Programmazione di GUI con GTK

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

About the author:
Uso Linux dal 1997. Libertà, flessibilità e opensource. Queste sono le caratteristiche che mi piacciono.

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

Abstract:

In questa serie di articoli impareremo come scrivere interfacce grafiche (GUI) con GTK. Non ho idea di quanto durerà. Per capire questi articoli dovreste conoscere le seguenti caratteristiche del linguaggio C:

- Variabili
- Funzioni
- Puntatori

Cos'è GTK?

GTK (GIMP Toolkit) è una libreria per la creazione di Interfacce Utente Grafiche (GUI, Graphical User Interface). È disponibile sotto licenza GPL. Usando questa libreria potete creare programmi open−source, free o commerciali.

La libreria si chiama GIMP Toolkit (GTK) perché originariamente è stata creata per sviluppare GIMP (General Image Manipulation Program). Gli autori di GTK sono:

- Peter Mattis
- Spencer Kimball
- Josh MacDonald

GTK è un'interfaccia utente orientata agli oggetti. Anche se è scritta in C, usa le idee di classe e funzione di callback.

Compilazione

Per compilare programmi GTK, dovete dire a gcc cosa sono e dove si trovano le librerie GTK. Il programma *gtk−config* "conosce" queste cose.

gtk−config −−cflags −−libs

L'output di questo comando è simile al seguente (dipende dal sistema):

−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

I parametri hanno il seguente significato:

−l libreria: Cerca una libreria con nome lib*libreria*.a nei path definiti.
−L path: Aggiunge un percorso dove cercare le librerie.
−I path: Aggiunge un percorso dove cercare i file header.

(NdT: il comando si riferisce alla versione fino alla 1.2 di Gtk+. Dalla versione 2.0 si usa pkg−config con una sintassi diversa)

Per compilare un programma GTK di nome hello.c, si può usare il seguente comando:

gcc −o hello hello.c `gtk−config −−cflags −−libs`

Quello che segue il parametro −o è il nome del programma compilato.

Un primo programma

Assumiamo che GTK sia installato nel vostro sistema. L'ultima versione di GTK può essere trovata a ftp.gtk.org.

Scriviamo il nostro primo programma. Questo programma crea una finestra 200x200, vuota.

```
#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 è un tipo variabile per definire diverse componenti come finestre, bottoni, etichette... In questo esempio viene definita una finestra nel modo seguente:

GtkWidget *window;

void gtk_init(int *argc, char ***argv) inizializza il toolkit, passandogli i parametri scritti nella linea di comando. Questa funzione deve essere usata dopo aver definito i componenti.

GtkWidget *gtk_window_new(GtkWindowType windowtype) crea una nuova finestra. Il tipo di finestra puó essere:

- GTK_WINDOW_TOPLEVEL
- GTK_WINDOW_DIALOG
- GTK_WINDOW_POPUP

void gtk_widget_show(GtkWidget *widget) viene usato per far apparire il componente nella finestra. Dopo aver definito un componente e averne cambiato gli attributi bisogna usare questa funzione.

void gtk_main(void) prepara le finestre e tutti i componenti per farli apparire sullo schermo. Questa funzione deve essere usata alla fine dei programmi GTK.

Usiamo un po' di proprietà della finestra, come titolo, dimensioni, posizione, ...

void gtk_window_set_title(GtkWindow *window, const gchar *title) viene usata per impostare o cambiare il titolo della *window*. Il primo parametro della funzione è di tipo GtkWindow. Ma *window* é di tipo GtkWidget. Durante la compilazione verremo avvisati di questo. Anche se il programma compilato funziona, è meglio correggerlo. GTK_WINDOW(GtkWidget *widget) viene usato per questo motivo. Il secondo parametro *title* é di tipo gchar. gchar viene definito nella libreria glib ed é la stessa cosa del tipo char.

void gtk_window_set_default_size(GtkWindow *window, gint width, gint height) imposta la dimensione di *window*. Come gchar, gint viene definito in glib ed é la stessa cosa di int.

La funzione

void gtk_window_set_position(GtkWindow *window, GtkWindowPosition position)

imposta la posizione di *window*. *position* può essere:

- GTK_WIN_POS_NONE
- GTK_WIN_POS_CENTER
- GTK_WIN_POS_MOUSE
- GTK_WIN_POS_CENTER_ALWAYS

Ecco un esempio:

```
#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);
}
```

Segnali ed Eventi

Nelle GUI si possono usare mouse e tastiera, per esempio si può clickare su un pulsante. Si deve quindi usare la seguente funzione GTK:

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

object é il componente che emette il segnale. Per esempio, se volete sapere quando un pulsante viene clickato, *object* sarà il pulsante. *name* è il nome dell'evento e può essere:

- 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 è il nome della funzione che verrà richiamata quando accadrà l'evento. Ecco un esempio:

```
#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 funzione

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

rimane in ascolto di ricevere l'evento di distruzione della finestra. Quando si tenta di chiudere la finestra viene chiamata la funzione *close*. La funzione *close* chiama gtk_main_quit() e il programma finisce.

Piú avanti verranno spiegati i dettagli dei segnali e degli eventi.

Un normale bottone

I bottoni vengono generalmente usati per fare determinate azioni quando vengono clickati. In GTK ci sono due modi per creare i bottoni:

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

La prima funzione crea un bottone senza etichetta (non c'è scritto niente dentro il bottone). La seconda crea un bottone con un'etichetta (*label* viene scritto nel bottone).

Qui useremo una nuova funzione:

void gtk_container_add(GtkContainer *container, GtkWidget *widget)

Usando questa funzione é possibile fare in modo che un bottone (o genericamente un qualsiasi componente) appaia in una finestra (o genericamente un qualsiasi contenitore). Nel prossimo esempio il contenitore é una finestra e il componente é un bottone. Piú avanti impareremo di piú sui contenitori.

La cosa piú importante riguardo i bottoni è sapere se sono stati clickati o meno. Anche qui si usa la funzione gtk_signal_connect. Questa volta chiameremo un'altra funzione che verrà eseguita "dietro" il bottone. Ecco un esempio:

```
#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);
}
```