第5章 按钮构件

5.1 普通按钮GtkButton

GtkButton(普通按钮构件)是应用程序中使用最广泛的构件。它一般用于当用户点击它时执行某个动作。因而，如果某个构件能够对点击进行响应，实际上可以将它用作按钮。按钮构件的创建和使用相当简单。

前面已经介绍过， GtkButton是从GtkBin派生而来的构件，它本身就是一个容器。因而，可以充分利用这一点来创建非常丰富的按钮类型。比如在按钮上放置一幅图片、动画等。甚至可以创建非常古怪的按钮。

按钮构件也是一个基类， GtkToggleButton（开关按钮）、GtkCheckBtton（检查按钮）等构件都是从GtkButton派生而来的。

有两种创建按钮的方法。可以用 gtk_button_new_with_label()创建带标题的按钮，或用gtk_button_new()创建空白按钮，然后可以将标签和 pixmap图片组装到新按钮中。要这么做，需要先创建一个新的GtkBox，并用常用的gtk_box_pack_start()函数将对象（文本或图片）组装到这个GtkBox里，然后用 gtk_container_add函数将box组装到按钮里。

下面是用 gtk_button_new创建一个带图片和标题的按钮的例子。此处将创建一个 GtkBox，且放置图片和标题的部分和其余部分分开了。可以将这段代码用在程序的其他地方。关于pixmaps的用法在本教程的后面有更进一步的例子 。

```
/* 按钮示例开始buttons.c */

#include <gtk/gtk.h>

/* 创建一个新的GtkHBox组装盒，其中带一个标签和一幅图片
 * 然后返回这个组装盒 */

GtkWidget *xpm_label_box( GtkWidget *parent,
                          gchar     *xpm_filename,
                          gchar     *label_text )
{
    GtkWidget *box1;
    GtkWidget *label;
    GtkWidget *pixmapwid;
    GdkPixmap *pixmap;
    GdkBitmap *mask;
    GtkStyle *style;

    /* 为xpm图片和标签创建组装盒*/
    box1 = gtk_hbox_new (FALSE, 0);
    gtk_container_set_border_width (GTK_CONTAINER (box1), 2);
```

```
    /* 从父构件中取得风格参数以设置组装盒的背景颜色等风格 */
    style = gtk_widget_get_style(parent);

    /* 下载获得要填充的xpm图片*/
    pixmap = gdk_pixmap_create_from_xpm (parent->window, &mask,
                                    &style->bg[GTK_STATE_NORMAL],
                                      xpm_filename);
    pixmapwid = gtk_pixmap_new (pixmap, mask);

    /*为按钮创建一个标签 */
    label = gtk_label_new (label_text);

    /* 将标签和图片组装到GtkHBox中*/
    gtk_box_pack_start (GTK_BOX (box1),
                        pixmapwid, FALSE, FALSE, 3);

    gtk_box_pack_start (GTK_BOX (box1), label, FALSE, FALSE, 3);
    gtk_widget_show(pixmapwid);
    gtk_widget_show(label);

    return(box1);
}

/* 常见的回调函数 */
void callback( GtkWidget *widget,
               gpointer   data )
{
    g_print ("Hello again - %s was pressed\n", (char *) data);
}

int main( int   argc,
          char *argv[] )
{
    /* 所有构件的存储类型都是GtkWidget */
    GtkWidget *window;
    GtkWidget *button;
    GtkWidget *box1;

    gtk_init (&argc, &argv);

    /* 创建一个新窗口 */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    /* 设置窗口的标题 */
    gtk_window_set_title (GTK_WINDOW (window), "Pixmap'd Buttons!");

    /* 最好对所有的窗口都做下面的工作 */
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC (gtk_exit), NULL);
    gtk_signal_connect (GTK_OBJECT (window), "delete_event",
                        GTK_SIGNAL_FUNC (gtk_exit), NULL);
    /* 设置窗口的边框宽度 */
```

```
    gtk_container_set_border_width (GTK_CONTAINER (window), 10);
    gtk_widget_realize(window);

    /* 创建一个新按钮 */
    button = gtk_button_new ();

    /* 将按钮的"clicked"信号连接到前面创建的回调函数上 */
    gtk_signal_connect (GTK_OBJECT (button), "clicked",
                        GTK_SIGNAL_FUNC (callback),
                        (gpointer) "cool button");

    /* 调用我们定义的组装盒创建函数以创建一个带图片和标签的盒子 */
    box1 = xpm_label_box(window, "info.xpm", "cool button");

    /* 组装所有的构件。并显示它们 */
    gtk_widget_show(box1);
    gtk_container_add (GTK_CONTAINER (button), box1);
    gtk_widget_show(button);
    gtk_container_add (GTK_CONTAINER (window), button);
    gtk_widget_show (window);
    /* 开始程序的主循环，等待用户的动作 */
    gtk_main ();
    return(0);
}
/*示例结束 */
```

将上面的源代码保存为button.c文件，然后写一个Makefile文件，如下所示：

```
CC = gcc
buttons: buttons.c
  $(CC) `gtk-config --cflags`  buttons.c -o buttons `gtk-config --libs`
clean:
  rm -f *.o buttons
```

在shell提示符下输入make命令，编译该示例，然后输入./button就可执行这个程序。当点击图5-1所示的cool button按钮时，会在终端上显示“ Hello again Coll Button was pressed ”信息。

上面的示例中使用了一个函数xpm_label_box()，它将一幅图片和一个标签构件放到一个 GtkBox里面，这样可以创建一个带图片和标签的按钮。这个函数也可以用在其他容器构件上。

图5-1 按钮构件

注意，要了解在xpm_label_box函数中是怎么调用gtk_widget_get_style的。每个构件都有自己的“ 风格 ”，包括各种情形下的前景色和背景颜色、字体以及其他与构件有关的图像数据。这些风格在每个构件中都有默认值，并且许多 GDK函数调用都需要这些值。比如 gdk_pixmap_create_from_xpm函数，在这里给定了“ 正常 ”的背景色。

还要注意到：设置窗口的边框宽度后，调用了 gtk_widget_realize函数。这个函数用GDK创建与构件相关的X窗口。当一个构件调用gtk_widget_show()函数显示该构件时，会自动调用这个函数。因而前面的例子里面没有使用这个函数。但是调用 gdk_pixmap_create_from_xpm时需要它的窗口参数指向一个实际存在的 X窗口，因而在使用这个 GDK调用前必须先调用

gtk_widget_realize函数显现该构件。

按钮构件有下列信号：

pressed 当鼠标按键在按钮构件内按下时引发。

released 当鼠标按键在按钮构件内部松开时引发。

clicked 当鼠标按键在按钮构件上按下并松开时引发。

enter 当鼠标按键进入按钮构件时引发。

leave 当鼠标按键离开按钮构件时引发。

有时候，用户没有点击按钮，但是需要执行点击按钮时所对应的动作。一种方法是直接调用按钮的clicked信号的回调函数，另一种方法就是调用一个函数，让这个函数引发 clicked信号，这样自然就会调用这个回调函数。对鼠标按下、松开、进入和离开等信号也有类似的函数。

对给定按钮构件button引发pressed信号，效果是直接调用pressed信号对应的回调函数：

```
void  gtk_button_pressed (GtkButton *button);
```

对给定按钮构件button引发released信号：

```
void  gtk_button_released (GtkButton *button);
```

对给定按钮构件button引发clicked信号：

```
void  gtk_button_clicked  (GtkButton *button);
```

对给定按钮构件button引发enter信号：

```
void  gtk_button_enter  (GtkButton *button);
```

对给定按钮构件button引发leave信号：

```
void  gtk_button_leave  (GtkButton *button);
```

如果想让按钮构件不能接受点击，也就是不对用户响应，可以将构件设置为“不敏感”的。这在许多情况下都是很有作用的。比如，应用程序必须按照某个流程执行，否则就会出错。这时不应该寄希望于用户会按照正确的步骤执行，而是应该让用户根本就不能点击它。先将按钮设置为“不敏感”，然后当满足特定条件时，再将按钮设置为“敏感”的。

下面的函数设置构件的敏感性。将sensitive设置为TRUE，按钮可以接受用户点击，否则，按钮是灰色的，不能对用户的点击响应。

```
void   gtk_widget_set_sensitive        (GtkWidget *widget,
                                        gboolean sensitive);
```

使用这个函数可以设置其他构件的敏感性。

5.2 开关按钮 GtkToggleButton

GtkToggleButton(开关按钮构件)是从普通按钮里派生出来的。它们的状态总是处于两种状态中的一种：按下或弹起。其他方面和普通按钮非常相似。它的状态可以是被按下，当你再次点击时，将向上弹起。再点击一次，它又会被按下去。

开关按钮是检查按钮(check button)和无线按钮(radio button)的基础。与此类似，无线和检查按钮继承了开关按钮的许多函数调用。

创建一个新GtkToggleButton：

```
GtkWidget *gtk_toggle_button_new(void);
```

```
GtkWidget *gtk_toggle_button_new_with_label( gchar *label );
```

这两个函数与GtkButton的函数工作原理是一样的。第一个函数创建空的开关按钮，第二个函数创建一个带标签的开关按钮。

可以使用下例所示的代码取得开关按钮 (包括无线和检查按钮)的状态。这个函数先使用GTK_TOGGLE_BUTTON宏将构件转换为指向开关按钮的指针，然后访问开关按钮的结构的active域的值，以检测开关按钮的状态。对开关按钮（包含它的派生构件检查按钮和无线按钮），我们最感兴趣的信号是 toggled。要检查这些按钮的状态，设置一个信号处理函数以捕获toggled信号，然后访问按钮结构中的 active域以判定它的状态值。“ toggled ”信号的回调函数应该是下面这个样子：

```
void toggle_button_callback (GtkWidget *widget, gpointer data)
{
    if (GTK_TOGGLE_BUTTON (widget)->active)
    {
        /* 如果执行这段程序，表明按钮是按下的 */
    } else {
        /* 如果执行这段程序，按钮是弹起的 */
    }
}
```

要强行设置一个开关按钮（以及它的派生构件：无线按钮和检查按钮）的状态，使用下面的函数：

```
void gtk_toggle_button_set_active( GtkToggleButton *toggle_button,
                                   gint              state );
```

上面的函数适用于开关按钮以及检查按钮和无线按钮。将要设置的按钮作为第一个参数，第二个参数设置为 TRUE或FALSE，明确指定它是按下（压下或选中）或弹起（未选中）的。缺省是向上或FALSE。

注意，如果调用 gtk_toggle_button_set_active()函数，并且按钮的状态发生了实际变化，按钮会引发一个clieked信号。

```
void gtk_toggle_button_toggled (GtkToggleButton *toggle_button);
```

这个函数简单切换按钮的状态，并引发一个 toggled信号。

5.3 检查按钮 GtkCheckButton

GtkCheckButton(检查按钮构件)从上面介绍的开关按钮继承了许多属性和函数，但是外观上略有不同。检查按钮的文本标签是在按钮的旁边，而开关按钮的则在按钮里面。检查按钮经常用于在应用程序中切换“ 开启 ”或者“ 关闭 ”选项。

创建GtkCheckButton的函数如下：

```
GtkWidget *gtk_check_button_new(void );
GtkWidget *gtk_check_button_new_with_label ( gchar * label );
```

第二个以new_with_label结尾的函数创建一个带标签的检查按钮。

获取检查按钮的状态的方法与前述的开关按钮完全一样。

5.4 无线按钮 GtkRadioButton

GtkRadioButton(无线按钮构件)与检查按钮类似，不同之处在于无线按钮是分组的，同一

组内的按钮一次只能有一个被选中，也就是说它们是互斥的。当需要从一个较小的选项列表中选择一项时，使用无线按钮非常合适。

用下面的函数创建新的GtkRadioButton：

```
GtkWidget *gtk_radio_button_new( GSList *group );

GtkWidget *gtk_radio_button_new_with_label( GSList *group,
                                            gchar  *label );
```

可以看到，上面的两个函数都有一个 group参数。因为无线按钮必须分组，所以必须指定按钮属于哪一个组以便它们能够正常工作。

第一次调用gtk_radio_button_new_with_label或gtk_radio_button_new_with_label函数时应该给group参数传递NULL值，然后用下面的函数创建一个组：

```
GSList *gtk_radio_button_group( GtkRadioButton *radio_button );
```

要注意，必须对每个新建的无线按钮调用 gtk_radio_button_group函数，并将其加到按钮组中，其中的参数就是要加入的按钮。然后将其结果传递到下一个调用 gtk_radio_button_new或gtk_radio_button_new_with_label的函数中。这样可以创建一系列的按钮。下面的示例详细介绍了这些内容。可以用下面的语法缩短这个过程。这种语法不需要维护一个按钮列表。在示例中用这种形式创建了第三个无线按钮：

```
button2 = gtk_radio_button_new_with_label(
              gtk_radio_button_group (GTK_RADIO_BUTTON (button1)),
               "button2");
```

最好在创建按钮时明确指出哪一个按钮是缺省选中的按钮。用下面的方法：

```
void gtk_toggle_button_set_active( GtkToggleButton *toggle_button,
                                   gint             state );
```

这个函数在GtkToggleButton中已经介绍过了，在这里按同样的方法使用。一旦无线按钮已经分组，一组按钮中就只能有一个处于活动状态（被选中）。如果用户点击某个无线按钮，然后点击同组的另一个无线按钮，前一个无线按钮会首先引发一个 toggled信号（报告它变成不活动的），然后第二个按钮会引发一个toggled信号（报告它变成活动按钮）。

使用上面的方法，我们能够在窗口上添加几个无线按钮，将它们分为几组，让同组之间是互斥的，不同组之间互不相干。最好能用 GtkFrame或者类似的构件将它们从布局上分开。否则，用户可能会误操作。

下列例子将创建一个含有三个无线按钮的按钮组。

```
/* 无线示例开始 radiobuttons.c */
#include <gtk/gtk.h>
#include <glib.h>
void close_application( GtkWidget *widget,
                        GdkEvent  *event,
                        gpointer   data )
{
gtk_main_quit();
}
int main( int   argc,           char *argv[] )
{   GtkWidget *window = NULL;
    GtkWidget *box1;
```

```
GtkWidget *box2;
GtkWidget *button;
GtkWidget *separator;
GSList *group;

gtk_init(&argc,&argv);
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);

gtk_signal_connect (GTK_OBJECT (window), "delete_event",
                    GTK_SIGNAL_FUNC(close_application),
                    NULL);
gtk_window_set_title (GTK_WINDOW (window), "radio buttons");
gtk_container_set_border_width (GTK_CONTAINER (window), 0);

box1 = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (window), box1);
                   gtk_widget_show (box1);
box2 = gtk_vbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_box_pack_start (GTK_BOX (box1), box2, TRUE, TRUE, 0);
gtk_widget_show (box2);

button = gtk_radio_button_new_with_label (NULL, "button1");
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);
group = gtk_radio_button_group (GTK_RADIO_BUTTON (button));
button = gtk_radio_button_new_with_label(group, "button2");
gtk_toggle_button_set_active (GTK_TOGGLE_BUTTON (button), TRUE);
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);
button = gtk_radio_button_new_with_label
            (gtk_radio_button_group (GTK_RADIO_BUTTON (button)),
            "button3");

gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
gtk_widget_show (button);
separator = gtk_hseparator_new ();
gtk_box_pack_start (GTK_BOX (box1), separator, FALSE, TRUE, 0);
gtk_widget_show (separator);
box2 = gtk_vbox_new (FALSE, 10);
gtk_container_set_border_width (GTK_CONTAINER (box2), 10);
gtk_box_pack_start (GTK_BOX (box1), box2, FALSE, TRUE, 0);
gtk_widget_show (box2);
button = gtk_button_new_with_label ("close");
gtk_signal_connect_object (GTK_OBJECT (button), "clicked",
                           GTK_SIGNAL_FUNC(close_application),
                           GTK_OBJECT (window));
gtk_box_pack_start (GTK_BOX (box2), button, TRUE, TRUE, 0);
GTK_WIDGET_SET_FLAGS (button, GTK_CAN_DEFAULT);
gtk_widget_grab_default (button);
```

```
    gtk_widget_show (button);
    gtk_widget_show (window);
    gtk_main();
    return(0);
}
/* 示例结束*/
```

将上面的代码保存为 radiobuttons.c，然后写一个像下面这样的Makefile文件。在shell提示符下输入make命令进行编译。

```
CC = gcc
radiobuttons: radiobuttons.c
   $(CC) `gtk-config --cflags`  radiobuttons.c -oradiobuttons `\
         gtk-config --libs`
clean:
   rm -f *.o radiobuttons
```

运行结果如图5-2所示。在button1、button2和button3上点击，看一下它们的状态有什么变化。这几个按钮应该是互斥的。

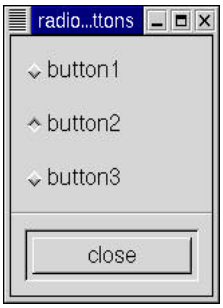

图5-2 无线按钮示例