LinuxFocus article number 295
http://linuxfocus.org

GUI Programming with GTK

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

About the author:
I use Linux since 1997.Freedom, flexibility and opensource. These are the properties I like.

Translated to English by:
Özcan Güngör
<ozcangungor(at)netscape.net>

Abstract:

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

- Variables
- Functions
- Pointers

What is GTK?

GTK (GIMP Toolkit) is a library for creating Graphical User Interfaces. The library is available under the GPL license. Using this library, you can create open-source, free or commercial programs.

The library has the name GIMP toolkit (GTK) because it was originally created for developing GIMP (General Image Manipulation Program). The authors of GTK are:

- Peter Mattis
- Spencer Kimball
- Josh MacDonald

GTK is object-oriented application user interface. Although written in C, it uses idea of classes and

callback functions.

Compiling

In oreder to compile GTK programs, you need to tell gcc what GTK libraries are and where they are. The *gtk-config* command is "knows" this.

gtk-config --cflags --libs

The output of this command is something like following (depending on the system):

-I/opt/gnome/include/gtk-1.2 -I/opt/gnome/include/glib-1.2 -I/opt/gnome/lib/glib /include -I/usr/X11R6/include -L/opt/gnome/lib -L/usr/X11R6/lib -lgtk -lgdk -rdynamic -lgmodule -lglib -ldl -l Xext -lX11 -lm

Explanations of these parameters are:

-l library: Searches for a library in the form like lib*library*.a in defined paths.
-L path: Adds a path to search libraries.
-I path: Adds a path to search header file used in program.

To compile a GTK program named hello.c, following command can be used:

gcc -o hello hello.c ‘gtk-config --cflags --libs‘

The input used after -o parameter is the name of compiled program.

A First Program

It is assumed that GTK is installed on your system. The lastest versions of GTk can found at ftp.gtk.org.

Let’s write our first program. This program creates a 200x200 pixel wide, empty window.

```
#include <gtk/gtk.h>

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

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_widget_show  (window);

    gtk_main ();
```

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

GtkWidget is a variable type to define various components like window, button, label... In this example, a window is defined like the following:

GtkWidget *window;

void gtk_init(int *argc,char ***argv) initiates the toolkit and gets the parameters entered in command line. This function must be used after definening components.

GtkWidget *gtk_window_new(GtkWindowType windowtype) creates a new window. Window type can be:

- GTK_WINDOW_TOPLEVEL
- GTK_WINDOW_DIALOG
- GTK_WINDOW_POPUP

void gtk_widget_show(GtkWidget *widget) is used to make the component appear in a window. After defining a component and changing attributes, this function must be used.

void gtk_main(void) prepares windows and all components to appear in the screen. This function must be used at the end of GTK programs.

Let’s use some properties of window such as titles, size, position...

void gtk_window_set_title(GtkWindow *window,const gchar *title) is used to set or change the title of *window*. First parameter of this function is in GtkWindow type. But *window* variable is in GtkWidget type. While compiling, we will be warned about it. Atlhough compiled program works, it is better to correct it. GTK_WINDOW(GtkWidget *widget) is used for that. Second parameter *title* is in gchar type. gchar is defined in glib library and the same as char type.

void gtk_window_set_default_size(GtkWindow *window, gint width, gint height) sets the size of *window*. Like gchar, gint is defined in glib and the same as int.

The function

void gtk_window_set_position(GtkWindow *window, GtkWindowPosition position)

sets the position of *window.position* can be:

- GTK_WIN_POS_NONE
- GTK_WIN_POS_CENTER
- GTK_WIN_POS_MOUSE
- GTK_WIN_POS_CENTER_ALWAYS

Here is an example:

```
#include <gtk/gtk.h>
```

```
int main( int   argc,
          char *argv[] )
{
    GtkWidget *window;

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window),"Ýlk Program");
    gtk_window_set_position(GTK_WINDOW(window),GTK_WIN_POS_CENTER);
    gtk_window_set_default_size(GTK_WINDOW(window),300,300);
    gtk_widget_show  (window);

    gtk_main ();

    return(0);
}
```

Signals and Events

In GUIs, you need to use mouse and keyboard, ie. you can click on a button. For that, the following GTK function is used:

guint gtk_signal_connect_object(GtkObject *object,const gchar *name,GtkSignalFu nc func,GtkObject *slot_object);

object is the component that emits signals. For example, if you want to know if a button is clicked, *object* will be button.*name* is the name of event and can be :

- event
- button_press_event
- button_release_event
- motion_notify_event
- delete_event
- destroy_event
- expose_event
- key_press_event
- key_release_event
- enter_notify_event
- leave_notify_event
- configure_event
- focus_in_event
- focus_out_event
- map_event
- unmap_event
- property_notify_event
- selection_clear_event
- selection_request_event
- selection_notify_event

- proximity_in_event
- proximity_out_event
- drag_begin_event
- drag_request_event
- drag_end_event
- drop_enter_event
- drop_leave_event
- drop_data_available_event
- other_event

func is the name of function that will be called when the event occurs. Here is an example:

```
#include <gtk/gtk.h>

void close( GtkWidget *widget,gpointer *data)
{
    gtk_main_quit();
}

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

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC (close), NULL);
    gtk_widget_show  (window);

    gtk_main ();

    return(0);
}
```

The function

gtk_signal_connect (GTK_OBJECT (window), "destroy",GTK_SIGNAL_FUNC (close), NULL)

listens to the window's destroy event. When window is attempted to close, then the *close* function is called. The *close* function calls gtk_main_quit() and the program ends.

Details about signals and event will be explained later...

A ordinary button

Normal buttons, are usually used to do certain things when the button is clicked. In GTK library, there are two ways of creating buttons:

1. GtkWidget* gtk_button_new (void);

2. GtkWidget* gtk_button_new_with_label (const gchar *label);

The first function creates a button without a label (nothing written on the button). The second one creates a button with label (*label* is written on button).

Here, we will use a new function:

void gtk_container_add(GtkContainer *container,GtkWidget *widget)

Using this function, it is possible to create a button (generally all componenets) appear on window (generally on a container). In the next example, the container is a window and the component to be added is button. We will learn about some other containers later.

The most important thing about a button is to know if it is clicked or not. Again, the gtk_signal_connect function is used for this purpose. With this function, an other function will be called and execute the functionallity "behind" the button. Here is an example:

```
#include <gtk/gtk.h>

void close( GtkWidget *widget,gpointer *data)
{
    gtk_main_quit();
}

void clicked(GtkWidget *widget,gpointer *data)
{
        g_print("Button Clicked\n");
}
int main( int   argc,char *argv[] )
{
    GtkWidget *window,*button;

    gtk_init (&argc, &argv);

    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC (close), NULL);

    button=gtk_button_new_with_label("Button");
    gtk_container_add(GTK_CONTAINER(window),button);
    gtk_signal_connect(GTK_OBJECT(button),"clicked",
                       GTK_SIGNAL_FUNC(clicked),NULL);
    gtk_widget_show(button);

    gtk_widget_show(window);

    gtk_main ();

    return(0);
}
```

2005-01-14, generated by lfparser_pdf version 2.51