CMAKE

Cmake is an open source software tool for managing the build process of software using a compiler-independent method.

Cmake is used for creating a build environment. It takes CMakeLists.txt as input, and generates a *MakeFile*. *Make* is used for building. On Windows, something like *msbuild* might be used instead of *make*.

*CMakeLists.txt* files are were configurations should be placed.

*CMakeLists.txt*

message(“creating build…”) # print to terminal

cmake\_minimum\_required(VERSION 3.8) # minimum cmake version required

project(projectName) # set the project name

set(CMAKE\_CXX\_STANDARD 17) # specify c++ standard

set(myName ali) # create variable

message(${myName}) # use variable

if (myName) # conditionals

message(“myName is set”)

endif()

if(NOT WIN32) # conditional negation

message(“This is running on windows”) # WIN32 built in variable

endif()

file(COPY assets DESTINATION ${CMAKE\_BINARY\_DIR}) # copy dir assets to bin dir

include\_directories(${PROJECT\_SOURCE\_DIR}/includes) # specify include dir

add\_executable(projectName main.cpp) # list source files to create executable

*bash*

# To make and build your executable, do the following to place the executable in a build/ directory.

1. mkdir build && cd build
2. cmake ..
3. cmake –build

CREATING A LIBRARY

Instead of creating an executable, you could instead create a static or dynamic library. A static library is created by default. Static libraries are .lib extension on windows, .a on linux systems. A shared library (or dynamic), is .dll on windows, .so on linux.

Instead of *add\_executable*, use *add\_library*:

add\_library(libraryName STATIC functions.cpp other.cpp) # STATIC (default) is optional

add\_library(libraryName SHARED functions.cpp other.cpp) # dynamic lib

ADD A LIBRARY

To add a library, and therefore use it, use *add\_subdirectory*. Sub-directories must of course have their own CMakeLists.txt files. For example; say we had a lib directory which was a static library:

add\_subdirectory(lib) # the CMakeLists.txt file must exist in this dir

add\_executable(projectName main.cpp)

target\_link\_libraries(projectName PUBLIC lib/name.lib) # .lib or location to cmake file

USING CMAKE IN VS

You can use cmake on Windows too exactly the same way. Instead of using *make* to build, use *msbuild*. This will create visual studio files such as .sln, .proj, etc. Open the .sln file to open the project in Visual Studio. Right click and build. To execute, right click project, ‘set as startup project’, then click run (or f5).

In the build dir:

cmake --build . # . is the path to .sln

you can also specify config

cmake –build . –config release # release, debug, etc.

You can even open the solution in VS as if it was native => right click folder directory > ‘open in VS’.