LinuxFocus article number 332
http://linuxfocus.org

# GUI Programming with GTK - 3

by Özcan Güngör
<ozcangungor(at)netscape.net>

*About the author:*
I'm doing at the moment my military service as Linux, Oracle administrator and web programmer.

*Abstract*:

In these article series, we will learn how to write graphical user interfaces (GUIs) using GTK. I do not have any idea how long it will last. In order to understand these articles, you should know the following about the C programming language:

- Variables
- Functions
- Pointers

It is recommended to read the previous articles:
GUI Programming with GTK,
GUI Programming with GTK - 2 .
This article is a little bit shorter than the others as I'm doing my military service.

## Toggle Button

This button looks like a normal button but has two states: Pressed or not. To create a toogle button one of the following function is used:

```
GtkWidget *toggle=gtk_toggle_button_new(void);
GtkWidget *toggle=gtk_toggle_button_new_with_label(const gchar *label);
```

The first function creates a toggle button without a text label in it but the second one creates it with the string *label* on it.

You can set its state with the following function:

```
gtk_toggle_button_set_active (GtkToggleButton *toggle_button,
                              gboolean is_active);
```

where *toggle_button* is the button whose state you want to change and *is_active* is the state (0 or 1). When it is 0, then the button is not in "pressed" state; when it is 1, the button is in "pressed" state.

To get the state of the button, the following function can be used:

```
gboolean gtk_toggle_button_get_active(GtkToggleButton *button);
```

The "toggled" event can be connected to a toggle button.

Here is an example:

```
#include <gtk/gtk.h>
void togg(GtkWidget *widget, gpointer *data){
   if (gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(data)))
    g_print("State is 1\n");
   else
    g_print("State is 0\n");
}

int main( int argc,char *argv[] )
   {

   GtkWidget *window;
   GtkWidget *button;

   gtk_init (&argc, &argv);

   /* Create a new window */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title (GTK_WINDOW (window), "Toggle Button");

   /* Connect destroy event to the window. */
   gtk_signal_connect (GTK_OBJECT (window), "destroy",
                       GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

   /* Creates a toggle button */
   button=gtk_toggle_button_new_with_label("I'm a toggle button");

   /* Add the button to window */
   gtk_container_add(GTK_CONTAINER(window),button);

   /* Connect "toggled" event to the button */
   gtk_signal_connect (GTK_OBJECT (button), "toggled",
                       GTK_SIGNAL_FUNC(togg),(gpointer *)button);

   gtk_widget_show(button);
   gtk_widget_show (window);

   gtk_main ();
   return(0);
}
```

## Check Button

The check button ( it is also known as check box ) is a subclass of the toggle button. It can be used to select some options.
To create a check button, the following fuction is used:

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

The explanations are the same as for toggle button.

The example :

```
#include <gtk/gtk.h>
void togg(GtkWidget *widget, gpointer *data){
   if (gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(data)))
       g_print("State is 1\n");
   else
    g_print("State is 0\n");
}

int main( int argc,char *argv[] )
{

    GtkWidget *window;
    GtkWidget *button;

    gtk_init (&argc, &argv);

    /* Create a new window */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Check Button");

    /* Connect destroy event to the window. */
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

    /* Creates a check button */
    button=gtk_check_button_new_with_label("I'm a check button");

    /* Add the button to window */
    gtk_container_add(GTK_CONTAINER(window),button);

    /* Connect "toggled" event to the button */
    gtk_signal_connect (GTK_OBJECT (button), "toggled",
                        GTK_SIGNAL_FUNC(togg), (gpointer *)button);
    gtk_widget_show(button);
    gtk_widget_show (window);

    gtk_main ();
    return(0);
}
```

## Label

Labels let you put any text anywhere in window.
To create a label just use the following function:

```
GtkWidget* gtk_label_new(const gchar *text);
```

With the function

```
gtk_label_set_text(GtkLabel *label, gchar *text);
```

you can change the string on a label at anytime.

```
gtk_label_set_justify(GtkLabel *label, GtkJustification jtype);
```

The gtk_label_set_justify funtion is used to justify the text on the label. *jtype* can be

- GTK_JUSTIFY_LEFT to put the text to the left,
- GTK_JUSTIFY_RIGHT to put the text to the rigth,
- GTK_JUSTIFY_CENTER to put the text in the center,
- GTK_JUSTIFY_FILL to make the text cover the whole label.

```
gtk_label_set_line_wrap (GtkLabel *label,gboolean wrap);
```

is used to make the text splittable into many pieces when the text is longer than the place it should fit. When *wrap* is 1, then text will *wrap* to the next line, otherwise not.

```
gtk_label_get(GtkLabel *label, gchar **str)
```

is used to get the text on the label into *str*.

## ToolTips

Tooltip is a text that appears when the mouse is on a widget. For example, you can set a tip for a button and the text "send the information" appears when your mouse is over it.

To do that a GtkToolTips widget must be created first:

```
GtkToolTips* gtk_tooltips_new();
```

Then this tooltip is attached to a widget:

```
gtk_tooltips_set_tip(GtkTooltips *tooltips, GtkWidget *widget,
                     const gchar *tip_text,const gchar *tip_private);
```

A little example:

```
#include <gtk/gtk.h>
int main( int argc,char *argv[] )
{
    GtkWidget *window;
    GtkWidget *button;
```

```
    GtkTooltips *tip;

    gtk_init (&argc, &argv);

    /* Create a new window */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Tooltips");

    /* Connect destroy event to the window. */
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC (gtk_main_quit), NULL);

    /* Creates a button */
    button=gtk_button_new_with_label("I'm a Button");

    /* Add the button to window */
    gtk_container_add(GTK_CONTAINER(window),button);

    /* Creates a tooltips*/
    tip=gtk_tooltips_new();

    /* Attache this tooltips to button with text*/
    gtk_tooltips_set_tip(tip, button, "Click me!",NULL);

    gtk_widget_show(button);
    gtk_widget_show (window);

    gtk_main ();
    return(0);
}
```

Some other functions:

```
gtk_tooltips_enable (GtkTooltips *tooltips);
```

Enables the tooltips.

```
gtk_tooltips_disable (GtkTooltips *tooltips);
```

Disables tooltips.

To get any tooltip data of a wigdet, we need

```
GtkTooltipsData* gtk_tooltips_get_data(GtkWidget *widget);
```

GtkTooltipsData is a struct in the following way:

```
struct _GtkTooltipsData
{
   GtkTooltips *tooltips;
   GtkWidget *widget;
   gchar *tip_text;
   gchar *tip_private;
   GdkFont *font;
   gint width;
   GList *row;
};
```

To set the delay of the appearing text,

```
gtk_tooltips_set_delay (GtkTooltips *tip, guint delay)
```

## Combo Box

A combo box is an editable text field combined with a pull-down menu. You can enter a value or select one of the pull-down entries.

A combo box can be created with

```
GtkWidget *gtk_combo_new();
```

And we need a list of options which is a GList struct.

```
GList *glist=NULL;
```

And the options can be appended to the list with

```
GList *g_list_append(GList *list, gchar *option);
```

Then this list needs to be added to the combo box

```
gtk_combo_set_popdown_strings(GtkCombo *combo, GList *List);
```

The combo box is ready. To read the selected option use:

```
gchar *gtk_entry_get_text(GtkEntry *entry);
```

*entry* is GTK_ENTRY(GTK_COMBO(combo)->entry)) in this case.

```
gtk_combo_set_use_arrows(GtkCombo *combo,gint val);
```

This function is used to enable or disable up/down keys on the keyboard to change the value on a combo box. When *val* is 0, these keys do not function, otherwise these keys change the value. But when the value on a combo box is different from the values in the list, **these keys don't function**.

```
gtk_combo_set_use_arrows_always(GtkCombo *combo,gint val);
```

This function is the same as *gtk_combo_set_use_arrows* but when the value on a combo box is different form the values in the list, **these keys function**.

```
gtk_combo_set_value_in_list(GtkCombo *combo, gboolean val,
                            gboolean ok_if_empty);
```

When *val* is 1, you can enter a value in the list. When *ok_if_empty* is 1, the value may be blank.

```
#include <gtk/gtk.h>
void act(GtkWidget *widget, gpointer *data){
   g_print((gchar *)data);
```

```
}

int main( int argc,char *argv[] ) {
   GtkWidget *window;
   GtkWidget *combo;
   GtkWidget *button;
   GtkWidget *box;
   GList *list=NULL;

   list=g_list_append(list,"Slackware");
   list=g_list_append(list,"RedHat");
   list=g_list_append(list,"SuSE");

   gtk_init (&argc, &argv);

   /* Create a new window */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title (GTK_WINDOW (window), "Combo Box");

   /* Connect destroy event to the window. */
   gtk_signal_connect (GTK_OBJECT (window), "destroy",
                       GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

   /* Create a new horizontal box */
   box=gtk_hbox_new(1,0);
   gtk_container_add(GTK_CONTAINER(window),box);

   /* Creates a combo box */
   combo=gtk_combo_new();

   /* Sets the list */
   gtk_combo_set_popdown_strings(GTK_COMBO(combo),list);

   /* Enables up/down keys change the value. */
   gtk_combo_set_use_arrows_always(GTK_COMBO(combo),1);

   gtk_box_pack_start(GTK_BOX(box), combo,1,1,1);

   button=gtk_button_new_with_label("Write it");
   gtk_signal_connect(GTK_OBJECT(button),"clicked",GTK_SIGNAL_FUNC(act),
             gtk_entry_get_text(GTK_ENTRY(GTK_COMBO(combo)->entry)));
   gtk_box_pack_start(GTK_BOX(box), button,1,1,1);

   gtk_widget_show(box);
   gtk_widget_show(combo);
   gtk_widget_show(button);
   gtk_widget_show (window);

   gtk_main ();
   return(0);
   }
```

All comments are welcome.

2005-01-14, generated by lfparser_pdf version 2.51