

## 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、OK，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

```

1 /* 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
5 * gtk_info_bar_get_content_area 26秒
6 * gtk_info_bar_get_message_type
7 * gtk_info_bar_new
8 * gtk_info_bar_new_with_buttons
9 * gtk_info_bar_response
10 * gtk_info_bar_set_default_response ***** /
11 gtk_info_bar_set_message_type
12 gtk_info_bar_set_response_sensitive
13 #in gtk_init
14 gtk_init_abi_check
15 gtk_init_add
16 int gtk_init_check
17 { 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

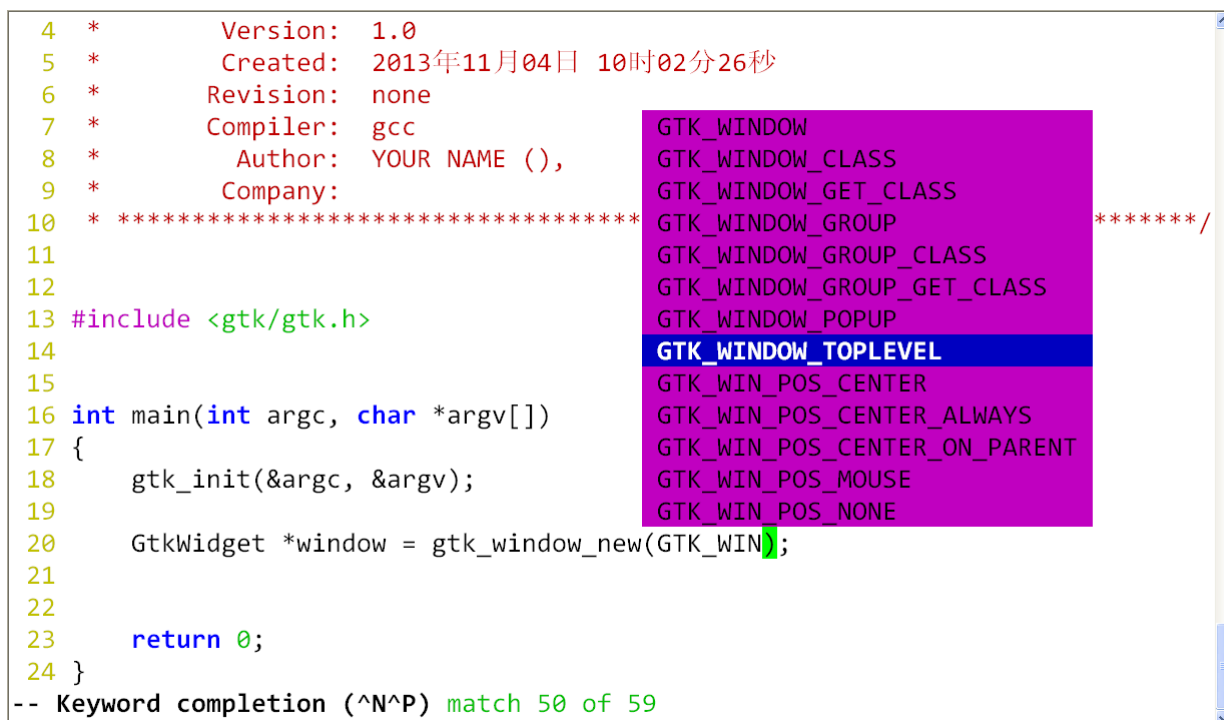
```

1 /* GtkWidget *****
2 * GtkWidgetAuxInfo
3 * GtkWidgetClass
4 * GtkWidgetClassPrivate
5 * GtkWidgetFlags 4日 10时02分26秒
6 * GtkWidgetHelpType
7 * GtkWidgetPath
8 * GtkWidgetPrivate ),
9 * GtkWidgetShapeInfo
10 * GtkWidget ***** /
11 GtkWidgetClass
12 GtkWidgetGeometryInfo
13 #in GtkWidgetGroup
14 GtkWidgetGroupClass
15 GtkWidgetGroupPrivate
16 int GtkWidgetKeysForeachFunc )
17 { GtkWidgetPosition
18     GtkWidgetPrivate
19     GtkWi
20     return 0;
21 }
-- Keyword completion (^N^P) match 1 of 19

```

### 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



```
4 *      Version:  1.0
5 *      Created:  2013年11月04日 10时02分26秒
6 *      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 {
18     gtk_init(&argc, &argv);
19
20     GtkWidget *window = gtk_window_new(GTK_WIN);
21
22
23     return 0;
24 }
-- Keyword completion (^N^P) match 50 of 59
```

The screenshot shows a list of GTK-related constants in a completion menu. The list includes: GTK\_WINDOW, GTK\_WINDOW\_CLASS, GTK\_WINDOW\_GET\_CLASS, GTK\_WINDOW\_GROUP, GTK\_WINDOW\_GROUP\_CLASS, GTK\_WINDOW\_GROUP\_GET\_CLASS, GTK\_WINDOW\_POPUP, **GTK\_WINDOW\_TOPLEVEL** (highlighted), GTK\_WIN\_POS\_CENTER, GTK\_WIN\_POS\_CENTER\_ALWAYS, GTK\_WIN\_POS\_CENTER\_ON\_PARENT, GTK\_WIN\_POS\_MOUSE, and GTK\_WIN\_POS\_NONE.

### 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

```

4 *      Version:  1.0
5 *      Created:  2013年11月04日 10时02分26秒
6 *      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 {
18     gtk_init(&argc, &
19     GtkWidget *window
20     GtkWidget *button
21     gtk_container_add(GTK_CON
22
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
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 {
18     gtk_init(&argc, &argv);
19
20     GtkWidget *window = gtk_w
21     GtkWidget *button = gtk_b
22
23     gtk_container_add(GTK_CON
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

```

4 *      Version:  1.0
5 *      Created:  2013年11月04日
6 *      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 {
18     gtk_init(&argc, &argv);
19
20     GtkWidget *window = gtk_window_
21     GtkWidget *button = gtk_button_
22
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

```

6 *      Revision: none
7 *      Compiler: gcc
8 *      Author:   YOUR NAME (),
9 *      Company:
10 * *****/
11
12
13 #include <gtk/gtk.h>
14
15 void on
16
17 int on_click_button
18 {
19     on_select_clist_row
20     on_select_row_tree_view
21     on1
22     on2
23     on3
24     onColorNdx
25     onConnection
26     onOff
27     on_color_ndx
28
-- 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:



```
15 void on_click_button(GtkButton *button, gpointer user_pointer)
16 {
17     g_print("hello\n");
18 }
19 ctrl+o(t)
20 int main(int argc, char *argv[])
21 {
22     gtk_init(&argc, &argv); ctrl+]
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
```

"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 {
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 [[
```

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 {
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
```

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 {
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
```

search hit BOTTOM, continuing at TOP

15,33

Bot