CSCE 155 - C

Lab 14.0 - Graphical User Interface Programming

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.

Some additional resources that may help with this lab:

- GTK tutorial: http://developer.gnome.org/gtk-tutorial/stable/
- Another GTK Tutorial: http://www.yolinux.com/TUTORIALS/GTK+ProgrammingTips.html#BASICCONFIGURATION
- GTK API: https://developer.gnome.org/gtk3/
- Possible solution for compiling, running on Windows: http://stackoverflow.com/questions/1450445/

Peer Programming Pair-Up

For students in the online section: you may complete the lab on your own if you wish or you may team up with a partner of your choosing, or, you may consult with a lab instructor to get teamed up online (via Zoom).

For students in the face-to-face section: your lab instructor will team you up with a partner.

To encourage collaboration and a team environment, labs are be structured in a *peer programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Graphical User Interface and Event-based Programming in C using GTK
- Compiling and running a GUI program

2 Background

Many programs interact with users using a Graphical User Interface (GUI). Traditional GUI design involves the creation and interaction of widgets: general graphical elements that include labels, text boxes, buttons, etc. Most GUI programming is done using an Application Programmer Interface (API) which is a library or framework that provides a lot of the core functionality including:

- A window manager that handles the interaction of widgets
- A windowing system that works with the underlying operating system and hardware to render the graphics
- Factory functions that can be used to construct and configure widgets

The particular API that we will work with is GTK (GNU Tool Kit) which is written in C and has support for linux and windows systems. However, given the configuration issues we will focus on linux.

For this lab, be sure to login to your CSE account through the linux system in the labs.

A simple GUI program

We have provided a simple calculator program, (`gtk_calculator.c`) that simulates a simple calculator. Two editable input boxes and one non-editable output box have been implemented along with several buttons to support arithmetic operations (addition, subtraction, multiplication). The operations work by grabbing the values in the two input boxes, performing the arithmetic operation and placing the result into the output box.

You will extend the functionality of this program by adding another button to support division. To do this, you will need to add code to create and configure the button, implement its callback function, and demonstrate the running program to your lab instructor.

3 Activities

Clone the project code for this lab from GitHub using the following URL: https://github.com/cbourke/CSCE155-C-Lab14.

3.1 Getting Familiar with Linux, GTK

For this lab you will need to use the SUSE Linux operating system. All the lab computers are dual-boot. Follow the instructions to login to the SUSE partition on your computer.

1. Login to your account using Linux
   a) Press Alt-Control-Delete as if you were going to login.
   b) Click the "restart" red button at the bottom right and restart the computer
   c) When it restarts, an option appears: select openSUSE
   d) Login with your cse login credentials
   e) You can open a terminal or multiple terminals (which is very much like putty) by clicking the terminal icon at the bottom left
   f) You can edit your code using the usual utilities or use a WISIWYG editor called gvim by entering gvim at the command line; example: `gvim gtk_calculator.c`
2. Build the application by typing `make` and run the resulting executable. Try the program out on several values to get a feel for how it works.
3. Open the source file and look it over to get an idea of its design and structure.
4. Close the calculator program and answer the questions from your worksheet.

3.2 Modifying the Program

You will now modify this program to support division.

1. Create a new division button:
   a) In the `main` function, declare a `GtkWidget` pointer for the division button
   b) Set the pointer to a new button by calling the `gtk_button_new_with_label` function
   c) Make sure that the button is visible by calling the `gtk_widget_show` function
   d) Add the new button to the layout by using the `gtk_box_pack_start` function
2. Make the new button functional by defining and associating an appropriate callback function:
   a) Create a new function (call it `division`; there is already a function called `div` that you should not redefine) with the appropriate parameters. For a function to be used as a GTK callback, it must have a specific signature (return type, parameter type and list)
   b) In this function, compute the division of the two input boxes and place the result into the output box.
      - Utilize the `getInput` helper function we have provided
      - Text boxes can only handle strings, so you'll need to use `sprintf` to convert the numerical result to a string
      - Use the `gtk_entry_set_text` function to set the text of the output text box
      - Use the other methods as examples for how to accomplish all of this
   c) Register (bind) your division function to your division button by using the `g_signal_connect` function
3. Compile your program (using `make`), run it on several values to make sure it works; demonstrate your working version to a lab instructor.

4 Handin/Grader Instructions

1. Hand in your completed files:
   - `gtk_calculator.c`

- `worksheet.md`

through the webhandin (https://cse-apps.unl.edu/handin) using your cse login and password.

2. Even if you worked with a partner, you *both* should turn in all files.
3. Verify your program by grading yourself through the webgrader (https://cse.unl.edu/~cse155e/grade/) using the same credentials.
4. Recall that both expected output and your program's output will be displayed. The formatting may differ slightly which is fine. As long as your program successfully compiles, runs and outputs the *same values*, it is considered correct.

5 Advanced Activity (optional)

Test how your program reacts to certain "bad" inputs: if you enter non-numeric characters in the input, or attempt to divide by zero. How might you prevent and/or handle these bad inputs? One solution is to simply display **ERROR** (red, bold) in the output box. Use the "Calculator!" label as a model and read GTK's API Documentation on styling to understand how text elements are "decorated". Add additional code to check for division by zero and instead of placing an answer in the output box, place **ERROR** in red, bold font.