 :grep.  
537         //let g:proj\_run1='**!p4 edit %f**'    //g:proj\_run1 ...  g:proj\_run9 用法.  
538         let g:proj\_run3='**silent !gvim %f**'  
539   
540 (5). 使用例子  
541     1. 在源码目录下建立一个工程文件: exampleProject  
542         $ gvim exampleProject  
543   
544     2. 在exampleProject文件中定入:  
545        MyProject="**E:/desktop\_item/tmp/0virtual/nehe2/LVHM/test**" CD=. flag=r  
546        filter="**\*akefile\***" {  
547         Header file=. CD=. flag=r filter="\*.h" {  
548         }  
549         Source file=. CD=. flag=r filter="**\*.cpp**" {  
550         }  
551        }  
552   
553     3. 在光标定位到第一行, 敲入:  
554         /R  
555        exampleProject文件改变, 如下:  
556              MyProject="**E:/desktop\_item/tmp/0virtual/nehe2/LVHM/test**" CD=./  
557                              flag=r filter="**\*akefile\***" {  
558               makefile  
559               Header file=. CD=. flag=r filter="\*.h" {  
560                MainFrm.h  
561                Resource.h  
562                StdAfx.h  
563                test.h  
564                testDoc.h  
565                testView.h  
566               }  
567               Source file=. CD=. flag=r filter="**\*.cpp**" {  
568                MainFrm.cpp  
569                StdAfx.cpp  
570                test.cpp  
571                testDoc.cpp  
572                testView.cpp  
573               }  
574              }  
575   
576     4. 小测一下:  
577         把光标某个文件名上, 敲下 Enter 可以看到, 对应的文件在左边的窗口显示出  
578         来.  
579   
580     5. 更多用法参考.vimrc的相关设置的说明或 help project.txt 查找帮助.  
581   
582   
583 #=============================================================================  
584 **14. NERD\_tree.vim**  
585 (1). 帮助手册查看  
586     help NERD\_tree  
587   
588 (2). 功能  
589     目录树, 同时还可以对文件进行打开操作等.  
590   
591 (3). 安装  
592     下载NERD\_tree压缩包, 然后把解压的两个文件NERD\_tree.vim 和 NERD\_tree.txt 分  
593     别放到$HOME/.vim/plugin 和 $HOME/.vim/doc 目录中.  
594   
595 (4). 使用方法  
596   
597 (5). 使用例子  
598     // let loaded\_nerd\_tree=1    // 禁用所有与NERD\_tree有关的命令  
599     nmap <silent> <leader>tto :NERDTreeToggle<cr>  
600     let NERDTreeIgnore=['/.vim$'**,**'/~$']    // 不显示指定的类型的文件  
601     let NERDTreeShowHidden=0    // 不显示隐藏文件(好像只在linux环境中有效)  
602     let NERDTreeSortOrder=['//$'**,**'/.cpp$'**,**'/.c$'**,**'/.h$'**,** '\*']    // 排序  
603     let NERDTreeCaseSensitiveSort=0     // 不分大小写排序  
604     let NERDTreeWinSize=30  
605     // let NERDTreeShowLineNumbers=1  
606     let NERDTreeShowBookmarks=1  
607     let NERDTreeQuitOnOpen=1    // 打开文件后, 关闭NERDTrre窗口  
608     // let NERDTreeHighlightCursorline=1     // 高亮NERDTrre窗口的当前行  
609     // nmap <silent> <leader>tmk :Bookmark expand(/"<cword>/")<cr>  
610   
611   
612 #=============================================================================  
613 **15. NERD\_commenter.vim**  
614 (1). 帮助手册查看  
615     help NERD\_commenter.  
616   
617 (2). 功能  
618     源码文档注释.  
619   
620 (3). 安装  
621     下载NERD\_commenter压缩包, 然后把解压的两个文件NERD\_commenter.vim 和  
622     NERD\_commenter.txt 分别放到$HOME/.vim/plugin 和 $HOME/.vim/doc 目录中.  
623   
624 (4). 使用方法  
625 (5). 使用例子  
626     "**let NERD\_java\_alt\_style=1**  
627 "**Default mapping: [count],cc**" 以行为单位进行注释.  
628     "**,c<space>**" comment <--> uncomment.  
629     "**,cm**" 以段作为单位进行注释.  
630     "**,cs**" 简洁美观式注释.  
631     "**,cy**" Same as ,cc except that the commented line(s) are yanked first.  
632     "**,c$**" 注释当前光标到行未的内容.  
633     "**,cA**" 在行尾进行手动输入注释内容.  
634     "**,ca**" 切换注释方式(/\*\*/ <--> //).  
635     "**,cl**" Same cc, 并且左对齐.  
636     "**,cb**" Same cc, 并且两端对齐.  
637     "**,cu**" Uncomments the selected line(s).  
638   
639   
640 #=============================================================================  
641 **16. sketch.vim   用鼠标作画**  
642 map <silent> <leader>ske :call ToggleSketch()<CR>  
643   
644   
645   
646   
647 #=============================================================================  
648 **17. Calendar.vim**  
649 map <silent> <leader>cal :Calendar<cr>  
650 map <silent> <leader>cah :CalendarH<cr>  
651   
652   
653   
654 #=============================================================================  
655 **18. DoxygenToolkit.vim**  
656 (1). 代码文档工具  
657 let g:DoxygenToolkit\_commentType = "**C**"  
658 let g:DoxygenToolkit\_briefTag\_pre="**@Synopsis**"  
659 let g:DoxygenToolkit\_paramTag\_pre="**@Param**"  
660 let g:DoxygenToolkit\_returnTag="**@Returns**"  
661 let g:DoxygenToolkit\_blockHeader="**--------------------------------------------------------------------------**"  
662 let g:DoxygenToolkit\_blockFooter="**----------------------------------------------------------------------------**"  
663 let g:DoxygenToolkit\_authorName="**Mathias Lorente**"  
664 let g:DoxygenToolkit\_licenseTag="**My own license**"  
665   
666   
667   
668 #=============================================================================  
669 **19. cpp.vim**  
670 (1). c/c++类名, 函数等高亮  
671 install details  
672 Make a backup copy of syntax/vim.cpp and overwrite syntax/vim.cpp with this file.  
673   
674   
675   
676 #=============================================================================  
677 **20. javacomplete.vim**  
678 (1). 帮助手册查看  
679     :help javacomplete.txt  
680 (2). 功能  
681     进行java类, 包, 方法等补全.  
682 (3). 安装  
683     1. 下载, 解压, 把相应的文件copy到相应的目录下.  
684     2. 对Reflection.java进行编译, 编译命令为:  
685         javac -source 1.4 Reflection.java  
686     3. 把编译生成的Reflection.class文件移动到vim的$HOME目录下.(注意是移动, 而  
687             不是复制, 最好保证系统中只有一个Reflection.class文件, 且在vim的  
688             $HOME目录下.)  
689     4. 在.vimrc中加入:  
690         setlocal completefunc=javacomplete#CompleteParamsInfo  
691         autocmd FileType c set omnifunc=ccomplete#Complete  
692         autocmd FileType css set omnifunc=csscomplete#CompleteCSS  
693         autocmd FileType html set omnifunc=htmlcomplete#CompleteTags  
694         autocmd FileType java set omnifunc=javacomplete#Complete  
695         autocmd FileType javascript set omnifunc=javascriptcomplete#CompleteJS  
696         autocmd FileType php set omnifunc=phpcomplete#CompletePHP  
697         autocmd FileType python set omnifunc=pythoncomplete#Complete  
698         autocmd FileType xml set omnifunc=xmlcomplete#CompleteTags  
699 (4). 使用方法  
700 (5). 使用例子  
701      Math.<C-X><C-O>  
702   
703   
704   
705 #=============================================================================  
706 **21. JumpInCode\_Plus.vim**  
707 (1). 帮助手册查看  
708     <http://www.vim.org/scripts/script.php?script_id=2862>  
709   
710 (2). 功能  
711     <leader>jc  Generate tags and cscope database from current directory to :  
712                    CurrentDirectory/OutDB/cscope.out,tags  
713     <leader>jst       list existed tags full name and choose tags  
714     <leader>jsc      list existed cscope database full name and choose cscope.out  
715   
716 (3). 安装  
717     把JumpInCode\_Plus.vim 放到 $VIM/vimfiles/plugin  
718   
719 (4). 使用方法  
720 (5). 使用例子  
721   
722   
723   
724 #=============================================================================  
725 **22. txtbrowser.zip**  
726 (1). 帮助手册查看  
727     <http://www.vim.org/scripts/script.php?script_id=2899>  
728     <http://guoyoooping.blog.163.com/blog/static/1357051832009112910162389/>  
729     :help txtbrowser  
730   
731 (2). 功能  
732        show the document map and syntax highlight in plain text.  
733   
734 (3). 安装  
735     :help txtbrowser  
736     1. 请首先安装taglist插件(<http://www.vim.org/scripts/script.php?script_id=273>),  
737     我相信玩Vim的没有几个人没有安装这个插件吧.  
738   
739     2. 下载插件后解压缩到目录$HOME/.vim(或$HOME/vimfiles, $VIM/vimfiles). 然后将解压后  
740     的.ctag, 剪切到$HOME目录下即可, 安装完成后你的文件夹下应该有下面的几个文件(请  
741     注意目录结构):  
742         $HOME/.ctags - 用于标题标签的生成.  
743         $HOME/.vim/syntax/txt.vim - .txt文件的语法高亮文件.  
744         $HOME/.vim/plugin/txtbrowser.vim - 浏览工具.  
745         $HOME/.vim/doc/txtbrowser.txt - 英文帮助文档.  
746         $HOME/.vim/doc/txtbrowser.cnx - 中文帮助文档.(别忘了打开然后执行  
747                           ":helptags ."生成标签)  
748     由于版本1.1.1之前采用老的插件名, 请在安装前先删除低版本插件(主要是帮助文件, 原来的  
749     名字叫txt.txt和txt\_cn.txt), 否则在安装新的帮助文档时会提示标签已经存在..  
750   
751     3. 在你的.vimrc文件里加入下面三行, 然后重启你的Vim:  
752         syntax on  
753         let tlist\_txt\_settings = '**txt;c:content;f:figures;t:tables**'  
754         au BufRead,BufNewFile \*.txt setlocal ft=txt  
755   
756     4. 本插件所有文件以fenc=utf8发布. 签于taglist对中文的支持有限, 你需要保证你的文  
757     档的enc值和fenc的值相等(例如, enc=cp936, fenc=cp936). 如果不等请将fenc的值设为  
758     相应的enc的值保(如:set fenc=cp936)存后并刷新(:TlistUpdate).  
759   
760 (4). 使用方法  
761     :help txtbrowser  
762 (5). 使用例子  
763   
764   
765   
766 #=============================================================================  
767 **23.  FindMate.vim**  
768 (1). 帮助手册查看  
769     help FindMate  
770     <http://www.vim.org/scripts/script.php?script_id=2871>  
771   
772 (2). 功能  
773     快速查找文件  
774   
775 (3). 安装  
776     下载FindMate压缩包, 然后把解压的两个文件FindMate.vim 和 FindMate.txt 分别放到  
777     $HOME/.vim/plugin 和 $HOME/.vim/doc 目录中.  
778   
779 (4). 使用方法  
780         You can launch FindMate by typing:  
781               ,, File\_name  
782         Or  
783               :FindMate File\_name  
784         The shortcut can be redefined by using:  
785               map your\_shortcut   <Plug>FindMate  
786         In your .vimrc file  
787   
788 (5). 使用例子  
789   
790   
791   
792   
793 #=============================================================================  
794 **24. ZoomWin.vim**  
795 (1). 帮助手册查看  
796     <http://www.vim.org/scripts/script.php?script_id=508>  
797   
798 (2). 功能  
799     Press <c-w>o : the current window zooms into a full screen  
800     Press <c-w>o again: the previous set of windows is restored  
801   
802 (3). 安装  
803     Press <c-w>o : the current window zooms into a full screen  
804     Press <c-w>o again: the previous set of windows is restored  
805   
806 (4). 使用方法  
807 (5). 使用例子  
808   
809   
810   
811 #=============================================================================  
812 **25. cpp\_src.tar.bz2**  
813     tags for std c++ (STL, streams, ...) : Modified libstdc++ headers for use with ctags  
814 (1). 帮助手册查看  
815     <http://www.vim.org/scripts/script.php?script_id=2358>  
816   
817 (2). 功能  
818 (3). 安装  
819     install details:  
820     1. unpack  
821     2. run  
822         $ ctags -R --c++-kinds=+p --fields=+iaS --extra=+q --language-force=C++ cpp\_src  
823         $ mv tags cpp # or whatever  
824     3. In Vim:  
825         set tags+=/my/path/to/tags/cpp   
826   
827 (4). 使用方法  
828 (5). 使用例子  
829   
830   
831   
832   
833 #=============================================================================  
834 **26. grep.vim**  
835 (1). 帮助手册查看  
836     <http://www.vim.org/scripts/script.php?script_id=311>  
837   
838 (2). 功能  
839     The grep plugin integrates the grep, fgrep, egrep, and agrep tools with  
840     Vim and allows you to search for a pattern in one or more files and jump  
841     to them.  
842   
843 (3). 安装  
844     1).  
845     To use this plugin, you need the grep, fgrep, egrep, agrep, find and  
846     xargs utilities. These tools are present in most of the Unix installations.  
847     For MS-Windows systems, you can download the GNU grep and find utilities  
848     from the following sites:  
849        <http://gnuwin32.sourceforge.net/packages/grep.htm>  
850        <http://gnuwin32.sourceforge.net/packages/findutils.htm>   
851     2).  
852     把下载得到的grep.vim复制到 $HOME/plugin 目录中.  
853   
854 (4). 使用方法  
855     :Grep  
856   
857 (5). 使用例子  
858   
859 #=============================================================================  
860 **27. autocomplpop.zip**  
861 (1). 帮助手册查看  
862     :help acp  
863     <http://www.vim.org/scripts/script.php?script_id=1879>  
864   
865 (2). 功能  
866     With this plugin, your vim comes to automatically opens popup menu for  
867     completions when you enter characters or move the cursor in Insert mode. It  
868     won't prevent you continuing entering characters.  
869   
870 (3). 安装  
871     autocomplpop.zip  
872     mv plugin/acp.vim $HOME/.vim/plugin/acp.vim  
873     mv autoload/acp.vim $HOME/.vim/autoload/acp.vim  
874     mv doc/acp.jax $HOME/.vim/doc/acp.jax   
875     mv doc/acp.txt $HOME/.vim/doc/acp.txt   
876       
877 (4). 使用方法  
878     Once this plugin is installed, auto-popup is enabled at startup by default.  
879     Which completion method is used depends on the text before the cursor. The  
880     default behavior is as follows:  
881         kind      filetype    text before the cursor ~  
882         Keyword   \*           two keyword characters  
883         Filename  \*           a filename character + a path separator  
884                               +0 or more filename character  
885         Omni      ruby        "**.**", "**::**" or non-word character + "**:**"  
886                               (|+ruby| required.)  
887         Omni      python      "**.**"(|+python| required.)  
888         Omni      xml         "**<**", "**</**" or("<" + non-">" characters + " ")  
889         Omni      html/xhtml  "**<**", "**</**" or("<" + non-">" characters + " ")  
890         Omni      css         (":", ";", "{", "^", "@", or "!")  
891                               +0 or 1 space  
892   
893 (5). 使用例子  
894   
895   
896   
897   
898   
899   
900   
901   
902   
903   
904   
905   
906   
907   
908   
909   
910   
911 #=============================================================================  
912 "<http://www.vim.org/scripts/script.php?script_id=1638>" SourceCodeObedience  
913 "<http://www.vim.org/scripts/script.php?script_id=1581>" lookupfile  
914 "<http://insenvim.sourceforge.net/index.php>" Vim Intellisense