Outline

- GTK packages
- Introduction to GTK toolkit
- GTK "Hello World" example
- Layout containers
GTK Packages

● The currently installed GTK development toolkit package is **libgtk-3-dev**. For the full GNOME development package that includes the GTK packages, you can install the **gnome-devel** package.

● The supplemental reference page has links to the GTK Home Page (GTKH), Reference Manual (GTKR), FAQ (GTKF) and Tutorial (GTKT) as well as a Beginner's Tutorial (GTKB)

● You can find complete documentation at http://developer.gnome.org/
Introduction to GTK

• GTK+ is a multi-platform toolkit for creating graphical user interfaces. It was initially developed for and used by the GIMP (the GNU Image Manipulation Program), but is now used by several applications. It is the toolkit used by the GNOME desktop.

• GTK+ applications can be compiled to run under X windows, MS Windows and the Mac OS.
Introduction to GTK

- GTK+ consists (primarily) of 3 libraries:
  - GLib is the low-level core library that forms the basis of GTK+ and GNOME.
  - Pango is a library for layout and rendering of text, with an emphasis on internationalization.
  - The ATK library provides a set of interfaces for accessibility. By supporting the ATK interfaces, an application or toolkit can be used with such tools as screen readers, magnifiers, and alternative input devices.
Introduction to GTK

- Under X Windows GLib depends upon the GDK (GTK+ Drawing Kit) library which is a wrapper around corresponding Xlib routines.
- GTK+ is an object-oriented library written in C. There are bindings for many other languages including: C++, Perl, Python, Guile, Java, and C#. Only the C interface will be discussed.
- We will start with an example from the GTK Tutorial.
Windows and Widgets

- Every GUI consists of one or more widgets (i.e., GUI elements). There is at least one main widget that is a window. Other widgets are contained in or controlled by the main window widget.

- The simplest widget is a button. Today all the examples will use buttons. Next class we will look at other common input widgets.
GTK – A First Example

• A first GTK program in example-0.c (cont on next slide):

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

static void activate (GtkApplication* app, gpointer user_data);
Int main (int argc, char **argv)
{
    GtkApplication *app;
    int status;
    app = gtk_application_new ("cs375.xmpl",
                   G_APPLICATION_FLAGS_NONE);
    g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
    status = g_application_run (G_APPLICATION (app), argc, argv);
    g_object_unref (app);
    return status;
}
```
static void activate (GtkApplication* app, gpointer user_data)
{
    GtkWidget *window;

    window = gtk_application_window_new (app);
    gtk_window_set_title (GTK_WINDOW (window), "Window");
    gtk_window_set_default_size (GTK_WINDOW (window), 200, 200);
    gtk_widget_show_all (window);
}
To compile the program use:

\$ \texttt{gtkflags=\}\n\n\texttt{`pkg-config --cflags --libs gtk+-3.0`}\n\n\$ \texttt{gcc -o example-0 example-0.c $gtkflags}\n
\section*{pkg-config}

\texttt{pkg-config} returns information about libraries. It is used to get compiler and linker arguments:

\$ \texttt{pkg-config --cflags --libs gtk+-3.0}\n-\texttt{l/usr/include/gtk-3.0}\n-\texttt{l/usr/include/gtk-3.0}\n-\texttt{l/usr/include/cairo}\n-\texttt{l/usr/include/pango-1.0}\n-\texttt{lgtk-3}\n-\texttt{lgdk-3}\n-\texttt{lcairo}\n-\texttt{lgobject-2.0}\n-\texttt{lglib-2.0}\n...
GTK – A First Example

- See the Makefile for use of a macro for the compilation command using pkg-config.
- When we run this program, we see:
GTK – A First Example

• All programs must include `<gtk/gtk.h>`.

• The purpose of `main()` is to create a `GtkApplication` object and run it. In this example an application pointer named app is initialized using `gtk_application_new()`.

• A GTK application should have a unique name (identifier). Here `cs375.xmpl` is used. (Domain names or email prefixes are recommended – edu.evansville.ar63.example-0)
Next the activate signal is connected to the `activate()` function (second slide). The activate signal is sent when the app is launched with `g_application_run()`. `gtk_application_run()` also takes as arguments the pointers to the command line arguments counter and string array. This allows GTK+ to parse specific command line arguments that control the behavior of GTK+ itself.
GTK – A First Example

• Within `g_application_run` the `activate()` signal is sent and triggers our `activate()` function.

• Inside `activate()` we construct our GTK window. The call to `gtk_application_window_new()` returns a pointer to the new window. The window will have a frame, a title bar, and window controls depending on the platform.
GTK – A First Example

• The title is set using `gtk_window_set_title()`. This function takes a `GtkWidget*` pointer and a string. As our window pointer is a `GtkWidget` pointer, we cast it to `GtkWidget*`. Instead of casting via `(GtkWidget*)`, we use the macro `GTK_WINDOW()`. `GTK_WINDOW()` will check if the pointer is an instance of the `GtkWidget` class, before casting, and emit a warning if the check fails.
GTK – A First Example

• The window size is set using `gtk_window_set_default_size()` and the window is shown by GTK via `gtk_widget_show_all()`.

• When you exit the window (by pressing the X), the `g_application_run()` in the main loop returns with a number which is saved inside an integer named `status`. Afterwards, the `GtkApplication` object is freed from memory with `g_object_unref()`. Finally the `status` is returned and the GTK application exits.
Before looking at the next example, we need to discuss signals and callbacks. GTK is an event-driven toolkit. It sleeps in `g_application_run()` until an event occurs. Control is then passed to the appropriate function.

When an event occurs, e.g., a mouse button press, the widget that was pressed emits a signal (not a UNIX process signal, but the same terminology is used.)
**GTK – Signals and Callbacks**

- **`g_signal_connect()`** is used to connect an event (signal) to a function (signal handler):
  
  ```
  gulong g_signal_connect(
      gpointer *object, const gchar *name, 
      GCallback func, gpointer func_dat
  );
  ```

- The first argument is a pointer to the widget that will emit the signal. The second is the signal name. The third is the handler (a **callback**), the fourth is used to pass data to the handler.
GTK – Signals and Callbacks

• `g_signal_connect_swAPPED()` is similar to `g_signal_connect()` but is used to set up a callback to a standard GTK function that takes a single argument (the data argument).

• The signals that each widget can emit are described in the GTK Reference Manual (GTKR). (Don't program in GTK without it!)

• Examine the `example-1.c` program. Note the macros used to cast objects to the correct type.
In-class Exercise

• Modify the code in example-1.c so that the application does not exit when the button is pressed, but does exit upon getting a delete_event signal. (Connect the signal handler to the window object instead of the button. Check the documentation for the window object hierarchy and the signals received by the GtkWidget).
In-class Exercise

• Modify the code in example-1.c so that the button label changes to “Ouch!” when the button is pressed and changes back to “Hello World!” when the button is released. (See the description of the GtkWidget widget in the GTK reference manual for a list of button related signals and functions.)