Exercise 0

CMake Tutorial

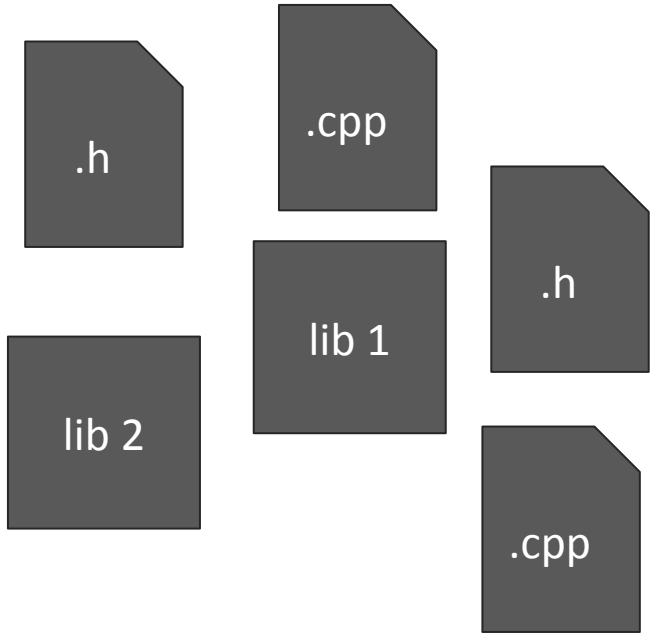
CMake

- Not a build system, but generates another system's build files (e.g. GNU Make, MSVC Toolset)
- Instructions to CMake are written in CMakeLists.txt
- Instructions specify how to create targets (executables, libraries) from project source files. These instructions are then translated into particular build system files to be executed independently of CMake

CMake

1 Take

Source files, header files, libraries



2 Specify

How to combine them to produce the build results ("targets") in a CMakeLists.txt

3 Generate

The project files for a desired build system (e.g. Visual Studio project for MSVC Toolset or a Makefile for a GNU Make build system)

4 Build

The libraries and the executables of the project using the chosen build system

CMake - Linux (GNU Make)

- Create or copy the source code directory
- Create a build directory for cmake output, cd to it
- Run **cmake <source directory>** from the build directory
- Run **make** to build project (targets specified in CMakeLists.txt)
- Run **make install** to install the targets in a directory specified by CMAKE_INSTALL_PREFIX (is set by default, don't forget to overwrite)

CMake - Windows (MSVC Toolset)

- Use Cmake GUI
- Specify source and build directories
- Choose the generator (a particular compiler from MSVC Toolset)
- Configure & Edit CMake variables & Generate
- Open generated project in your favorite IDE

CMake - Windows (MSVC Toolset)

Creates auxiliary projects:

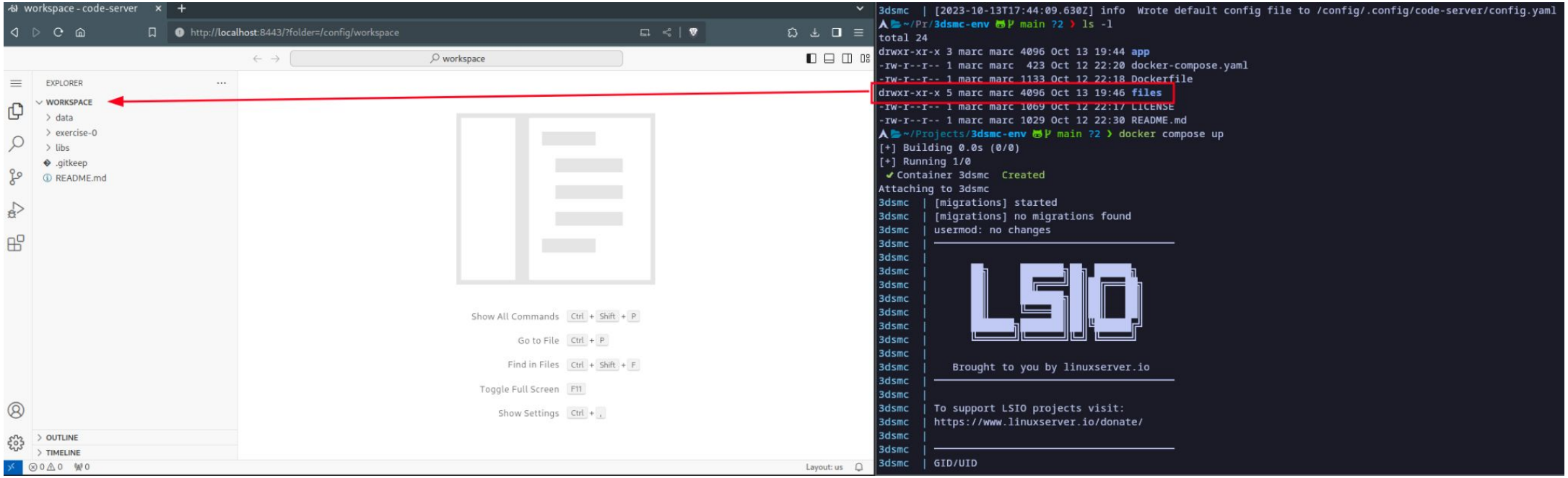
- ALL_BUILD - build all the projects in the active solution (like **make all**)
- ZERO_CHECK - building the project effectively reruns cmake (to do after a change in CMakeLists.txt)
- INSTALL (optional) - building the project installs the build targets to the path specified by CMAKE_INSTALL_PREFIX (like **make install**)

CMake - Installing 3rd-party Libraries (e.g. Eigen)

- First get the source files - get latest stable release (no suffixes) https://gitlab.com/libeigen/eigen/-/releases/
- Create a build directory for CMake to store outputs
- Specify installation directory by choosing to
  - CMake GUI: Configure; edit CMAKE_INSTALL_PREFIX; Generate
  - run cmake -D CMAKE_INSTALL_PREFIX:PATH=”PATH”
- (In case of Eigen disable BUILD_TESTING to skip generating tests)
- Build generated build files (build ALL_BUILD project or make all)
- Install (build INSTALL project or make install)

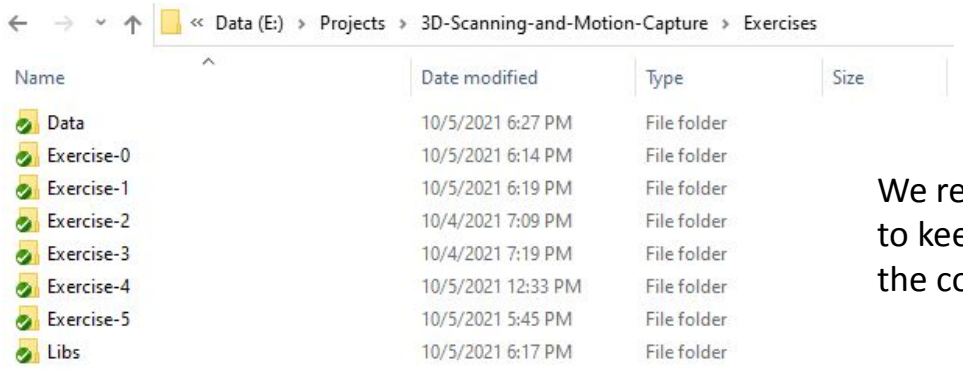
Docker environment

- https://github.com/marcbenedi/3dsmc-env
- Safe environment to install dependencies



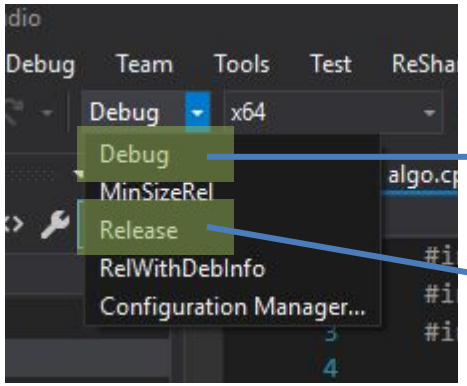
Exercise 0

- Download Exercise-0.zip from Moodle
- Get familiar with CMakeLists.txt
- Install Eigen library
- Build the project & run created executable
- Save the output of the executable in a .txt file
- Submit the .txt file to Moodle



We recommend the following file structure to keep libraries and data independent of the code base

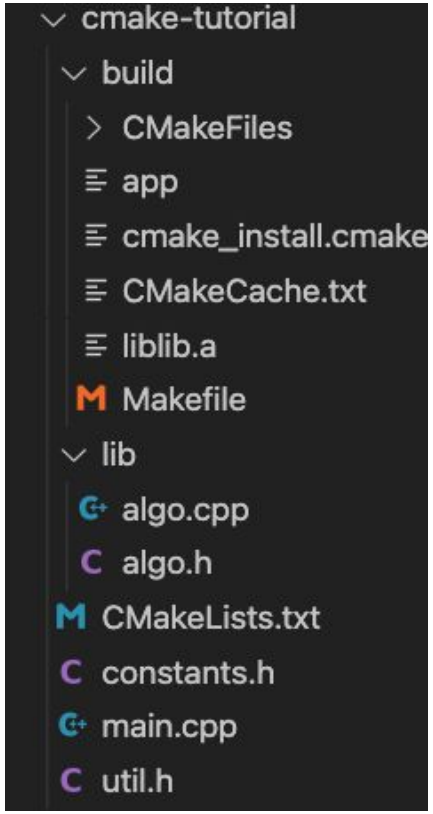
Debug/Release configuration (Visual Studio)



Use the Debug configuration to debug your code → Slow!

Use the Release configuration to run your code → Much faster!

CMake - Linux (GNU Make)



cmake-tutorial

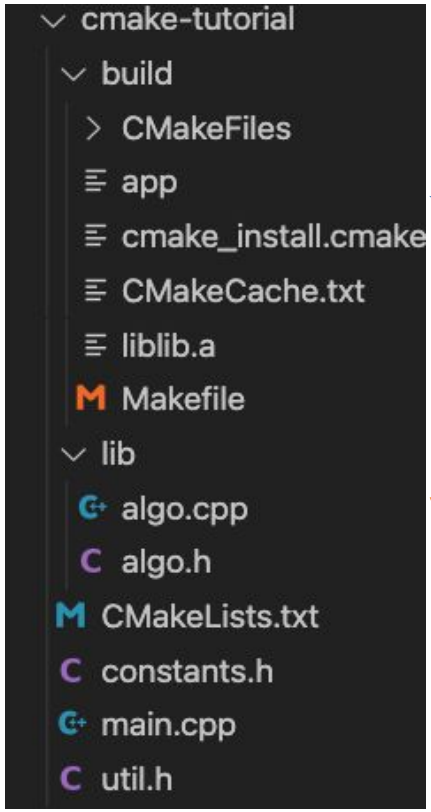
```
LIST(APPEND HEADER_FILES "util.h" "constants.h")
LIST(APPEND SOURCE_FILES "main.cpp")

# Define an executable with a name and the files
add_executable(app ${SOURCE_FILES} ${HEADER_FILES})

target_include_directories(app PUBLIC "lib")
target_link_libraries(app PUBLIC lib)
```

```
LIST(APPEND LIB_FILES "lib/algo.h" "lib/algo.cpp")
add_library(lib STATIC ${LIB_FILES})
```

CMakeLists.txt

Check CMake Tutorial for more info
https://cmake.org/cmake/help/latest/guide/tutorial/index.html

CMake - Linux (GNU Make)

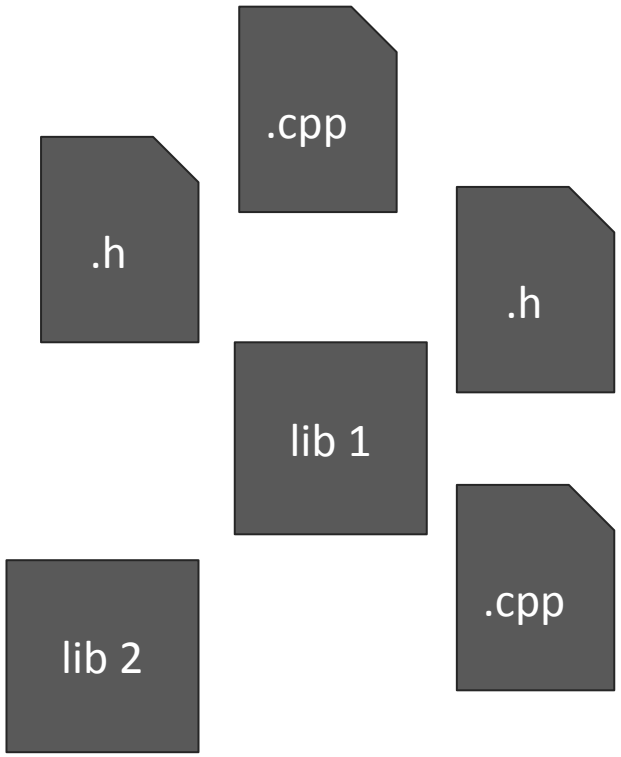


- Projects are built separately

```
Scanning dependencies of target lib
[ 25%] Building CXX object CMakeFiles/lib.dir/lib/algo.cpp.o
[ 50%] Linking CXX static library liblib.a
[ 50%] Built target lib
Scanning dependencies of target app
[ 75%] Building CXX object CMakeFiles/app.dir/main.cpp.o
[100%] Linking CXX executable app
[100%] Built target app
```

See you next time!

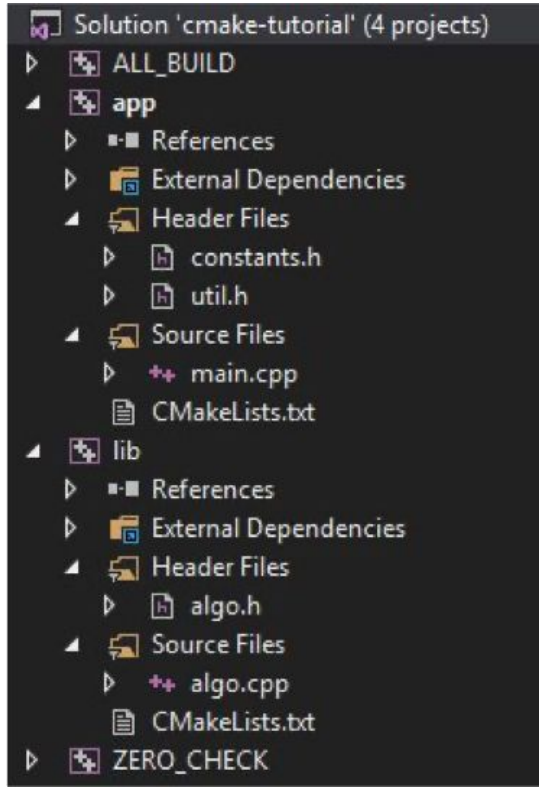
CMake - Windows (MSVC Toolset)



Dependencies, C++ standards, etc.

CMakeLists.txt

CMake

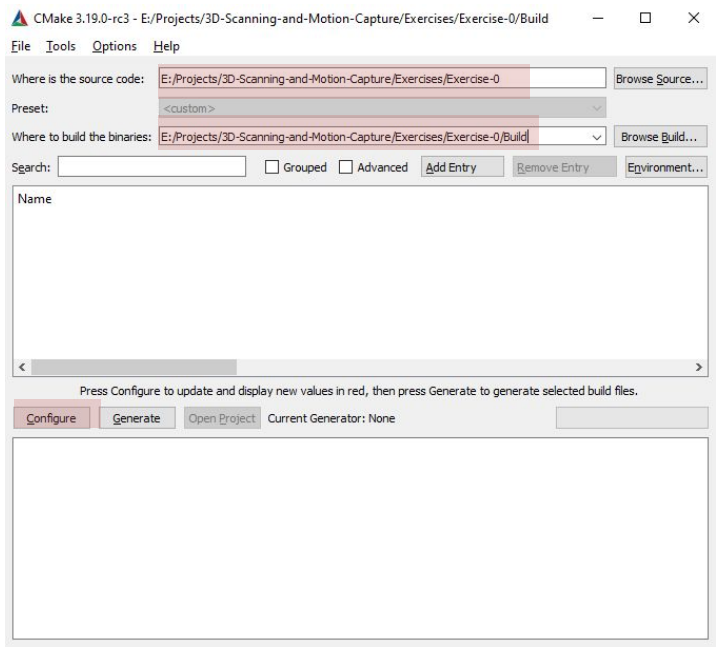


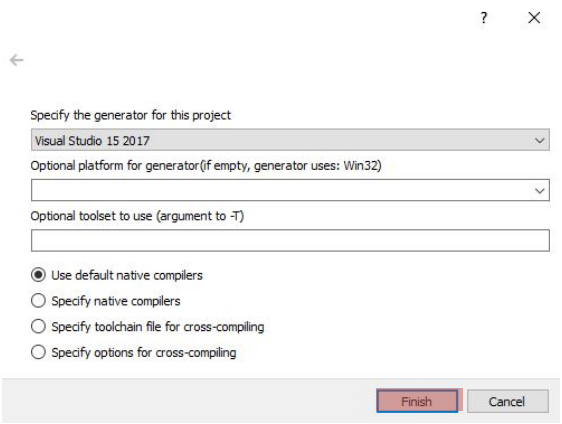
Visual Studio 2017

CMake GUI

1)

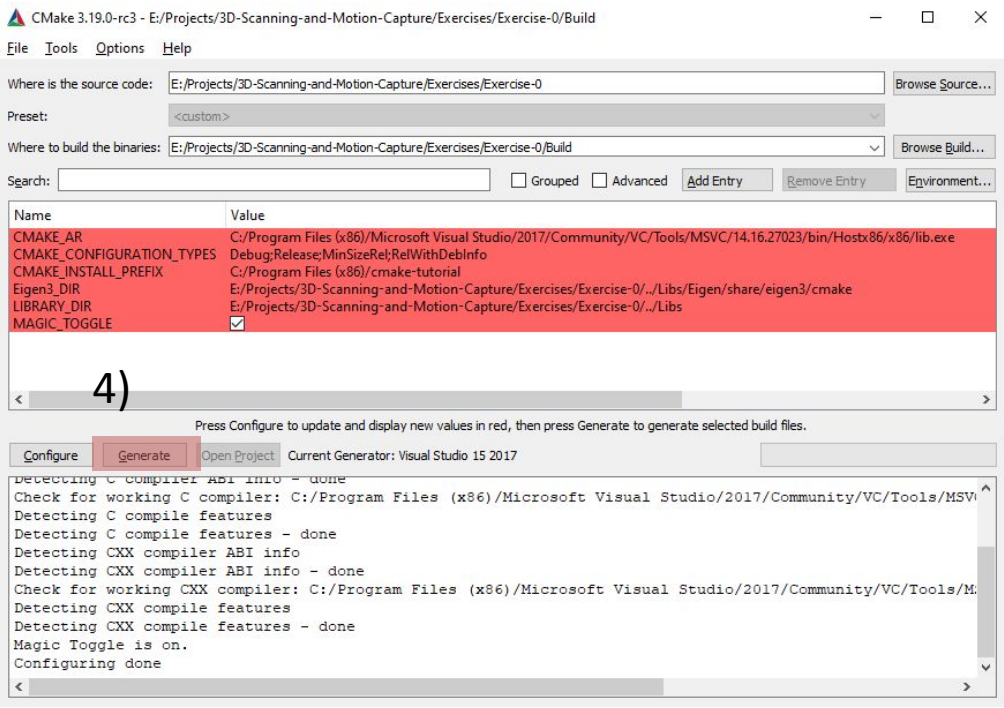
2)



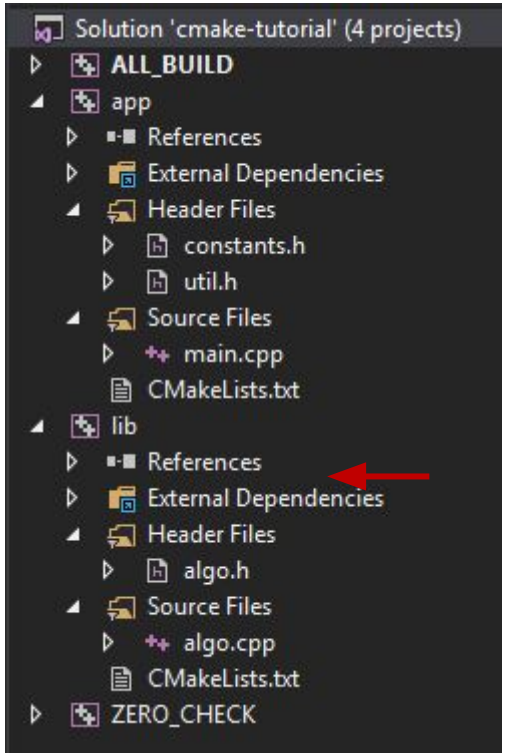


3)

4)



CMake - Windows (MSVC Toolset)



cmake-tutorial-bin/cmake-tutorial.sln

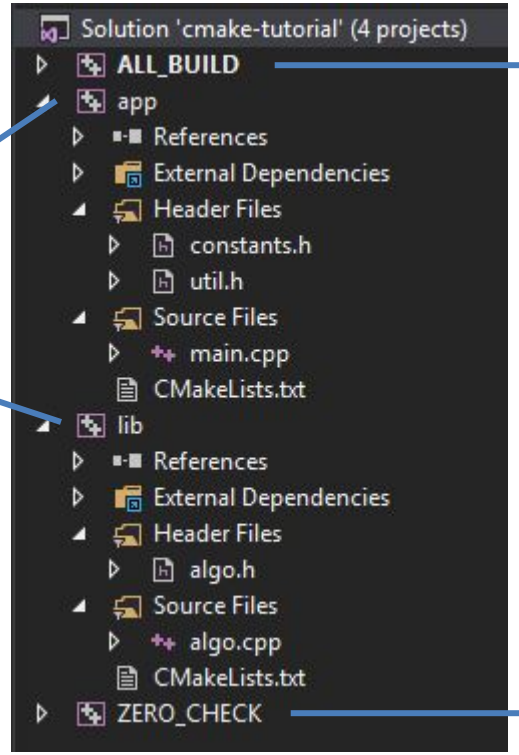
```
LIST(APPEND HEADER_FILES "util.h" "constants.h")
LIST(APPEND SOURCE_FILES "main.cpp")

# Define an executable with a name and the files
add_executable(app ${SOURCE_FILES} ${HEADER_FILES})

target_include_directories(app PUBLIC "lib")
target_link_libraries(app PUBLIC lib)
```

CMakeLists.txt

```
LIST(APPEND LIB_FILES "lib/algo.h" "lib/algo.cpp")
add_library(lib STATIC ${LIB_FILES})
```

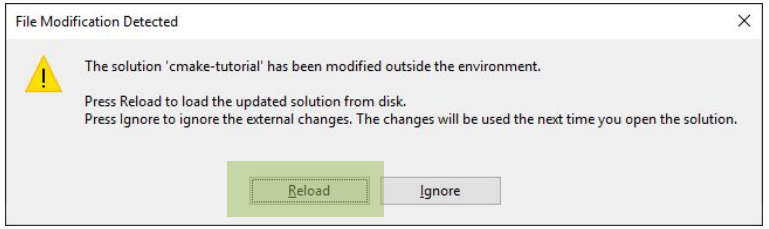
Check CMake Tutorial for more info
https://cmake.org/cmake/help/latest/guide/tutorial/index.html

CMake - Windows (MSVC Toolset)



This project builds all projects (i.e., app & lib)

Projects can be built separately



Check for changes in CMakeLists.txt by "building" this project

`cmake-tutorial-bin/cmake-tutorial.sln`