Lab 8: CMake Revisited and GUIs
CSE 2100-001

Natnal Kebede

April 8, 2017

Date Performed:	April 6, 2017
Partners:	Natnael Kebede
	Asif khan

1 Objective

Watch both videos posted on YouTube for Lab-8. The first video shows how you should organize your code when working on small to medium sized projects with cmake. The second video walks through building a dummy user interface.

Look at the simple GUI and corresponding code in the GUI_Calculator/ Simple_Calculator folder. Familiarize yourself with the directory structure. Look at the CMakeLists.txt file. Study the main.cpp and global.h codes. Build the simple calculator from the build subdirectory (cmake .. ; make).

Look at the more useful calculator in GUI_Calculator/ Calculator. This is a sample of what we expect you to create. We are providing the skeleton for both the GUI builder and the actual GUI code in GUI_Calculator/ Skeleton_Calculator. Use the build directory for building only, any edits should be done in the *src* and *include* directories. After changing the glade file in the src directory, the
cmake ..
command in the build directory will copy the glade file over into the build directory. A clean build can be obtained by issuing the command:
*rm -rf **
from within the build directory. Be careful with this command as it erases everything from inside the directory in which it was issued!

In addition to turning in a completed version of this document on blackboard, you will need to turn in a .tgz archive of the completed calculator from the Skeleton_Calculator folder. Once you have a working calculator with which you are happy, make sure that your build directory is empty (from inside the build directory, issue: *rm -rf **). Then from the main Skeleton_Calculator directory issue the

*tar -cvzf - * > ~/My_Calculator.tgz*
command. This will create the archive My_Calculator.tgz in your home directory. You will need to upload this to blackboard.

1.1 Definitions and Quick Questions

GUI Builder: : it's also known as a graphical user interface builder and allows the creation of GUIs by letting the user arrange widgets using a drag-and-drop and modifying them with graphic tools.

pkg-config: : is a computer program used to retrieve information about installed libraries in the system. It is essentially used to compile and link against one or more libraries providing information about include- directories, libraries and their required compile flags.

The two parameters of pkg_check_modules in CMakeLists.txt: : The main interface between pkg-cong and autoconf is the package check modules macro.it provides a way to check for the presence of a given package in the system.

tree: : is a data structure with each node possibly having a left and right child. they are used to perform efficient searching such as the sorting tree utility on the Skeleton Calculator folder (on a cleaned build).

2 Question-set 1 – CMake

Let us consider adding an object to our project (we use cmake to create Makefiles). We have defined the object in my_object.h and provide the implementation details (e.g., constructors, destructors, member functions) in my_object.cpp. Where in the directory structure would you place these two files?

The .h header file will go to the include file directory and the .cpp file goes to the source file in the directory.

Let us further assume that in the implementation of your new object you are using functions from a third party library called libmatrix. You know that the header you are including for this (matrix.h) is located in /usr/include/libmatrix/. Since this is a library the linker should also link libmatrix.a which is located in /usr/lib/libmatrix/. How would you need to modify the CMakeLists.txt file (include folders, library folders, linkables)?

first we need to identify the version being used in the language (the minimum requirement for cmake is version2.6).

then we include directories using include. after that set(SOURCES/ matrix.cpp src/worker thread.cpp src/main.cpp) followed by file(GLOBSOURCES"src/".cpp) and then add excetuable (Matrix/SOURCES) to create a subdirectory. this will create a subdirectory build, include and complete the source file.

Digging more into libmatrix, you find that it came with a pkg-config description. In this case, how would you need to modify the CMakeLists.txt file (include folders, library folders, linkables)?

config-file
"PROJECT SOURCE DIR/MatrixCong.h.in"
"PROJECT BIRNARY DIR/MatrixCong.h"

3 Question-set 2 – GUIs

You use glade to make a user interface and write c++ code that loads it, displays it, and uses it. It works perfectly on your computer. You give the executable to your buddy who is running the same OS that you do. He complains that your code throws an error when starting up. What likely happened?

The use of multiple system mediums can result in the files getting corrupted easily. There are a lot of reasons why errors might occur. some of them are Reserved filenames, incompatible characters for operating systems. additionally, files that aren't synced properly due to system requirements can prevent us from accessing them.

Should any of your event handling callback functions contain infinite loops or sleep (or sleep-like) statements? Why?

The lab demonstrated that the GUI's were sequentially ordered leading to the creation of different events. the functions will implement the various events in order to register them with libraries. Once we have everything set up in the main code and with the user interface, the interface will register callbacks with infinite loop execution over the GUI library. The interaction will be going over OSM signals to the GUI library. Then library will then translate the signals and call the registered callback functions. due to these reasons the an infinite-loop would be useful

You made a GTK+3 based user interface (i.e., in your main code you turn over execution to GTK by calling gtk_main()). Do some research and describe what the use of the gdk_threads_timeout() function could be (hint: timed events). Can you think of (and describe) a specific scenario where that function (and events created by it) could

be useful?

Gtk timeout() functions are basically called called every few milliseconds. The first argument gets called whenever the timeout occurs and The second argument is numbers of the milliseconds between call to the method. Using this function we can create a timeout function that will be called every interval milliseconds.