Creating GUIs using GTK+ on Linux Operating System

By: Terrell Black
A30230307

ECE 480
Date: Friday, November 9, 2007

CONTENTS

Brief Background of GTK+

The GIMP Toolkit (GTK+) is a cross platform widget toolkit for developing a graphical user interface (GUI) for a computing system. Its original purpose was originally designed for a raster graphics editor called the GNU Image Manipulation Program (GIMP). It was created in 1997 by three individuals, Peter Mattis, Spencer Kimball, and Josh MacDonald, while working in the eXperimental Computing Facility at the University of California, Berkeley. GTK+ was primarily developed to be used in Linux desktop environments, which still remains its primary usage. It is also worth mentioning that platform binding have been created to work with other major operating systems, such as Microsoft Windows, BeOS, Solaris, Mac OS X, and others.

GTK+ uses the C programming language, although its design uses the GObject[1] object system. Other programming language binding have been created for GTK+ for C++, Perl, Ruby, Python, Java, C#, plus several others. In the context of this application note we will be focusing on developing a GTK+ application using the standard C programming language on a Linux operating system.

Linux X-Window System

The X-Window system, or commonly X11 or X, is is a display protocol which provides windowing on bitmap displays. Developed in 1984 by Jim Gettys and Bob Scheifler at Massachusetts Institute of Technology, its purpose was to be a platform-independent display environment for debugging the Argus system. In a more basic definition X-Window provides functionality for Linux to display graphical windows to the user. It draws windows on the screen and handles their movements. X11 also controls input devices, such as mice and keyboards. X11 is a essential component to any Linux operating system.

One of the main reasons for creation and features of X11 is its separation of the operating system and the graphical user interface. This is a unique feature among other display managers, in which X11 assumes the client and server are treated independently of each other. In this definition the client is the computer that the application is being executed on, and the server is where the GUI is displayed. This allows the client to exist at a remote location independent of the server. It is possible and also most common that both the server and client is the same physical device.

X11 provides by default all the functionality that a programmer would need to create a GUI. So one might ask why you need GUI toolkits such as GTK+. X11 has a basic programming interface, named Xlib, but creating a GUI using it would be very tedious. Using Xlib the programmer would have to manage all of the low level system calls, making life a lot more difficult. Thus enters GUI toolkits, such as GTK+, which hides all low level system calls and dynamically manages them through library methods

GTK+ and Libraries Needed

GTK+ relies on multiple supporting libraries, each providing a different level of functionality to the programmer. GTK+ is written in the C programming language in an object oriented format. Its implementation is done with classes which create an extensible system that can be built upon. GTK+ initially contained the object oriented framework as part of its library. In the current lease this has been taken out and added to the external Glib library. The GTK+ library provides all the functionality for creating GUI elements, such as buttons, radio button, windows, ect. GUI elements created with GTK+ will be further referred to as widgets. Other basics necessary for creating a GUI, such as even processing, are handles outside of the GTK+ library. However GTK+ does provide access to other libraries through its own API (application programming interface).

I. Glib

Glib is a cross platform library used by GTK+ to provide many useful none graphical functionalities. It is required for GTK+ to operate, but it can also be used independently for other applications. Some of its library features are: basic types and their limits, type conversions, memory allocation, timers, string utilities and threads. Glib also provides important data structures such as: memory chunks, doubly- and singly-linked lists, hash tables, dynamic strings, vector style arrays, and balanced binary trees.

II. GObject

As mentioned before the GLib Object System (GObject) was originally part of the GTK+ library, but has been separated in the newest release of GTK+. It was originally created to allow easy access to C objects from other programming languages. This is particularly important because different programming languages gave different implementations of data types. So it is very useful for a cross platform toolkit to have this level of modularity. GObject also provides the object oriented interface in C. This system is used by all GTK+ widgets plus various other data types.

III. GDK

GDK (GIMP Drawing Kit) is the computer graphics library that acts as a wrapper around the low-level X11 display manager of the Linux operating system. GDK lies between the X server (X11) and the GTK+ library. It handles all rendering of drawings, raster graphics, cursors, and fonts for GTK+ applications.

IV. GdkPixbuf

GdkPixbuf is the library that provides all of the 2D graphic manipulation for GTK+ applications. Using this library we can shear, rotate and scale images to our own satisfaction.

V. Pango

Pango controls text and font output in conjunction with Cairo or Xft, depending on your GTK+ version. When adding text to your GUI, the Pango library will handle all properties such as size, color, style and fonts. Many of the markup features provided by Pango can be implemented in a HTML format, which makes its usage very easy.

VI. ATK

The Accessibility Toolkit (ATK) provides all GTK+ widgets with a built-in method of handling accessibility issues. ATK adds support for are screen readers, high-contrast visual themes, and keyboard behavior modifiers

.

Installing GTK+

Before we can begin installing GTK+ we must first have all of its supporting libraries installed. The method for installation is somewhat independent to your specific distribution of Linux. For most common distributions such as Ubuntu, Debian, Fedora Core, or one of many others the package manager is provided by default. In this instance you should install the precompiled binaries for GTK+. If you are installing the development package of GTK+, most modern package managers will take care of all of the necessary dependencies automatically. You should reference your Linux distribution's documentation for more information on installing distributed packages.

If you are going to install GTK+ and its dependencies from the source archives, the most recent GTK+ sources can be found at www.gtk.org/download. After installing all of the packages from the dependencies directory on the GTK+ FTP site, you will need to install GLib, Pango, ATK, and then finally GTK+ in that specific order.

In the directory of each package the following command prompt commands should be entered one at a time:

```
./configure –prefix=/usr
make

make install

ldconfig
```

Further in depth explanation of these commands can be found online and in the references at the end of this document. To test your system for a successful installation the following command can be executed:

```
/usr/bin/gtk-demo
```

If your install was successful, you will be presented with a window with the title "GTK+ Code Demos".

Creating a simple GUI

Below is a very simple example of a GTK+ GUI application. It shows the bare necessities for any GTK+ application. All the following explanations in this section will refer to the following source code:

Listing 1. (helloWorld.c)

```
1.  #include <gtk/gtk.h>
2.
3.  Int main (int argc, char *argv[])
4.  {
5.      GtkWidget *window;
6.
7.      /* Initialize GTK+ and all of its supporting libraries. */
8.      gtk_init (&argc, &argv);
9.
10.     /* Create a new window, give it a title and display it to the user. */
11.     window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
12.     gtk_window_set_title (GTK_WINDOW (window), "Hello World");
13.     gtk_widget_show (window);
14.
15.     /* Hand control over to the main loop. */
16.     gtk_main ();
17.     return 0;
18. }
```

I. Adding Libraries

First of all at the beginning of the GTK+ application you must include all the necessary functionality. This information is in the <gtk/gtk.h>

file listed on line 1, which includes all of the widgets, variables, functions, and structures available in GTK+ as well as header files from other libraries that GTK+ depends on, such as <glib/glib.h> and <gdk/gdk.h>. Initializing GTK+ and all of its libraries is a very simple process and is done in most cases by one command. This command found on line 8, also takes any arguments provided to the application and makes appropriate use of them.

II. Setting up our window widget

Next we want to create a widget for the GUI window which is done by created a pointer variable to the base widget class structure (GtkWidget). After creating the appropriate pointer we must use GTK+ built in interface to actually create the GUI element and also set its properties. First on line 11 we use the gtk_window_new (GTK_WINDOW_TOPLEVEL) function to create us a new window to be drawn to the screen. The argument GTK_WINDOW_TOPLEVEL is a GTK+ constant telling the application that this window is main parent widget for the application. The next two lines on 12 and 13 set the title property of the window and tell the OS to draw the window on your screen.

III. Handing over control

Any type of GUI application runs on events. Events can be multiple types of actions, mostly created by user interaction like button clicks, typing, ect. Independent of the type of event we need some way of monitoring for these actions and then making the appropriate decision. To implement this we need to hand over processing control of our GUI to GTK+ main loop. This is done on line 16 of the source code. This line should always be the last line in any GTK+ main function.

IV. Compiling

To execute this code into an executable the foll wing code can be used:

```
gcc -Wall -g helloworld.c -o helloworld `pkg-config --cflags gtk+-2.0` \
`pkg-config --libs gtk+-2.0`
```

Here I am using the GCC compiler, because it is the standard C compiler on Linux. Most C and C++ compilers will work also.

Reference Sources

1. Krause, Andrew. Foundations of GTK+ Foundations in GTK+ Development. New York City: Apress, 2007. 1-630.
2. "GObject - Wikipedia." Wikipedia. 21 Nov. 2007 <1.http://en.wikipedia.org/wiki/GObject>.
3. "Linux - Wikipedia." Wikipedia. 21 Nov. 2007 <http://en.wikipedia.org/wiki/Linux>.
4. "X Window System - Wikipedia." Wikipedia. 21 Nov. 2007 <http://en.wikipedia.org/wiki/X_Window_System>.
5. "Object Oriented Programming." Object Oriented Programming - Wikipedia. 21 Nov. 2007 <http://en.wikipedia.org/wiki/Object-oriented_programming>.