

# Lab 8: CMake Revisited and GUIs

## CSE 2100-001

Apar Pokhrel

October 22, 2019

Date Performed: October 15 , 2019  
Partners: Apar Pokhrel  
Bivash Yadav

### 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 swalks 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:** Graphical User Interface Builder is a development tool that simplifies the creation of GUIs by allowing the designer to arrange widgets through a drag-and-drop interface

**pkg-config:** a program what defines and supports a unified interface for querying installed libraries. It is mostly used to retrieve information about installed libraries in the system and compile and link against one or more libraries.

**The two parameters of `pkg_check_modules` in `CMakeLists.txt`:** The two parameters are `PREFIX` and `QUIET`.

**tree:** In most OS, tree is a recursive directory listing program that produces a depth and indented listing of files. It lists the files in the current directory and displays the contents of directories in a tree-like format.

```
build
CMakeLists.txt
include
    global.h
src
    calculator.glade
    calculator.glade
    global.cpp
    main.cpp
```

## 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 two files `object.h` and `object.cpp` should be placed in the `build` 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)?

Matrix.h should be include in the include directories.

```
INCLUDE(CheckIncludeFiles)
CHECK_INCLUDE_FILES(matrix.h HAVE_MATRIX_H)
...
//object.cpp

include "object.h"

#ifdef HAVE_MATRIX_H

include [matrix.h]

else

include [stdlib.h]

endif

void function()
```

**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)?**

find\_package(pkgConfig) should be used.

PkgConfig has -L(for lib files .lib/.dll etc)

```
target_link_libraries(...LIB)
target_include_directories(...PUBLIC DIRS)
target_compile_options(...PUBLIC OTHER)
```

### 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?**

Glade might not have been installed in his current Operating System.

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

Yes. Event handling callback functions should contain infinite loops because some functions are called when a boolean value of false is returned.

**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()`/`dk_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?**