Programmation d'interfaces

Cours 2 - Les signaux : Interagir avec les composants graphiques

H. Djerroud

LIASD - Université Paris 8

Automne 2020

Objectif

- Pour interagir avec les utilisateurs, chaque composant graphique possède un certain nombre de signaux....

Plan de cours

- Théorie des signaux de des Callbacks
- Les évènements
- Éviter les variables globales
- Gestion des signaux des widgets
- Dessiner avec Cairo
- **Distribution des projets et formation des binômes**

Théorie des signaux

- **Remarque :** Depuis la version *GTK 2.0*, le système de signaux a été déplacé de *GTK* vers *GLib*, donc les fonctions et types expliqués dans cette section ont un préfixe "g_" plutôt qu'un préfixe "gtk_".

Théorie des signaux

- *GTK* utilise le paradigme de la *programmation événementielle*, ce qui signifie que le programme dormira dans `gtk_main()` jusqu'à ce qu'un événement se produise. Dans ce cas le **contrôle est passé** à la fonction appropriée.
- Le passage de contrôle se fait en utilisant les **"signaux"**.
- Lorsqu'un événement se produit, comme la pression d'un bouton de souris, le signal approprié sera **"émis"** par le widget qui a été pressé.
- Pour effectuer une action correspondante à un événement (signal), il faut configurer le gestionnaire de signaux pour capturer ce signal et appeler la fonction appropriée

Les signaux dans GTK+

- Un signal : Évènement collecté par GTK (création, clic souris...) et transmis aux widgets.
- Les signaux sont identifiés par une chaîne de caractère : `activate`, `clicked`, ...
- Pour recevoir un signal, un widget connecte une fonction appelé *callback* :
  - Un fonction qui fait partie de votre programme
  - Signature bien précise
  - appelée par `gtk_main()` à réception d'un signal

Connecter un signal

- Pour connecter une callback :

```
gulong g_signal_connect( gpointer     *object,
                         const gchar   *name,
                         GCallback     func,
                         gpointer      user_data );
```

Les callbacks

- La fonction callback :

```
void callback_func( GtkWidget *widget,
                    ... /* other signal arguments */
                    gpointer   callback_user_data );
```

Premier programme avec GTK

```
#include <gtk/gtk.h>
int main(int argc, char* argv[]){
  GtkWidget *win, *b1;
  gtk_init(&argc, &argv);
  win = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  b1  = gtk_button_new_with_label("Hello");
  gtk_container_add(GTK_CONTAINER(win),b1);
  g_signal_connect (G_OBJECT(win),"delete_event",
                    G_CALLBACK(delete_event) ,NULL);
  g_signal_connect (G_OBJECT(b1),"clicked",
                    G_CALLBACK(tell_hello) ,NULL);
  gtk_widget_show_all(win);
  gtk_main();
  return 0;
}
```

Premier programme avec GTK (suite)

```
void tell_hello(GtkWidget *gwt, gpointer *data){
    g_warning("hello");
}
```

Variables globales

- Les variables globales sont à utiliser avec précaution, puisqu'elles créent des liens invisibles entre les fonctions. La modularité d'un programme peut en souffrir et le programmeur risque de perdre la vue d'ensemble.

Principe

- Mécanisme pour éviter les variables globales : les **“user data”**.

Principe :

- On crée un struct **“fourre-tout”** qui contiendra toutes nos données
- On associe ce struct aux signaux
- GTK nous le donne en **user_data** dans les **callbacks**

Exemple d'utilisation

```
typedef struct {
    GtkWidget *window;
    char *title;
}
Mydata;

int main (int argc, char *argv[]){
    Mydata my;
    my.title = "Hello World Again!";
    ...
    g_signal_connect (GTK_BOX(my.window), "clicked",
                      G_CALLBACK(my_callback_fct), &my);
    ...
}
```

Exemple d'utilisation (suite)

```
void my_callback_fct (GtkApplication* app, gpointer user_data){
    Mydata *my = user_data;
    ...
}
```

Les évènements

- Hérités : Un widget implémente les évènements des classes mères
- Spécifique : Un widget implémente aussi ses propres événements

Documentation

- La documentation de chaque widge inclue une section **Signals** dans laquelle on trouve l'ensemble des évènements que ce widget peut traiter
- Les évènements des classes mères ne sont pas inclus