

# vim+ctags+cscope 安装与使用总结

## ctags

### 1、安装 ctags

- 1) `sudo apt-get install ctags` 或者
- 2) 在 <http://ctags.sourceforge.net/> 下载源代码包之后，解压缩生成源代码目录  
进入源代码目录执行 `./configure & make & make install`

### 2、vim 中 ctags 简单使用

详细的使用 ctags 用法，在 vim 中使用 `:help tags`

#### 1) 生成 tags 文件

在源码根目录下执行 `ctags -R` 命令来为程序源代码生成标签文件，其 `-R` 选项表示递归操作，同时为子目录也生成标签文件，vim 利用生成的标签文件，可以进行检索，并在不同的文件、元素之间来回切换。  
也可以用 `ctags file_name1.c filename2.c filename3.h` 来产生 ctags 文件或者 `ctags *.c *.h`

#### 2) 字段补全

为了使得字段补全有效，在生成 tags 时需要一些额外的参数，推荐的 c++ 参数主要是：

`ctags -R --c++-kinds=+px --fields=+iaS --extra=+q`

我自己使用的如下：

`ctags -R --c++-kinds=+px --c-kinds=+px --fields=+iaS --extra=+f`

其中：

选项 `c++-kinds` 用于指定 C++ 语言的 tags 记录类型，`--c-kinds` 用于指定 c 语言的，通用格式是 `--{language}-kinds`

选项 `fields` 用于指定每条标记的扩展字段域

`extra` 选项用于增加额外的条目：`f` 表示为每个文件增加一个条目，`q` 为每个类增加一个条目

#### 3) 指定 tags 文件位置

可以手动指定 tags 文件

- a) 在 vim 命令中输入或者修改 .vimrc 文件 `set tags=./tags`（当前路径下的 tags 文件）  
如果要引用多个不同目录的 tags 文件，可以用逗号隔开，`set tags=path1, path2...`，或者  
`set tags+=path`  
`set tags+=path`

- b) 如果经常在不同工程里查阅代码，那么可以在 ~/.vimrc 中添加：

`set tags=tags;`

`set autochdir`

第一个命令里的分号是必不可少的，这个命令让 vim 首先在当前目录里寻找 tags 文件，如果没有找到 tags 文件，就到父目录中查找，一直向上递归。因为 tags 文件中记录的路径总是相对于 tags 文件所在的路径，所以要使用第二个设置项来改变 vim 的当前目录。

#### 4) 跳转到指定函数

在变量或函数处 `ctrl+]` 来跳转到变量或者函数定义的地方。`ctrl+t` 返回到跳转前的位置。

或者使用命令 `:tag func_nameshi` 来跳转到变量或者函数定义的地方。

ctags 不会生成局部变量的索引。

`:tags` 会列出查找 / 跳转过程

### 3、ctags 的局限性

tags 文件只能查看函数，类或变量的定义，而没有被调用信息。

如果要知道一个函数在什么地方被使用，需要使用 `cscope` 工具。

添加的 tags 最好是 source code 的索引，对于 include 头文件索引没有效果，

## taglist

taglist 插件是以 vim 脚本的形式存在，因此只需要将其下载下来放到相应的目录即可。

taglist 基于 ctags 才能发挥作用，因此要确保安装了 ctags。

通过在 vim 命令行下运行 `help taglist.txt` 查询 taglist 的用法。

其英文原版手册：<http://vim-taglist.sourceforge.net/manual.html>

### 1、安装 taglist

- 1) 从 <http://vim-taglist.sourceforge.net/index.html> 下载 taglist 安装包。
- 2) 进入 `~/vim` 目录，将 taglist 安装包解压，将解压后的 `/doc` 和 `/plugin` 目录复制到 `vim` 目录下
- 3) 进入 `~/vim/doc` 目录，在 vim 下运行 `helptag` .命令。这个步骤是将 doc 下的帮助文档加入到 vim 的帮助主题中，这样我们就可以通过在 vim 中运行 `help taglist.txt` 查看 taglist 帮助。
- 4) 打开配置文件 `~/vimrc`，加入以下两行：

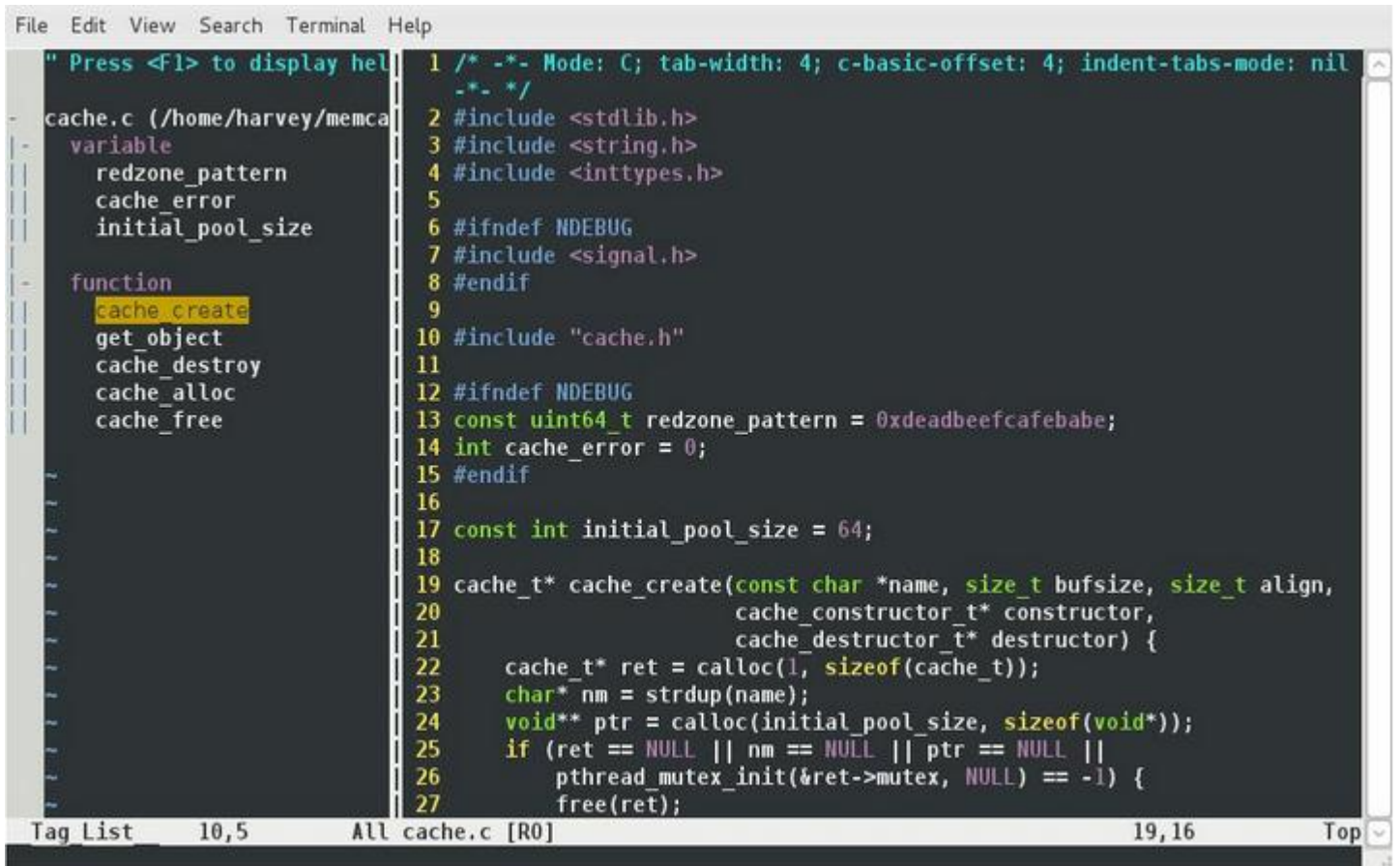
```
let Tlist_Show_One_File=1
```

```
let Tlist_Exit_OnlyWindow=1
```

到此安装已经完成。

在 vim 命令行下运行 `:Tlist(:TlistOpen, :TlistToggle)` 打开 Taglist 窗口，

运行 `:Tlist(:TlistClose, :TlistToggle)` 关闭 Taglist 窗口，



```
File Edit View Search Terminal Help
" Press <F1> to display help
cache.c (/home/harvey/memca
variable
  redzone_pattern
  cache_error
  initial_pool_size
function
  cache_create
  get_object
  cache_destroy
  cache_alloc
  cache_free
1 /* -*- Mode: C; tab-width: 4; c-basic-offset: 4; indent-tabs-mode: nil
  -*- */
2 #include <stdlib.h>
3 #include <string.h>
4 #include <inttypes.h>
5
6 #ifndef NDEBUB
7 #include <signal.h>
8 #endif
9
10 #include "cache.h"
11
12 #ifndef NDEBUB
13 const uint64_t redzone_pattern = 0xdeadbeefcafebabe;
14 int cache_error = 0;
15 #endif
16
17 const int initial_pool_size = 64;
18
19 cache_t* cache_create(const char *name, size_t bufsize, size_t align,
20                       cache_constructor_t* constructor,
21                       cache_destructor_t* destructor) {
22     cache_t* ret = calloc(1, sizeof(cache_t));
23     char* nm = strdup(name);
24     void** ptr = calloc(initial_pool_size, sizeof(void*));
25     if (ret == NULL || nm == NULL || ptr == NULL ||
26         pthread_mutex_init(&ret->mutex, NULL) == -1) {
27         free(ret);
28     }
29 }
```

我们可以通过 `ctrl+w` 快捷键或者鼠标点击在 Taglist 窗口和编辑区之间切换焦点，在 Taglist 窗口用鼠标选择某个符号并点击或者用键盘选择某个符号并回车，就可以跳转到该符号定义的位置。

## 2、其他一些选项可在 `~/.vimrc` 脚本中添加。如：

```
"设置 ctags 路径
let Tlist_Ctags_Cmd = '/usr/bin/ctags'
"启动 vim 后自动打开 taglist 窗口
let Tlist_Auto_Open = 1
"不同时显示多个文件的 tag，仅显示一个
let Tlist_Show_One_File = 1
"taglist 为最后一个窗口时，退出 vim
let Tlist_Exit_OnlyWindow = 1
"taglist 窗口显示在右侧，缺省为左侧
let Tlist_Use_Right_Window = 1
"设置 taglist 窗口大小
"let Tlist_WinHeight = 100
let Tlist_WinWidth = 40
"设置 taglist 打开关闭的快捷键 F8
noremap <F8> :TlistToggle<CR>
"更新 ctags 标签文件快捷键设置
noremap <F6> :!ctags -R<CR>
```

## NERDTree（感觉用处不大，没有安装）

1、下载地址：[http://www.vim.org/scripts/script.php?script\\_id=1658](http://www.vim.org/scripts/script.php?script_id=1658)

2、解压得到 doc/, nerdtree\_plugin/, plugin/, syntax/

将后面 3 个文件夹的 xxx.vim 结尾的插件复制到./vim/plugin/目录下，并 chmod a+x 权限，

将 doc/目录下的 NERD\_tree.txt 复制到./vim/doc 目录下并在当前目录下执行 vim，在 vim 命令行中:helptags .，这样以后就能在 vim 中通过:help NERD\_tree.txt 来查看对应的帮助文档。

现在已经可以通过:NERDTreeToggle 来在 vim 中显示文件列表，只不过现在文件列表栏显示在左边，需要修改配置。

## 3、配置

在./vimrc 内容中添加

```
let NERDTreeWinPos='right'
```

```
noremap <F9> :NERDTreeToggle<CR>
```

这样以后在 vim 中直接按 F9 就能在右边显示文件列表栏了。

## 4、知识

现在 NERDTree 好像不支持搜索文件的功能，只能一个一个找，如果要使 vim 支持搜索文件还要装对应的插件。

## cscope

### 1、安装 cscope

```
sudo apt-get install cscope
```

### 2、知识点

1) 生成索引文件 cscope -Rbq 则生成 cscope.in.out cscope.out cscope.po.out 三个文件

1) :cs add path/cscope.out 添加索引文件的位置

2)

:cs find c function 查看 function 被调用的位置

:cs find s 查找本 c 符号

:cs find g 查找本定义

:cs find d 查找本函数调用的函数

:cs find t 查找本字符串  
:cs find f 查找本文件  
:cs find i 查找包含本文件的文件

3)在~/.vimrc 中添加 set cscopequickfix=s-,c-,d-,i-,t-,e-

这样通过:cs find xxx 查找某个符号后，会立即跳转到第一个找到该符号出现的位置，  
然后通过:copen 来打开 quickfix 窗口，在 quickfix 窗口中显示所有出现的位置。

[Cscope 的使用（领略 Vim + Cscope 的强大魅力）](#)

## vim

### 1、高亮所有搜索模式匹配

shift + \* 向后搜索光标所在位置的单词  
shift + # 向前搜索光标所在位置的单词  
n 和 N 可以继续向后或者向前搜索匹配的字符串

:set hlsearch 高亮所有匹配的字符串  
:nohlsearch 临时关闭  
:set nohlsearch 彻底关闭，只有重新:set hlsearch 才可以高亮搜索

vim 高亮显示光标所在的单词，在单词的地方输入 gd

### 2、语法高亮

syntax on  
syntax off

### 3、vimgrep

vimgrep /匹配模式/[g]lj 要搜索的文件/范围  
g: 表示是否把每一行的多个匹配结果都加入  
j: 表示是否搜索完后定位到第一个匹配的位置  
vimgrep /pattern/% 在当前打开文件中查找  
vimgrep /pattern/\* 在当前目录下查找所有  
vimgrep /pattern/\*\* 在当前目录及其子目录下查找所有  
vimgrep /pattern/\*.c 查找当前目录下所有的.c 文件  
vimgrep /pattern/\*\*/\* 只查找子目录

cn 查找下一个

cp 查找上一个  
copen 打开 quickfix  
cw 打开 quickfix  
cclose 关闭 quickfix  
help vimgrep 查看 vimgrep 帮助

通过:vimgrep /pattern/gj path 来查找字符串，最好将右边的文件列表显示栏关闭，这样 quickfix 窗口显示比较方便查看。

**4、修改了配置文件.vimrc 并保存后，如果希望不重启 vim 而让配置立即生效，可以在打开的 vim 中执行:source ~/.vimrc**

## 5、vim 颜色配置方案

highlight 可以查看具体的颜色配置  
highlight LineNr term=underline,bold ctermfg=3 guifg=Brown

Set colorcolumn=80 设置边界线为 80 列  
Set colorcolumn=0 取消边界线设置  
Highlight colorcolumn ctermbg=4 guibg=Blue  
:help cterm-colors 可以查看颜色对应的值

## 6、vim 打开多个文件

vim file1 file2 file3 file4  
或者进入 vim 后使用:e 文档名来打开文档

:n 跳到后面那个文件  
:N 跳到前面那个文件  
:files 查看打开了哪些文件  
:file 查看当前的文件名

## 7、vim 自动补全

ctrl + n 或者 ctrl + p  
<http://jingyan.baidu.com/article/76a7e409b5d525fc3b6e15fc.html>

## 8、复制 vim 文件中所有内容

gg 回到文件首

shift + v 进入 VISUAL LINE 模式

shift + g 全选所有内容

ctrl + insert 复制所选的内容

## 9、[[跳到函数头部，]]跳到函数尾部

## 10、quickfix 是 vim 的标准插件，本身就带有

quickfix 功能将编译过程中产生的错误信息保存到文件中，然后 vim 利用这些信息跳转到源文件的对应位置，我们就可以进行错误的修正，之后跳到下一个错误重复上述操作，从而极大地提高编译错误的修改效率

quick 常用命令：

:cc 显示详细错误信息

:cp 跳到上一个错误

:cn 跳到下一个错误

:cl 列出所有的错误

:copen 打开 quickfix 窗口，可以在后面添加窗口高度参数，如 10 行，:copen 10

:cclose 关闭 quickfix 窗口

## 11、按 v 进入可视化界面，然后通过左右键选择文本，再按 y 进行复制，p 就可以进行粘帖了

## 12、不退出 vim，直接在 vim 中执行"!gcc file.c -o file"就可以编译程序了

## 13、vim 中查找特定字符串

/pattern enter

n 朝同一个方向搜索

N 朝反方向搜索

/ pattern enter

要查找单个单词，键入该单词，并在这个单词的两边都加上空格

/^pattern enter

要查找仅出现在行首的单词，请在该单词前面加上^

`/pattern$ enter`

要查找仅出现在行末的单词，请在该单词的后面加上\$

如果要把^和\$当作普通的符号，就需要在前面加上\