

ctags 助力 vim 实现 gtk 接口的自动补全

一、什么是 ctags

参考:

http://easwy.com/blog/archives/exuberant-ctags-chinese-manual/

- 1、ctags 为源代码产生标签文件。
- 2、ctags 和 etags 程序 (下文中统称为 ctags, 除非特别指明) 为文件中的各种语言对象生成一个索引(或称为标签) 文件。标签文件允许这些项目能够被一个文本编辑器或其它工具简捷迅速的定位。一个"标签"是指一个语言对象,它对应着一个有效的索引项(或者换言之,这个索引项为这个对象而创建)。
- 3、ctags 能够为多种程序语言文件的语言对象信息生成可读格式的交叉索引列表。请见 -list-languages 和 -list-kinds 选项。
- 4、标签索引文件被多种编辑器支持。这些编辑器允许用户通过出现在源文件中的名字定位对象,并且跳转到定义这个名字的文件和行。在这个版本发布时,我们所知的编辑器包括: Vi (1) 及其变种 (例如,Elvis,Vim,Vile, Lemmy), CRiSP, Emacs, FTE (Folding Text Editor), JED, jEdit, Mined, NEdit (Nirvana Edit), TSE (The SemWare Editor), UltraEdit, WorkSpace, X2, Zeus。

二、ctags 助力 vim

参考:

徐哥之《有关vim下gtk自动补全》



直接运行压缩包中的 copy_con. sh 脚本 (./copy_con. sh) 即可完成配置,

如果想自己动手实现,参考下面的步骤

- 1、首先进入/usr/include/gtk-2.0/gtk 目录,下面有很多头文件,我们 要在此目录下生成一个 tags 文件供使用。
- 2、执行ctags -R(我们之前就是这一步出了问题)

ctags --file-scope=yes --langmap=c: +. h --languages=c, c++

- --links=yes --c-kinds=+p -R
- --c-kinds=+p 则告诉 ctags 需要为函数原型的声明也生成 tags。
- --langmap=c: +. h 表示. h 视为 c 文件而不是 c++文件。
- 3、将生成的 tags 文件 copy 到家目录下的. vim 中
- cp tags \$HOME/.vim

修改\$HOME/.vimrc 大约 66 行增加以下语句

tags = ./tags,/usr/include/tags,\$HOME/.vim/tags即可

4、0K, 大功告成。

这下你写程序时,即可 ctrl-N/P 来自动补全 gtk 的 API 了。

- ctrl + P: 补全所有 gtk 函数
- ctrl + N: 可补全本文件你敲过的函数及变量
- 5、此方法不仅可以针对 gtk 开发, 也可以引申到其他工程(或者自建工 程)的开发
- 三、ctags 常用功能

强大的补全功能(在使用自动补齐时需要多敲一些关键字,gtk_in才行,gtk 就不行)

1、自动补全接口

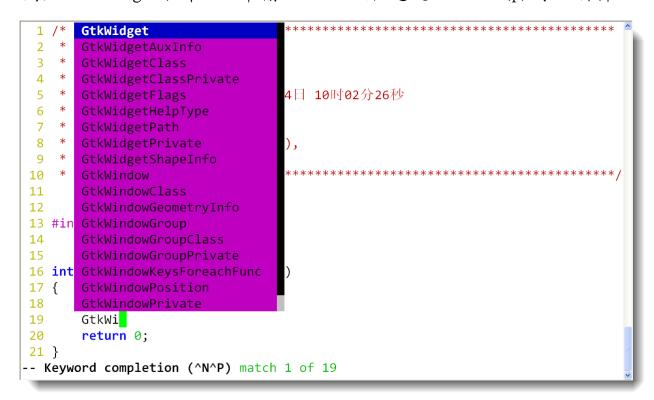


例如 gtk_init, 在 vi 中输入 gtk_in 后通过 ctrl+n(p) 可以补齐

```
gtk_info_bar_add_action_widget
 2
       gtk_info_bar_add_button
 3
       gtk_info_bar_add_buttons
 4
       gtk_info_bar_get_action_area
                                              26秒
       gtk info bar get content area
       gtk_info_bar_get_message_type
 7
       gtk_info_bar_new
 8
       gtk_info_bar_new_with_buttons
 9
       gtk info bar response
       gtk_info_bar_set_default_response
10
11
       gtk_info_bar_set_message_type
12
       gtk_info_bar_set_response_sensitive
13 #in gtk_init
       gtk_init_abi_check
14
15
       gtk_init_add
16 int gtk_init_check
       gtk_init_check_abi_check
18
       gtk in
19
       return 0;
20 }
21
-- Keyword completion (^N^P) match 13 of 26
```

2、自动补全对象

例如 GtkWidget, 在 vi 中输入 GtkWi 后通过 ctrl+n(p) 可以补齐





3、自动补全枚举类型

例如 GTK_WINDOW_TOPLEVEL, 在 vi 中输入 GTK_WI 后通过 ctrl+n(p)可以补齐

```
Version:
                        1.0
                        2013年11月04日 10时02分26秒
             Created:
            Revision:
 7
            Compiler:
                                            GTK_WINDOW
                        gcc
 8
                                            GTK_WINDOW_CLASS
              Author:
                        YOUR NAME (),
                                           GTK WINDOW GET_CLASS
 9
             Company:
                                           GTK WINDOW GROUP
11
                                           GTK WINDOW GROUP CLASS
                                            GTK WINDOW GROUP GET CLASS
12
13 #include <gtk/gtk.h>
                                            GTK_WINDOW_POPUP
                                           GTK_WINDOW_TOPLEVEL
14
15
                                            GTK_WIN_POS_CENTER
16 int main(int argc, char *argv[])
                                            GTK_WIN_POS_CENTER_ALWAYS
                                            GTK WIN POS CENTER ON PARENT
       gtk_init(&argc, &argv);
                                           GTK_WIN_POS_MOUSE
18
                                           GTK WIN POS NONE
19
       GtkWidget *window = gtk window new(GTK WIN);
20
21
22
23
       return 0;
24 }
-- Keyword completion (^N^P) match 50 of 59
```

4、自动补全宏

例如 GTK_CONTAINER, 在 vi 中输入 GTK_C 后通过 ctrl+n(p) 可以补齐



```
Version:
                       1.0
 4
             Created: 2013年11月04日 10时02分26秒
 5
            Revision: none
 7
            Compiler:
                       gcc
 8
              Author: YOUR NAME (),
 9
             Company:
10
11
12
13 #include <gtk/gtk.h>
14
15
16 int main(int argc, char *argv[])
17 {
       gtk_init(&argc, & GTK_CONTAINER
18
19
                         GTK CONTAINER CLASS
20
       GtkWidget *window GTK CONTAINER GET CLASS
21
       GtkWidget *button GTK CONTAINER RESIZE PENDING
22
                         GTK_CONTAINER_WARN_INVALID_CHILD_PROPERTY_ID
       gtk_container_add(GTK_CON);
23
24
-- Keyword completion (^N^P) match 168 of 225
```

5、自动补全信号、事件

例如信号 clicked, 在 vi 中输入 cli 后通过 ctrl+n(p)可以补齐

```
6
             Revision:
                        none
                                   cli$dcl_parse
 7
             Compiler:
                        gcc
                                   cli$get_value
                        YOUR NAME cli$present
 8
              Author:
 9
             Company:
                                   click
10
                                  click_cell
11
                                   click column
                                   click count
13 #include <gtk/gtk.h>
                                   click_pos
14
                                  clicked
15
                                   client
16 int main(int argc, char *argv clientCmaps
17 {
                                   clientDied
       gtk_init(&argc, &argv);
18
                                   clientId
19
                                   clientMajor
       GtkWidget *window = gtk_w clientMessage
20
21
       GtkWidget *button = gtk_b clientMinor
22
                                   clientStarted
23
       gtk container add(GTK CON clientSwapped
24
       g_signal_connect(button, "cli");
25
26
       return 0;
-- Keyword completion (^N^P) match 9 of 62
```

6、自动补全自己定义的变量和函数

例如 window, 在 vi 中输入 w 后通过 ctrl+n(p) 可以补齐 凌阳教育——全国唯一学员就业双 100%品牌: 100%就业, 100%满意就业 免费咨询: 400-705-9680



```
Version:
                        1.0
                                        window
 4
 5
             Created: 2013年11月04日
                                        wBlockLength
            Revision: none
                                        wBytesPerInterval
 7
            Compiler:
                        gcc
                                        wChannelConfig
 8
              Author:
                       YOUR NAME (),
                                        wCompQuality
 9
             Company:
                                        wCompWindowSize
10
                                        wCountyCode0
11
                                        wData
                                        wDatagramIndex
12
13 #include <gtk/gtk.h>
                                        wDatagramLength
14
                                        wDelay
15
                                        wFormatTag
16 int main(int argc, char *argv[])
                                        wHeaderLength
17 {
                                        wHeight
       gtk_init(&argc, &argv);
                                        wHubChange
18
19
                                        wHubCharacteristics
20
       GtkWidget *window = gtk window
                                        wHubDelay
       GtkWidget *button = gtk_button_
21
                                        wHubStatus
22
                                         wIndex
23
       gtk_container_add(GTK_CONTAINER(w));
24
-- Keyword completion (^N^P) match 1 of 301
```

例如 on_click_button, 在 vi 中输入 on 后通过 ctrl+n (p) 可以补齐

```
Revision:
 6
 7
            Compiler:
                        gcc
 8
                        YOUR NAME (),
               Author:
 9
             Company:
10
11
12
13 #include <gtk/gtk.h>
14
15 void on
16
17 int on_click_button
18 {
        on_select_clist_row
19
        on_select_row_tree_view
20
        on1
21
        on2
                                       new(GTK WINDOW TOPLEVEL);
                                       _new_with_label("button");
22
        on3
23
        onColorNdx
                                       R(window), button);
24
        onConnection
25
        onOff
                                       ked", on_click_button, NULL);
        on color ndx
-- Keyword completion (^N^P) match 2 of 21
```

7、其他的类型定义、类、结构体、联合体等都可以补齐 光标移动

参考:



http://361324767.blog.163.com/blog/static/11490252520127219362 3255/

1, vim -t foo-bar

这个命令将打开定义"foo_bar"(变量或函数或其它)的文件,并把光标 定位到这一行。

```
jinyan@3$vim -t on_click_button
   " Press <F1> to displa
                                       Filename:
                                    Description:
  3.c (/home/jinyan/clas
                                        Version: 1.0
                                        Created: 2013年11月04日 10时02分26秒
     function
       on_click_button
                                       Revision: none
       main
                                       Compiler: gcc
                                         Author: YOUR NAME (),
                                        Company:
                           12
                           13 #include <gtk/gtk.h>
                           15 void on_click_button(GtkButton *button, gpointer u
                           16
                           17
                                  g_print("hello\n");
                           18 }
                           19
                           20 int main(int argc, char *argv[])
```

2、把光标移到变量名或函数名上,然后按下"Ctrl-]"。用"Ctrl-o(t)" 退回原来的地方。



```
void on_click_button(GtkButton *button, gpointer user_pointer)
16
         print("hello\n");
17
18 }
     ctrl+o(t)
19
20 int main(int argc, char *argv[])
21 {
        gtk_init(&argc, &argv); ctrl+]
22
23
        GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
24
25
        GtkWidget *button = gtk_button_new_with_label("button");
26
        gtk_container_add(GTK_CONTAINER(window), button);
27
       g_signal_connect(button, "clicked", on click_button, NULL);
28
29
        gtk_widget_show_all(window);
30
31
32
        return 0;
33 }
34
35
"3.c" 35L, 832C
                                                                28,41-44
                                                                               Bot
```

3、[[转到上一个位于第一列的"{";]]转到下一个位于第一列的"{"

```
15 void on_click_button(GtkButton *button, gpointer user_pointer)
16 {
       g_print("hello\n");
17
18 }
19
20 int main(int argc, char *argv[])
21
       gtk_init(&argc, &argv);
22
23
24
       GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
25
       GtkWidget *button = gtk_button_new_with_label("button");
26
       gtk_container_add(GTK_CONTAINER(window), button);
27
28
       g_signal_connect(button, "clicked", on_click_button, NULL);
29
30
       gtk_widget_show_all(window);
31
32
       return 0;
33
34
35 [[
                                                               21,1
                                                                              Bot
```

4、{转到上一个空行;}转到下一个空行



```
15 void on_click_button(GtkButton *button, gpointer user_pointer)
16 {
17
       g_print("hello\n");
18
19
20 int main(int argc, char *argv[])
21 {
22
       gtk_init(&argc, &argv);
23
24
       GtkWidget *window = gtk window new(GTK WINDOW TOPLEVEL);
25
       GtkWidget *button = gtk_button_new_with_label("button");
26
27
       gtk_container_add(GTK_CONTAINER(window), button);
28
       g_signal_connect(button, "clicked", on_click_button, NULL);
29
30
       gtk_widget_show_all(window);
31
32
       return 0;
33 }
34
35
                                                               20,1
                                                                              Bot
```

5、gd 转到当前光标所指的局部变量的定义

```
15 void on_click_button(GtkButton *button, gpointer user_pointer)
16 {
17
       g_print("hello\n");
18 }
19
20 int main(int argc, char *argv[])
21 {
       gtk_init(&argc, &argv);
22
23
24
       GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
25
       GtkWidget *button = gtk_button_new_with_label("button");
26
27
       gtk_container_add(GTK_CONTAINER(window), button);
       g_signal_connect(button, "clicked", on_click_button, NULL);
28
29
30
       gtk_widget_show_all(window);
31
32
       return 0;
33 }
34
35
                                                               30,22-25
                                                                              Bot
```

6、* 转到当前光标所指的单词下一次出现的地方; # 转到当前光标所指的单词上一次出现的地方



```
15 void on_click_button(GtkButton *button, gpointer user_pointer)
16 {
       g_print("hello\n");
17
18 }
19
20 int main(int argc, char **argv[])
21 {
22
       gtk_init(&argc, &argv);
23
24
       GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
25
       GtkWidget *button = gtk_button*new_with_label("button");
26
27
       gtk_container_add(GTK_CONTAINER(window), button);
       g_signal_connect(button, "clicked", on_click_button, NULL);
28
29
30
       gtk_widget_show_all(window);
31
32
       return 0;
33 }
34
35
search hit BOTTOM, continuing at TOP
                                                               15,33
                                                                              Bot
```