Tutorial for GTK+

CS23

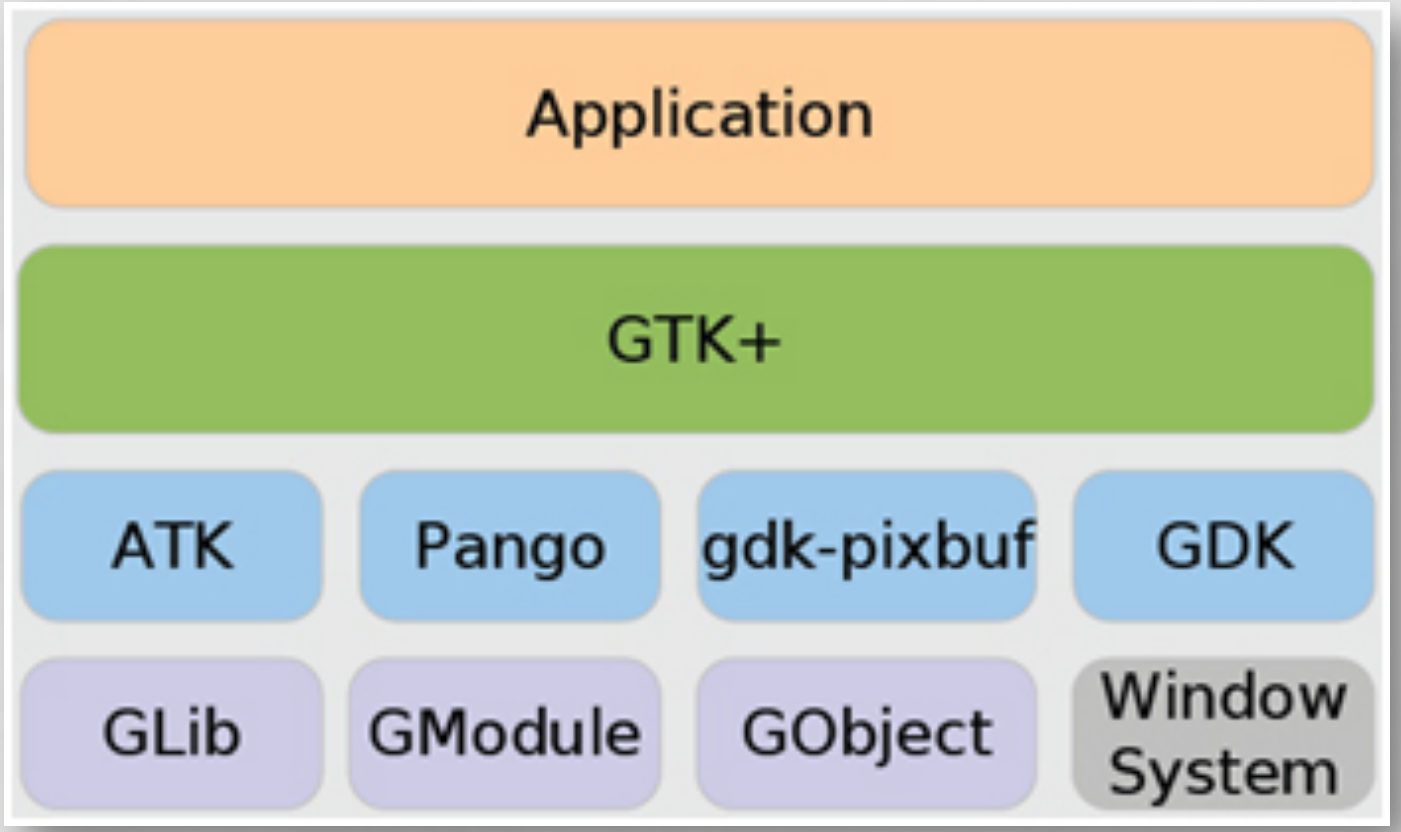
2010.2.20

What is GTK+?

Introduction

- GTK+ (GIMP Toolkit)
  - GIMP (GNU Image Manipulation)
- Default graphical toolkit of GNOME and XFCE
- Written entirely in C
- Majority of GTK+ software is also written in C
- Language binding
  - C++, Python, PHP, Ruby, Perl, C#, or Java

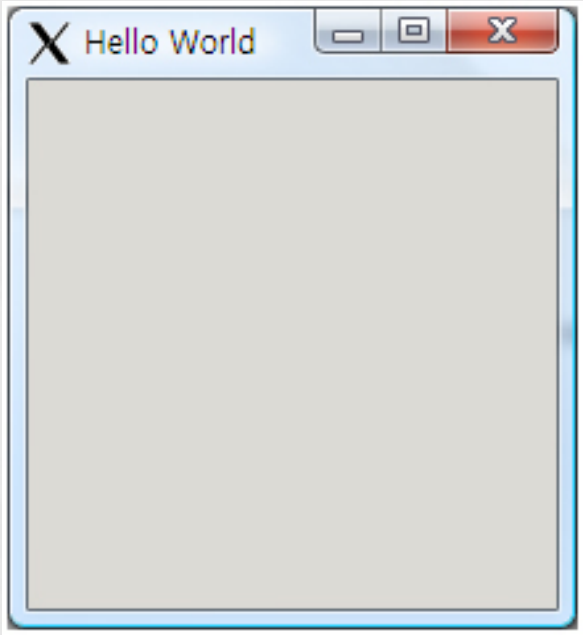
Architecture



How to make a GTK+ application

Simple example

- Basic structure of a GTK+ application
- Even if clicking on ‘X’ button, this application will not be killed
  - should use *ctrl*+*c*



Simple example

```
#include <gtk/gtk.h>

int main (int argc, char *argv[])
{
    GtkWidget *window;

    /* Initialize the GTK+ and all of its supporting libraries. */
    gtk_init (&argc, &argv);

    /* Create a new window, give it a title and display it to the user. */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Hello World");
    gtk_widget_show (window);

    /* Hand control over to the main loop. */
    gtk_main ();
    return 0;
}
```

How to compile

gcc hello.c -o hello `pkg-config --cflags --libs gtk+-2.0`

- ***Take care to type backticks, not apostrophes***
  - ***backticks are instructions to the shell to execute and append the output of the enclosed command.***
- ***` (backticks) is located on the upper side of 'tab' key***

Events, Signal, and Callback

- Event
  - A message emitted by the X Window system
  - When a user performs some action, it's sent to your application
- Signal
  - Reaction to an event
  - Emitted by a GtkObject
- Callback function
  - Run a function by a signal

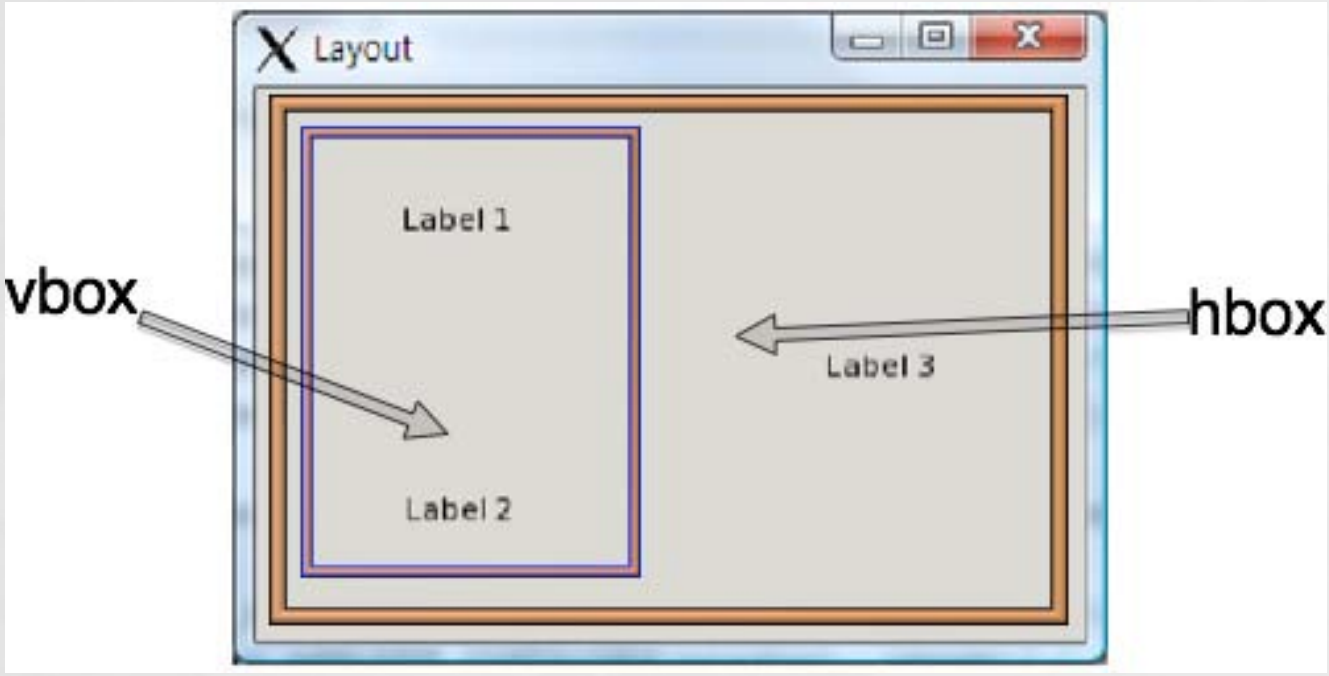
```
gulong g_signal_connect ( gpointer object,
                          const gchar *signal_name,
                          GCallback handler,
                          gpointer data);

static void callback_function ( GtkWidget *widget,
                                gpointer data);
```

```
#include <gtk/gtk.h>

void destroy(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_window_set_title(GTK_WINDOW(window), "Hello World!");
    gtk_widget_show (window);
    /* Connect the main window to the destroy */
    g_signal_connect(G_OBJECT(window), "destroy",
                                    G_CALLBACK(destroy), NULL);
    gtk_widget_show(window);
    gtk_main();
    return 0;
}
```

Widgets

- Basic building blocks of a GUI application
- Window widget
  - Basic element of all GTK+ application.
- Container widgets for layout
  - To contain one or more children
  - Non-visible
  - GtkBox, GtkHBox, GtkVbox, etc
- Basic widgets
  - GtkLabel, GtkButton, GtkEntry, GtkImage, etc

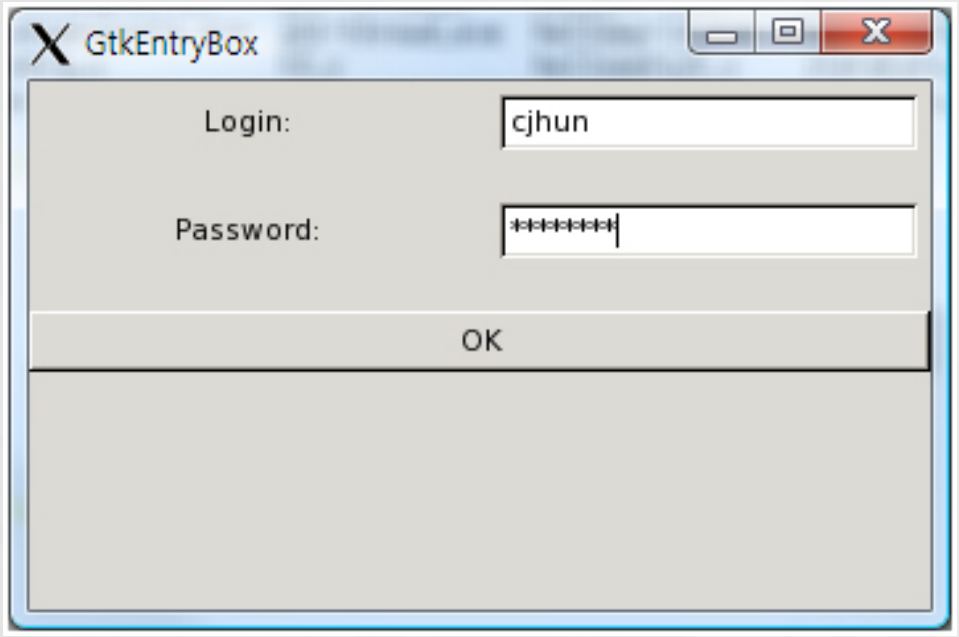
Container example




```
#include <gtk/gtk.h>

void closeApp(GtkWidget *widget, gpointer data)
{
    gtk_main_quit();
}
```

```
int main( int argc, char *argv[])
{
    ……...

    label1 = gtk_label_new("Label 1");
    label2 = gtk_label_new("Label 2");
    label3 = gtk_label_new("Label 3");

    hbox = gtk_hbox_new(TRUE, 5);
    vbox = gtk_vbox_new(FALSE, 10);

    gtk_box_pack_start(GTK_BOX(vbox), label1, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(vbox), label2, TRUE, FALSE, 5);

    gtk_box_pack_start(GTK_BOX(hbox), vbox, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox), label3, FALSE, FALSE, 5);

    gtk_container_add(GTK_CONTAINER(window), hbox);

    ……...
}
```

Basic widgets example



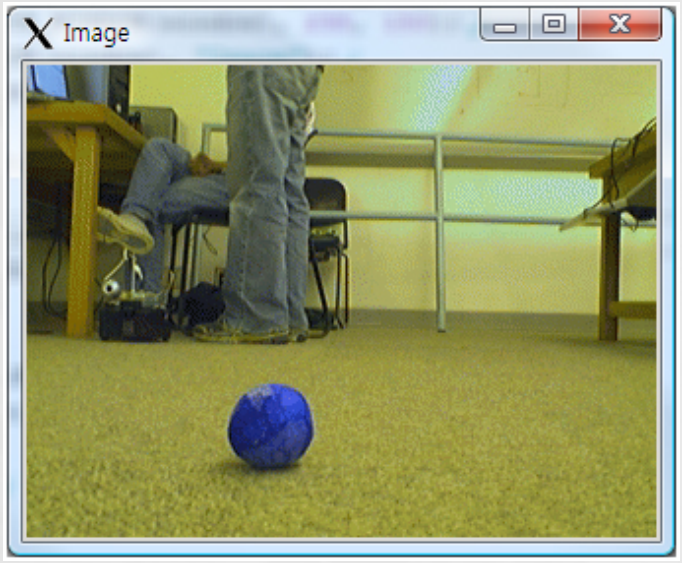

```
#include <gtk/gtk.h>

void closeApp(GtkWidget *widget, gpointer data)
{
    gtk_main_quit();
}
```

```
void button_clicked(GtkWidget *button, gpointer data)
{
    const char *password_text;
     password_text = gtk_entry_get_text(GTK_ENTRY((GtkWidget *)data));

    if(strcmp(password_text, password) == 0)
        printf("Access granted!\n");
    else
        printf("Access denied!\n");
}

int main(int argc, char *argv[])
{
    GtkWidget *window;
    GtkWidget *username_label, *password_label;
    GtkWidget *username_entry, *password_entry;
    GtkWidget *ok_button;
    GtkWidget *hbox1, *hbox2, *vbox;
```

```
gtk_init(&argc, &argv);

window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
gtk_window_set_title(GTK_WINDOW(window), "Basic Widgets");
gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
gtk_window_set_default_size(GTK_WINDOW(window), 200, 200);

g_signal_connect(G_OBJECT(window), "destroy", G_CALLBACK(closeApp),
                                                              NULL);


username_label = gtk_label_new("Login: ");
password_label = gtk_label_new("Password: ");

username_entry = gtk_entry_new();
password_entry = gtk_entry_new();
gtk_entry_set_visibility(GTK_ENTRY(password_entry), FALSE);

ok_button = gtk_button_new_with_label("OK");
g_signal_connect(G_OBJECT(ok_button), "clicked",
                              G_CALLBACK(button_clicked), password_entry);
```

```
    hbox1 = gtk_hbox_new(TRUE, 5);
    hbox2 = gtk_hbox_new(TRUE, 5);
    vbox = gtk_vbox_new(FALSE, 10);

    gtk_box_pack_start(GTK_BOX(hbox1), username_label, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox1), username_entry, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox2), password_label, TRUE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(hbox2), password_entry, TRUE, FALSE, 5);

    gtk_box_pack_start(GTK_BOX(vbox), hbox1, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(vbox), hbox2, FALSE, FALSE, 5);
    gtk_box_pack_start(GTK_BOX(vbox), ok_button, FALSE, FALSE, 5);

    gtk_container_add(GTK_CONTAINER(window), vbox);

    gtk_widget_show_all(window);
    gtk_main();

    return 0;
}
```

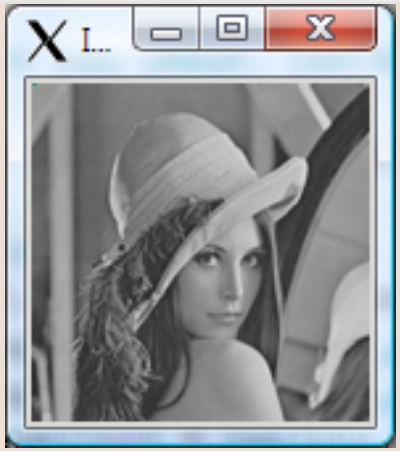
Image from file




```
GtkWidget *window, *image;

image = gtk_image_new_from_file("pic/60cm.gif");

gtk_container_add(GTK_CONTAINER(window), image);
```

Image from data

```
char *rgbImage;
GtkWidget *image;

int main( int argc, char *argv[])
{
    ……..

    image = gtk_image_new();
    gtk_container_add(GTK_CONTAINER(window), image);

    ……..

    loadImage(char *rgbImage);

    gtk_main();
    return 0;
}
```



```
int loadImage(char *data)
{
    printf("Got image!\n");
    GdkPixbuf *pixbuf;

    pixbuf = gdk_pixbuf_new_from_data(data, GDK_COLORSPACE_RGB, FALSE, 8,
                                      CAMERA_WIDTH, CAMERA_HEIGHT,
                                      CAMERA_WIDTH  * 3, NULL, NULL);

    gtk_image_set_from_pixbuf((GtkImage*) image, pixbuf);
    gtk_widget_queue_draw(image);
    printf("Loaded\n");

    return 0;
}
```

Threads in Gtk

- **Should use the following when compiling:**
  `pkg-config --cflags --libs gtk+-2.0 gthread-2.0`

- **First few lines of your gui main should look like:**

```
g_thread_init(NULL);
gdk_threads_init();
gtk_init(&argc, &argv);
```

- **All gtk functions calls should be protected by gdk_threads_enter(); and gdk_threads_leave();**

Threads in Gtk - Example

```
#include <gtk/gtk.h>
#include <unistd.h>
#include <pthread.h>

GtkWidget* window, label, container;

void* changelabeltext(void* ptr) {        // changes the label text to “B” after 3 seconds
  sleep(3);
  gdk_threads_enter();
  gtk_label_set_text((GtkLabel*)label, "B");
  gdk_threads_leave();
}

int main (int argc, char *argv[]) {        // opens a window, displays a label that says “A”
  g_thread_init(NULL);
  gdk_threads_init();
  gtk_init(&argc, &argv);
  gdk_threads_enter();
  label = gtk_label_new("A");
  window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title (GTK_WINDOW (window), "Hello World");
  gtk_container_add(GTK_CONTAINER(window), label);
  gtk_widget_show_all(window);
  pthread_t p1;
  pthread_create(&p1, NULL, changelabeltext, NULL);
  gtk_main ();
  gdk_threads_leave();
  return 0;
}
```

Useful links

- The GTK+ Project
  - http://www.gtk.org/index.html
- GTK+ Reference Manual
  - http://library.gnome.org/devel/gtk/stable/index.html
- GDK Reference Manual
  - http://library.gnome.org/devel/gdk/unstable/gdk-Threads.html
- GObject Reference Manual
  - http://library.gnome.org/devel/gobject/stable/index.html
- GTK+ tutorial
  - http://zetcode.com/tutorials/gtktutorial/

Reference

- Useful links
- Neil Matthew, Richard Stones, “Beginning Linux Programming 4th Edition”, WROX
- Andrew Krause, Foundation of GTK+ Development, APR ESS