LinuxFocus article number 295
http://linuxfocus.org

Programmer des interfaces graphiques avec GTK

par Özcan Güngör
<ozcangungor/at/netscape.net>

L´auteur:
J’utilise Linux depuis 1997. Liberté, flexibilité et opensource. Ce sont les propriétés que je préfère.

Traduit en Français par:
Frederic Nourry
<frederic.nourry/at/29laposte.net>

Résumé:

Dans cette série d’articles, nous allons voir comment écrire des programmes avec interfaces graphiques utilisateur (GUI) grâce à GTK. Je n’ai aucune idée du nombre d’articles qu’il faudra. Pour bien comprendre ces articles, vous devez posséder les notions suivantes de programmation en langage C:

- Variables
- Fonctions
- Pointeurs

Qu’est-ce que GTK?

GTK (GIMP Toolkit) est une bibliothèque pour créer des interfaces graphique utilisateur (Graphical User Interfaces ou GUI). Elle est disponible sous licence GPL. Grâce a cette bibliothèque, vous pouvez créer des programmes open-source, gratuits ou commerciaux.

La bibliothèque s’appelle GIMP toolkit (GTK) car elle a été créée à l’origine pour le developpement de GIMP (General Image Manipilation Program). Les auteurs de GTK sont:

- Peter Mattis
- Spencer Kimball

- Josh MacDonald

GTK est une bibliothèque orientée objet . Bien qu'écrite en C, elle utilise les notions de classes et de fontions callback.

Complilation

Afin de compiler des programmes GTK, vous devez dire au compilateur gcc ce que sont les bibliothèques GTK et où elles se trouvent. La commande *gtk-config* permet d'obtenir ces informations.

gtk-config --cflags --libs

Elle doit vous donner en sortie quelque chose du genre (selon votre système):

-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

Explication de ces paramètres:

-l library: Cherche une bibliothèque de la forme lib*library*.a dans les chemins définis.
-L path: Ajoute un chemin de recherche de bibliothèques.
-I path: Ajoute un chemin de recherche de fichiers d'entête utilisés par le programme.

Pour compiler un programme GTK nommé 'hello.c', utilisez la commande suivante:

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

Le paramètre après le '-o' est le nom du programme compilé.

Notre premier programme

Nous supposons que GTK est installé sur votre machine. La dernière version de GTK peut être obtenue à ftp.gtk.org.

Ecrivons notre premier programme. Il va créer une simple fenêtre vide de 200x200 pixels.

```
#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);
}
```

Le GtkWidget est un type de variable servant à déclarer divers composants graphiques comme une fenêtre, un bouton, une étiquette... Dans cet exemple, une fenêtre est déclarée par:

GtkWidget *window;

void gtk_init(int *argc,char ***argv) initialise le toolkit et récupère les paramètres entrés sur la ligne de commande. Cette fonction doit être utilisée après avoir déclaré les composants graphiques.

GtkWidget *gtk_window_new(GtkWindowType windowtype) crée une nouvelle fenêtre. Le type de fenêtre peut être:

- GTK_WINDOW_TOPLEVEL
- GTK_WINDOW_DIALOG
- GTK_WINDOW_POPUP

void gtk_widget_show(GtkWidget *widget) est utilisé pour faire apparaître un composant graphique dans la fenêtre. Apres avoir défini un composant et modifié ses attributs, cette fonction doit être utilisée.

void gtk_main(void) prépare les fenêtres et tous les composants à apparaître à l'écran. Cette fonction doit être appelée à la fin des programmes GTK.

Utilisons quelques propriétés de fenêtres comme le titre, la taille, la position...

void gtk_window_set_title(GtkWindow *window,const gchar *title) est utilisé pour définir ou changer le nom de *window*. Le premier paramètre de cette fonction se trouve dans le type GtkWindow, mais la variable *window* se trouve dans le type GtkWidget. Pendant la compilation, nous en serons avertis. Même si le programme compilé fonctionne correctement, il est préférable de le modifier par GTK_WINDOW(GtkWidget *widget). Le second paramètre *title* est de type gchar, défini dans la bibliothèque glib et il est équivalent au type char.

void gtk_window_set_default_size(GtkWindow *window, gint width, gint height) définit les dimensions de *window*. Comme gchar, gint est défini dans glib, et il est équivalent au type int.

La fonction

void gtk_window_set_position(GtkWindow *window, GtkWindowPosition position)

définit la position de *window*. La valeur de *position* peut être:

- GTK_WIN_POS_NONE
- GTK_WIN_POS_CENTER

- GTK_WIN_POS_MOUSE
- GTK_WIN_POS_CENTER_ALWAYS

Voici un exemple:

```
#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);
}
```

Signaux et événements

Avec les "GUI", vous devez utiliser la souris et le clavier, c'est-à-dire que vous pouvez cliquer sur un bouton. Pour ce faire, on utilise la fonction GTK suivante :

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

object est le composant graphique qui émet les signaux. Par exemple, vous voulez savoir si on a cliqué sur un bouton, *object* sera ce bouton. *name* est le nom de l'événement qui peut être :

- 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 est le nom de la fonction qui sera appelée lorsque l'événement aura lieu. Voici un exemple:

```
#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);
}
```

La fonction

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

attend l'événement "destroy" lié à la fenêtre. Lorsqu'on tente de fermer la fenêtre, la fonction *close* est appelée. Elle appelle gtk_main_quit() et le programme s'arrête.

Les détails sur les signaux et les événement seront abordés plus tard...

Un bouton simple

Les boutons normaux sont utilisés en général pour faire certaines choses quand on clique dessus. Avec la bibliothèque GTK, il y a deux façons de créer des boutons :

1. GtkWidget* gtk_button_new (void);
2. GtkWidget* gtk_button_new_with_label (const gchar *label);

La première fonction crée un bouton sans étiquette (pas de texte écrit sur le bouton). La deuxième crée un bouton avec une étiquette (*label* est écrit sur le bouton).

Ici, nous allons utiliser une nouvelle fonction:

void gtk_container_add(GtkContainer *container,GtkWidget *widget)

Avec cette fonction, il est possible d'ajouter un bouton (ou un composant graphique) dans une fenêtre (habituellement dans un conteneur). Dans le prochain exemple, le conteneur est une fenêtre et le composant à ajouter est un bouton. Nous verrons d'autres conteneurs plus tard.

La chose la plus importante pour un bouton est de savoir si on clique dessus ou non. Une nouvelle fois, on utilise la fonction gtk_signal_connect dans ce but. Grâce à cette fonction, une autre fonction pourra être appelée et exécutera le code associé au bouton. Voici un exemple:

```
#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