LinuxFocus article number 295
http://linuxfocus.org

# Grafisch programmeren met GTK

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

*Over de auteur:*
Sinds 1997 gebruik ik Linux. Vrijheid, flexibiliteit en open broncode. Dat zijn de eigenschappen die mijn hart hebben veroverd.

*Vertaald naar het Nederlands door:*
Samuel Derous
<cyberprophet(at)linux.be>

*Kort*:

In deze serie artikelen, zullen we leren hoe we grafische gebruikersinterfaces (GUI) moeten programmeren door gebruik te maken van GTK. Ik heb er geen idee van hoe lang deze serie zal zijn. Om deze artikelen te begrijpen, zou je volgende dingen over de C programmeertaal moeten kennen:

- Variabelen
- Functies
- Pointers

## Wat is GTK?

GTK (GIMP Toolkit) is een bibliotheek om Grafische gebruikersinterfaces aan te maken. De bibliotheek is onder de GPL licentie beschikbaar. Door gebruik te maken van deze bibliotheek kan je open broncode, gratis of commerciële programma's ontwerpen.

De bibliotheek heeft de naam GIMP tookit (GTK) gekregen omdat het in eerste instantie ontworpen was voor het ontwikkelen van de GIMP (General Image Manipulation Program). De auteurs van GTK zijn:

- Peter Mattis
- Spencer Kimball
- Josh MacDonald

GTK is een object-georiënteerd toepassingsgebruikersinterface. Daar het toch in C geschreven is, gebruikt het niettemin het idee van klasses en terugroep functies.

# Compileren

Om GTK programma's te compileren dien je gcc te vertellen wat de GTK bibliotheken zijn, en waar hij ze kan vinden. De gtk-config commando "weet" dit.

# gtk-config --cflags --libs

De output van dit commando is iets als het volgende (hangt af van het systeem):

-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

De verklaring voor deze parameters zijn:

-l bibliotheek: zoekt naar bibliotheken in de vorm van library.a in de gedefinieerde paden
-L pad: voegt een pad toe om de bibliotheken te zoeken.
-I pad: voegt een pad toe om te zoeken naar header-bestanden die in het programma gebruikt worden.

Het volgende commando kan gebruikt worden om een GTK programma, hello.c genaamd, te compileren:

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

De input die na de -o parameter gebruikt wordt, is de naam van het gecompileerde programma.

# Een eerste programma

Er wordt aangenomen dat GTK op je systeem is geïnstalleerd. De laatste versie van GTK kan gevonden worden op ftp.gtk.org.

Laten we ons eerste programma schrijven. Dit programma maakt een leeg venster van 200x200 pixels breed.

```
#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 een variabele type om verschillende componenten zoals window, button, label te definiëren... In dit voorbeeld wordt een venster op de volgende manier gedefinieerd:

GtkWidget *window;

void gtk_init(int *argc, char ***argv) initialiseert de toolkit en verkrijgt de parameters die in de shell worden ingegeven. Deze functie moet gebruikt worden nadat de componenten gedefinieerd zijn.

GtkWidget *gtk_window_new(GtkWindowType windowtype) ontwerpt een nieuw venster. De volgende venster types zijn mogelijk:

- GTK_WINDOW_TOPLEVEL
- GTK_WINDOW_DIALOG
- GTK_WINDOW_POPUP

void gtk_widget_show(GtkWidget *widget) wordt gebruikt om het component in een venster weer te geven. Nadat een component gedefinieerd is en de attributen veranderd, dient deze functie gebruikt te worden.

void gtk_main(void) bereidt vensters en alle componenten die op het scherm moeten verschijnen voor. Deze functie dient op het einde van GTK programma’s gebruikt te worden.

Laten we enkele venstereigenschappen zoals titel, grootte, positie... gebruiken

void gtk-window_set_title(GtkWindow *window,const gchar *title) wordt gebruikt om een titel aan window te geven of een titel te veranderen. Eerste parameter van deze functie is GtkWindow type. Maar de variabele van window bevindt zich in GtkWidget type. Wanneer we dit zullen compileren, zullen we hierover gewaarschuwd worden. Terwijl het gecompileerde programma zal werken, is het beter om het te verbeteren. Daarvoor wordt GTK_WINDOW(GtkWidget *widget) gebruikt. De tweede parameter title is in gchar type. gchar wordt in de glib bibliotheek gedefinieerd en is hetzelfde als het char type.

void gtk_window_set_default_size(GtkWindow *window, gint width, gint height) stelt de grootte van window in. Net zoals gchar, wordt gint gedefinieerd in glib en is hetzelfde als int.

de functie

void gtk_window_set_position(GtkWindow *window, GtkWindowPosition position)

stelt de grootte in van window.position en kan zijn:

- GTK_WIN_POS_NONE
- GTK_WIN_POS_CENTER
- GTK_WIN_POS_MOUSE
- GTK_WIN_POS_CENTER_ALWAYS

Hier is een voorbeeld:

```
#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);
}
```

# Signalen en events

In GUIs, dient je gebruik te kunnen maken van het toetsenbord en de muis, zodat je bijvoorbeeld op een knop kunt klikken. Daarvoor wordt de volgende GTK functie gebruikt:

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

object is het component dat signalen uitzendt. b.v. als je wilt weten of er op een knop geklikt is, zal het object button.name zijn, name zal de naam van het event bevatten, die kan bestaan uit:

- 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

*funct* is de naam van de functie die aangeroepen zal worden wanneer de gebeurtenis plaats vindt. Hier volgt een voorbeeld:

```
#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);
}
```

De functie

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

Luistert naar de vernietigingsevent van het venster. Wanneer men probeert het venster te sluiten, wordt de close functie aangeroepen. De close functie roept gtk_main_quit() aan en het programma wordt beëindigd.

Details over signalen en gebeurtenissen worden later uitgelegd...

# Een doodgewone knop

Gewone knoppen worden over het algemeen gebruikt om bepaalde dingen te doen wanneer er op de knop wordt gedrukt. In de GTK bibliotheek zijn er twee manieren om een knop te maken:

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

De eerste functie maakt een knop zonder label aan (er wordt niets op de knop geschreven). De tweede maakt een knop met een label aan (het label wordt op de knop geschreven).

We zullen hier een nieuwe functie gebruiken:

void gtk_container_add(GtkContainer *container,GtkWidget *widget)

Met het gebruik van deze functie is het mogelijk een knop aan te maken (in het algemeen met alle componenten) die op het venster zal verschijnen (in het algemeen in een vierkant venster). In het volgende voorbeeld is de container een venster en het component die moet worden toegevoegd een knop. Later zullen we over enkele andere containers leren.

Het belangrijkste van een knop is om te weten of erop geklikt is of niet. Voor dit doel wordt weer de gtk_signal_connect functie gebruikt. Met deze functie zal een andere functie aangeroepen worden en zal de functionaliteit "achter" de knop worden uitgevoerd. Hier volgt een voorbeeld:

```
#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