Programmer des interfaces graphiques avec GTK – 3

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

L´auteur:
J'effectue actuellement mon service militaire en tant qu'administrateur Linux, Oracle et programmeur web.

Traduit en Français par:
Iznogood
<iznogood%28at%29iznogood–factory.org>

Résumé:

Dans cette série d'articles; nous allons apprendre comment écrire des interfaces utilisateurs graphiques (GUIs) utilisant GTK. Je n'ai aucune idée du temps que cela va prendre. De manière à comprendre ces articles, vous devez connaître ce qui suit à propos de la programmation en C :

- Les Variables
- Les Fonctions
- Les Pointeurs

Il est recommandé de lire les articles suivants :
Programmer des interfaces graphiques avec GTK,
Programmer des interfaces graphiques avec GTK – 2 .
Ce article est un peu plus court que les autres car j'effectue mon service militaire.

Bouton à bascule

Ce bouton ressemble à un bouton normal mais a deux états : pressé ou non. Pour créer un bouton à bascule, une des fonctions suivantes est utilisée :

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

La première fonction crée un bouton à bascule sans texte à l'intérieur mais le second en crée un avec la chaîne *label* dedans.
Vous pouvez initialiser cet état avec la fonction suivante :

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

où *toggle_button* est le bouton dont vous voulez changer l'état et *is_active* est son état (0 ou 1). Lorsqu'il est à 0, alors le bouton n'est pas dans l'état «pressé»; lorsqu'il est à 1, le bouton est dans l'état «pressé».

Pour obtenir l'état du bouton, la fonction suivante peut être utilisée :

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

L'événement «basculé» peut être connecté à un bouton à bascule.

Vous avez ici un exemple :

```
#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);
}
```

Bouton de contrôle

Le bouton de contrôle (il est aussi connu comme bouton de validation) est une sous−classe du bouton à bascule. Il peut être utilisé pour sélectionner des options.
Pour créer un bouton de contrôle, la fonction suivante est utilisée :

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

Les explications sont les mêmes que pour le bouton à bascule.

L'exemple :

```
#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

Les Labels vous permettent de mettre du texte partout dans la fenêtre.
Pour créer un label, utilisez simplement la fonction suivante :

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

Avec la fonction

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

vous pouvez changer la chaîne sur un label à tout moment.

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

La fonction gtk_label_set_justify est utilisée pour justifier le texte sur le label. *jtype* peut être

- GTK_JUSTIFY_LEFT pour placer le texte à gauche,
- GTK_JUSTIFY_RIGHT pour placer le texte à droite,
- GTK_JUSTIFY_CENTER pour placer le texte au centre,
- GTK_JUSTIFY_FILL pour que le texte couvre tout le label.

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

est utilisé pour sectionner le texte en plusieurs pièces lorsque le texte est plus long que la zone dans laquelle il devrait rentrer. Lorsque *wrap* est à 1, alors le texte sera *coupé* vers la ligne suivante, sinon, il continuera.

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

est utilisé pour mettre le texte du label entre *str*.

ToolTips

Tooltip est un texte qui apparaît lorsque la souris est sur une widget. Par exemple, vous pouvez initialiser une information pour un bouton et le texte «envoyez l'information» apparaît lorsque la souris est dessus.

Pour ce faire, une widget GtkToolTips doit d'abord être créée :

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

Lorsque ce tooltip est attaché à une widget :

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

Un petit exemple :

```
#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();

    /* Attach 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);
}
```

Quelques autres fonctions :

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

Valide les tooltips.

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

Désactive les tooltips.

Pour obtenir les données tooltip d'une wigdet, nous avons besoin de

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

GtkTooltipsData est structuré de la manière suivante :

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

Pour ajuster le délai d'apparition du texte :

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

Boîte Combo

Une boîte combo est un champ de texte éditable associé avec un menu déroulant. Vous pouvez entrer une valeur ou en sélectionner une depuis les entrées déroulantes.

Une boîte combo peut être créée avec

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

Et nous avons besoin d'une liste d'options sous forme d'un struct GList.

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

Et les options peuvent être ajoutées à la liste avec

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

Alors la liste a besoin d'être ajoutée à la boîte combo

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

La boîte combo est prête : pour lire l'option sélectionnée, utilisez :

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

Pour récupérer la valeur entrée :

```
GTK_ENTRY(GTK_COMBO(combo)->entry))
```

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

Cette fonction est utilisée pour valider ou désactiver les touches haut/bas sur le clavier, pour changer la valeur d'une boîte combo. Lorsque *val* est à 0, ces touches ne fonctionnent pas; dans les autres cas, les touches changent la valeur. Mais lorsque la valeur d'une boîte combo est différente des valeurs de la liste, **ces touches ne fonctionnent pas**.

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

Cette fonction est la même que *gtk_combo_set_use_arrows* mais lorsque les valeurs de la boîte combo sont différentes des valeurs de la liste, **les touches fonctionnent**.

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

Lorsque *val* est à 1, vous pouvez entrer une valeur dans la liste. Lorsque *ok_if_empty* est à 1, la valeur peut être absente.

```
#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);
   }
```

Tous les commentaires sont les bienvenus.

2005−01−22, generated by lfparser_pdf version 2.51