

linux 下把 vim 搞成 C/C++ 的 IDE

本来之前写代码都是用 ubuntu 自带的 gedit 的~

但是偶然听说了 vim 是编辑器之神!! 🤖

编辑器中的高富帅有木有!!! 🤖

大神们手中牛逼闪闪的神器有木有!!! 🤖

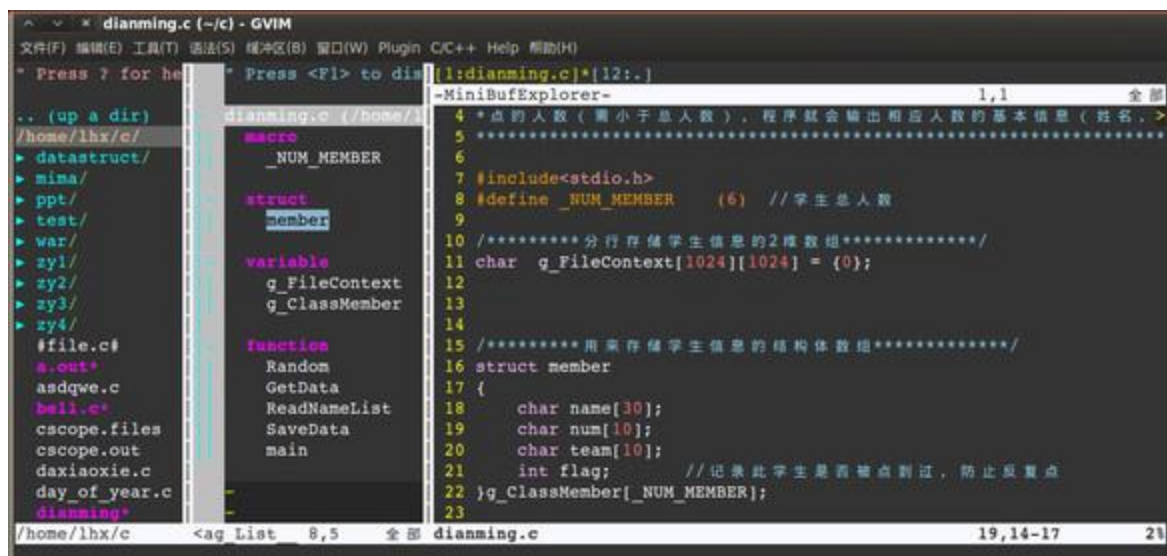


然后……就不淡定了~~~

作为一个屌丝码农，我也有颗装 B 的心啊!!! 🤖

所以，不行，我一定要用 vim 体验下大神们敲码的赶脚!!! 🤖

最后折腾了 2 天终于搞出了有点 IDE 感觉的样子：



(左边是目录；中间是各种变量名，函数名；右边是函数主体)

嘿嘿~~~~~看上去挺爽噶~

下面是步骤

之前就没碰过 vim，所以第一件事就是度娘。

vim 的百度百科：

Vim 是一个类似于 Vi 的文本编辑器，不过在 Vi 的基础上增加了很多新的特性，

Vim 普遍被推崇为类 Vi 中最好的一个，事实上真正的劲敌来自 Emacs 的不同变体。

1999 年 Emacs 被选为 Linuxworld 文本编辑分类的优胜者，Vim 屈居第二。

我是 ubuntu10.04 的

直接在终端输入 `sudo apt-get install vim ctags cscope` 就行了，会自己安装的

`ctags` 主要是用来生成什么标签的~~我也不太懂，反正就是你可以靠这个在你写的

各个函数之间跳来跳去的~~~~飞一般的感觉！！😄爽

`cscope` 呢 用 `Cscope` 自己的话说 - "你可以把它当做是超过频的 `ctags`", 反正就是加强版啦~

具体用法等会再说~~~~

2. 配置文件

具体就是定制你需要什么功能~~~

首先嘛看看你的 `home` 目录(就是~/啦)下有没有 `.vimrc` 这个文件，没有就建个~这就是你的配置文件了，

把下面这些乱七八糟的东西复制进去（我比较懒…就把自己的全部贴出来，每条的功能具体解释看注释，自行选择，其实我也是从大牛那复制过来的😄）：

```
if(has("win32") || has("win95") || has("win64") || has("win16")) "判定当前操作系统类型
let g:iswindows=1
else
let g:iswindows=0
endif
*****文件设置*****
set fileencodings=utf-8,gb18030,utf-16,big5 "编码设置
set nocompatible "不要 vim 模仿 vi 模式，建议设置，否则会有很多不兼容的问题\
set noswapfile "不生成 swap 文件
set nowb
set nobackup "不会生成备份文件

*****常规设置*****
"字体的设置
set guifont=Courier\ 10\ Pitch\ 12 ""记住空格用下划线代替哦
set gfw=幼圆:h10:cGB2312

"let mapleader = "," ""
"let g:mapleader = "," ""设置 leader 为',',默认为\

colo Dark "配色方案
syntax on "打开高亮
""set cursorline " 高亮当前行
set ruler " 在编辑过程中，在右下角显示光标位置的状态行
```

```
set incsearch " 查询时非常方便，如要查找 book 单词，当输入到/b 时，会自动找到
" 第一个 b 开头的单词，当输入到/bo 时，会自动找到第一个 bo 开头的
" 单词，依次类推，进行查找时，使用此设置会快速找到答案，当你
" 找要匹配的单词时，别忘记回车
set vb t_vb= " vim 进行编辑时，如果命令错误，会发出一个响声，该设置去掉响声
set mouse=a "允许鼠标的使用
```

```
"set listchars=tab:>-,trail:- "将 tab 符显示为>---，并将行尾的空格显示为一。如果你不喜欢这个特性，注
释这两句即可
```

```
"set list
```

```
#####编程设置#####
```

```
set tabstop=4 "设置 tab 键的宽度
set shiftwidth=4 " 换行时行间交错使用 4 个空格
set expandtab "空格代替 TAB
set nowrap "no 自动换行
set backspace=indent,eol,start whichwrap+=<,>,[,]" 允许退格键的使用
""set ai! " 设置自动缩进
set nu! " 显示行号
set showmatch " 设置匹配模式，类似当输入一个左括号时会匹配相应的那个右括号
```

```
filetype plugin on ""自动匹配文件
if has("autocmd")
filetype plugin indent on "根据文件进行缩进
augroup vimrcEx
au!
autocmd FileType text setlocal textwidth=78
autocmd BufReadPost *
\ if line("'"') > 1 && line("'"') <= line("$") |
\ exe "normal! g`'" |
\ endif
augroup END
else
"智能缩进，相应的有 cindent，官方说 autoindent 可以支持各种文件的缩进，
"但是效果会比只支持 C/C++的 cindent 效果会差一点，但笔者并没有看出来
set autoindent " always set autoindenting on
endif " has("autocmd")
```

```
#####ctags 和 cscope（绑定 F12）#####
```

```
map <F12> :call Do_CsTag()<CR>
nmap <C-@>s :cs find s <C-R>=expand("<cword>")<CR><CR>:copen<CR>
nmap <C-@>g :cs find g <C-R>=expand("<cword>")<CR><CR>
nmap <C-@>c :cs find c <C-R>=expand("<cword>")<CR><CR>:copen<CR>
```

```

nmap <C-@>t :cs find t <C-R>=expand("<cword>")<CR><CR>:copen<CR>
nmap <C-@>e :cs find e <C-R>=expand("<cword>")<CR><CR>:copen<CR>
nmap <C-@>f :cs find f <C-R>=expand("<cfile>")<CR><CR>:copen<CR>
nmap <C-@>i :cs find i ^<C-R>=expand("<cfile>")<CR>$<CR>:copen<CR>
nmap <C-@>d :cs find d <C-R>=expand("<cword>")<CR><CR>:copen<CR>
function Do_CsTag()
let dir = getcwd()
if filereadable("tags")
if(g:iswindows==1)
let tagsdeleted=delete(dir."\\". "tags")
else
let tagsdeleted=delete("./". "tags")
endif
if(tagsdeleted!=0)
echohl WarningMsg | echo "Fail to do tags! I cannot delete the tags" | echohl None
return
endif
endif
if has("cscope")
silent! execute "cs kill -1"
endif
if filereadable("cscope.files")
if(g:iswindows==1)
let csfilesdeleted=delete(dir."\\". "cscope.files")
else
let csfilesdeleted=delete("./". "cscope.files")
endif
if(csfilesdeleted!=0)
echohl WarningMsg | echo "Fail to do cscope! I cannot delete the cscope.files" | echohl None
return
endif
endif
if filereadable("cscope.out")
if(g:iswindows==1)
let csoutdeleted=delete(dir."\\". "cscope.out")
else
let csoutdeleted=delete("./". "cscope.out")
endif
if(csoutdeleted!=0)
echohl WarningMsg | echo "Fail to do cscope! I cannot delete the cscope.out" | echohl None
return
endif
endif
if(executable('ctags'))

```

```
"silent! execute "!ctags -R --c-types=+p --fields=+S *"
silent! execute "!ctags -R --c++-kinds=+p --fields=+iaS --extra=+q ."
endif
if(executable('cscope') && has("cscope"))
if(g:iswindows!=1)
silent! execute "!find . -name '*.h' -o -name '*.c' -o -name '*.cpp' -o -name '*.java' -o -name '*.cs' >
cscope.files"
else
silent! execute "!dir /s/b *.c,*.cpp,*.h,*.java,*.cs >> cscope.files"
endif
silent! execute "!cscope -b"
execute "normal :"
if filereadable("cscope.out")
execute "cs add cscope.out"
endif
endif
endif
endifunction
```

=====进行 Taglist 的设置=====

```
"TlistUpdate 可以更新 tags
map <F3> :silent! Tlist<CR> "按下 F3 就可以呼出了,可以改
let Tlist_Ctags_Cmd='ctags' "因为我们放在环境变量里,所以可以直接执行
let Tlist_Use_Right_Window=0 "让窗口显示在右边, 0 的话就是显示在左边
let Tlist_Show_One_File=1 "让 taglist 可以同时展示多个文件的函数列表, 如果想只有 1 个, 设置为 1
let Tlist_File_Fold_Auto_Close=1 "非当前文件, 函数列表折叠隐藏
let Tlist_Exit_OnlyWindow=1 "当 taglist 是最后一个分割窗口时, 自动推出 vim
let Tlist_Process_File_Always=0 "是否一直处理 tags.1:处理;0:不处理。不是一直实时更新 tags, 因为没有必要
let Tlist_Inc_Winwidth=0
```

=====对 NERD_commenter 的设置(注释)=====

```
let NERDShutUp=1
"支持单行和多行的选择, //格式
map <C-h> ,c<space>
"=====NERDTree"=====map <F2> :NERDTree<cr> "F2 打开
NERDTree
map <F4> :NERDTreeClose<cr> "F4 关闭·····
```

=====括号和引号补全=====

```
inoremap ( (<Esc>i
inoremap [ [<Esc>i
inoremap { {<CR>}<Esc>O
autocmd Syntax html,vim inoremap < <lt>><Esc>i| inoremap > <c-r>=ClosePair('>')<CR>
inoremap ) <c-r>=ClosePair('')<CR>
```

```

inoremap ] <c-r>=ClosePair('')<CR>
inoremap } <c-r>=CloseBracket()<CR>
inoremap " <c-r>=QuoteDelim('')<CR>
inoremap ' <c-r>=QuoteDelim('')<CR>

```

```

function ClosePair(char)
if getline('.')[col('.') - 1] == a:char
return "\<Right>"
else
return a:char
endif
endf

```

```

function CloseBracket()
if match(getline(line('.') + 1), '\s*}') < 0
return "\<CR>}"
else
return "\<Esc>j0f}a"
endif
endf

```

```

function QuoteDelim(char)
let line = getline('.')
let col = col('.')
if line[col - 2] == "\""
"Inserting a quoted quotation mark into the string
return a:char
elseif line[col - 1] == a:char
"Escaping out of the string
return "\<Right>"
else
"Starting a string
return a:char.a:char."\<Esc>i"
endif
endf

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%配置代码到此为止%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

3.好啦，现在可以添加各种插件了

(1) ctags

这个刚才已经安装好了，但是用之前要先生成 **tags**，先找到你写代码的目录（假设你写代码的目录为 `~/c`），输入 `"ctags -R"` OK~~~你就会发现这个目录下多了个 **tags** 文件，然后在 **vim** 的一般模式下输入 `:set`

tags=~c/tags 就可以开始用了~但是我觉得有点麻烦! 好在我找了大牛的配置, 就是我上面贴的
""""""ctags 和 cscope (绑定 F12)""""那段, 现在可以忘记上面的操作了, 你可以直接在 vim 的界面
下按一下 F12, tags 就会自动生成并设置了!!!!!!

现在说说具体用法: ctrl+] !!!

对, 你没看错……就 ctrl+] 就行了, 光标移到某个函数上, 像这样:

```
    }  
ef FEAT_GUI  
    if (need_mouse_correct)  
        gui_mouse_correct();  
if
```

然后 ctrl+] 当当当!!!

```
1,11  
4592 gui_mouse_correct()  
4593 {  
4594     int          x, y;  
4595     win_T        *wp = NULL;  
4596  
4597     need_mouse_correct = FALSE;  
4598  
4599     if (!(gui.in_use && p_mousef))  
4600         return;  
4601  
4602     gui_mch_getmouse(&x, &y);  
4603     /* Don't move the mouse when it's left or  
4604     if (x < 0 || x > Columns * gui.char_width)  
4605         return;  
4606 # ifdef FEAT_WINDOWS  
4607     if (Y_2_ROW(y) >= tabline_height())  
4608 # else  
4609     if (y >= 0)  
4610 # endif  
4611     wp = xy2win(x, y);
```

光标就跳到这个函数定义的地方了~~~如果你想跳回去: ctrl+t 就行了, 安逸啊!!!

(2) cscope

加强版的 ctags, 用法也差不多, 但是我没怎么用~~~还是说说用法吧: 一样先生成 cscope 文件, 然后倒入 vim 由于有了上面的配置, 可以一个 F12 就把上面的操作搞定然后实操吧: cscope 的主要功能是通过同的子命令"find"来实现的。

"cscope find"的用法:

:cs find c|d|e|f|g|i|s|t name

0 或 s	查找本 C 符号(可以跳过注释)
1 或 g	查找本定义
2 或 d	查找本函数调用的函数
3 或 c	查找调用本函数的函数
4 或 t	查找本字符串
6 或 e	查找本 egrep 模式
7 或 f	查找本文件
8 或 i	查找包含本文件的文件

如果你不想输入命令的话，我的配置里也映射了快捷键：当光标停在某个你要查找的词上时，按下 `<ctrl+@>g`，就是查找该对象的定义，其他的同理。按这种组合键有一点技巧，按了 `<ctrl+@>` 后要马上按下一个键，否则屏幕一闪就回到 normal 状态了。 `<ctrl+@>g` 的按法是先按 "Ctrl+Shift+2"，然后很快再按 "g"

(3)TagsList（高效浏览源码）

下载地址：http://www.vim.org/scripts/script.php?script_id=273

```

" Press <F1> to display help
gui.c (/home/lhx/下载/vim70/
macro
    SCROLL_PAST_END
    MAY_FORK
    FONTLEN

variable
    gui
    can_update_cursor
    prev_which_scrollbars
    longest_lnum

function
    gui_start
    gui_prepare
    gui_init_check
    gui_init
    gui_exit

```

其实就是我第一张图中间的那部分，用来集中看函数啊，变量啊什么的。东西下载好了直接解压到 `~/.vim` 文件夹下（没有的自己建一个）

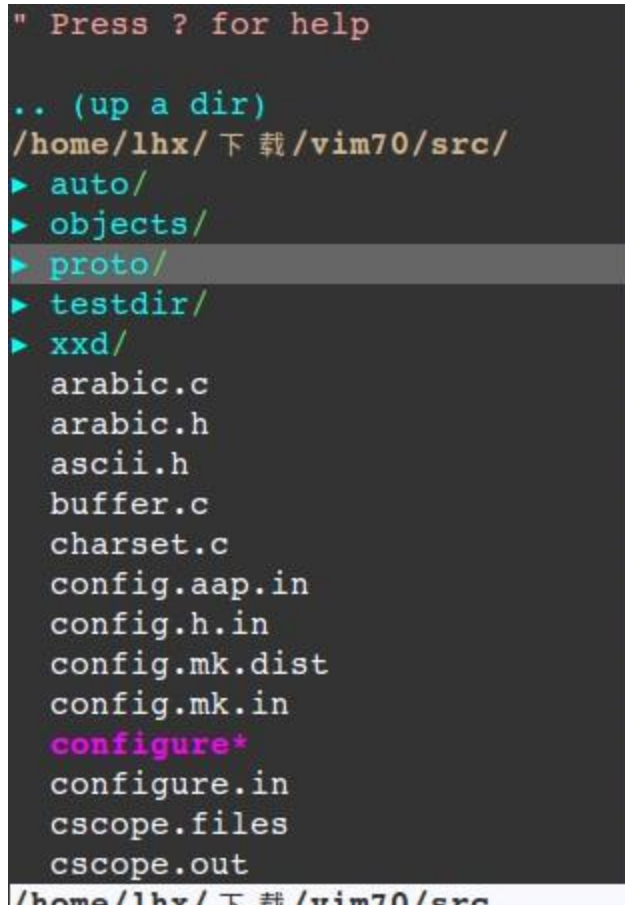
`:TlistOpen` 打开

`:TlistClose` 关闭

针对这个插件设置了一个快捷键 F3（可以在配置文件.vimrc 里面改），按一下打开，再按一下关了~方便吧

（4）NERD_tree(文件浏览)

下载地址：http://www.vim.org/scripts/script.php?script_id=1658 安装方法也是直接解压到.vim 文件夹里面



```
" Press ? for help

.. (up a dir)
/home/lhx/ 下载/vim70/src/
▶ auto/
▶ objects/
▶ proto/
▶ testdir/
▶ xxd/
  arabic.c
  arabic.h
  ascii.h
  buffer.c
  charset.c
  config.aap.in
  config.h.in
  config.mk.dist
  config.mk.in
  configure*
  configure.in
  cscope.files
  cscope.out
/home/lhx/ 下载/vim70/src
```

强大的文件浏览插件，就是我第一张图左边那块，同样也有快捷键：

F2 是开，F4 是关！

更多的操作快捷键我直接贴网上的了：

你可以双击文件在当前的窗口打开，也可以中键点击文件，在一个新的分割窗口内打开，也可以用 t 键，在一个新的标签页打开文件，C 键可以把当前的目录作为顶级目录，? 就可以得到一个常用命令手册，更详细的命令和功能可以查看 NERD tree 的帮助

:help NERD_tree.txt 。

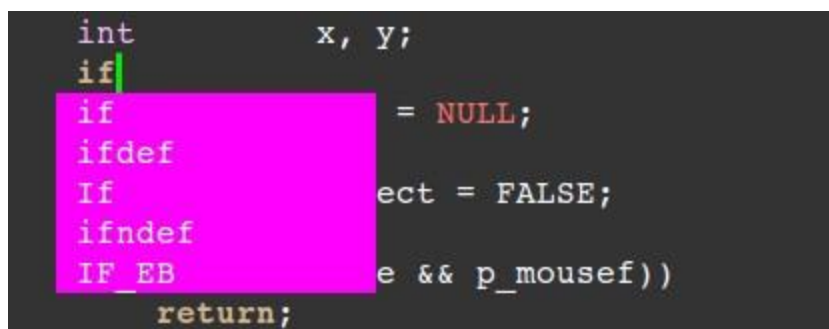
1. o 打开关闭文件或者目录
2. t 在标签页中打开
3. T 在后台标签页中打开
4. ! 执行此文件
5. p 到上层目录
6. P 到根目录
7. K 到第一个节点
8. J 到最后一个节点
9. u 打开上层目录
10. m 显示文件系统菜单（添加、删除、移动操作）
11. ? 帮助
12. q 关闭

(5) autocmplpop(代码自动补全)

下载地址: http://www.vim.org/scripts/script.php?script_id=1879

安装方法同上

不多说, 上图:



就是你敲代码的时候会跳出那个粉色的框框提醒你, 你可以自己选~~~~

(6)code_complete (函数补全)

下载地址: http://www.vim.org/scripts/script.php?script_id=1764

直接把下载的 `code_complete.vim` 文件放到 `.vim` 文件夹下的 `plugin` 文件夹（没有就自己建）里就行了

具体效果看这个链接里面的动态图:

<http://blog.chinaunix.net/space.php?uid=152156&do=blog&id=2774466>

爽了吧!!!!

(7) NERD_commenter (快速注释)

下载地址: http://www.vim.org/scripts/script.php?script_id=1218

一样是解压到.vim 中

操作快捷键:

\ca 在可选的注释方式之间切换, 比如 C/C++ 的块注释/* */和行注释//

\cc 注释当前行

\c 切换注释/非注释状态

\cs 以“性感”的方式注释

\cA 在当前行尾添加注释符, 并进入 Insert 模式

\cu 取消注释

Normal 模式下, 几乎所有命令前面都可以指定行数

Visual 模式下执行命令, 会对选中的特定区块进行注释/反注释

各命令前缀是可以自己设置的, 通常是逗号','或者'\'.

好了, 有了这些插件 **VIM** 就变成一个相当不错的 **C/C++** 的 **IDE** 了~~

4. 配色主题

如果觉得 vim 的配色主题不好看, 可以到这里

<http://vimcolorschemetest.googlecode.com/svn/html/index-c.html>

下载, 下载的文件保存成{主题名.vim}直接放到.vim 文件夹的 colors 文件夹

(没有就自己建)里, 然后在.vimrc 配置文件中加上一句

colo 主题名

就 OK 了!!!!!!

终于写完了~累死了…

