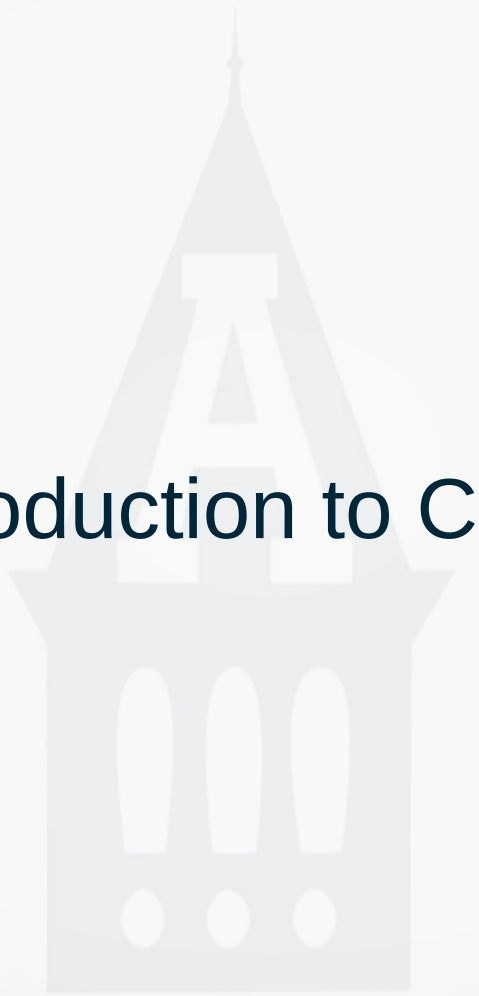
CS 3460



Introduction to CMake

What is CMake

“CMake is an extensible, open-source system that manages the build process in an operating system and in a compiler-independent manner. Unlike many cross-platform systems, CMake is designed to be used in conjunction with the native build environment. Simple configuration files placed in each source directory (called CMakeLists.txt files) are used to generate standard build files (e.g., makefiles on Unix and projects/workspaces in Windows MSVC) which are used in the usual way. CMake can generate a native build environment that will compile source code, create libraries, generate wrappers and build executables in arbitrary combinations. ”

What is CMake

- It is a meta-build system, it is not a build system itself
- It generates platform specific build systems
  - Visual Studio solution
  - makefiles
  - XCode project
  - etc

CMake Tools

- Command line utility
  - provides the full capability
  - I use this on Linux
- Graphical User Interface
  - provides the full capability
  - I use this on Windows, Linux, and macOS
- Installation
  - Be sure to install version 3.12 or greater, for C++ 20 support

CMake Hello World

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)
add_executable(HelloWorld main.cpp)
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)
```

- Placed in a file called `CMakeLists.txt`

CMake Hello World

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)
add_executable(HelloWorld main.cpp)
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)
```

- Specifies the minimum version of CMake that may be used
- Don’t set it to higher than necessary
- While optional, more and more, CMake is requiring this

CMake Hello World

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)
add_executable(HelloWorld main.cpp)
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)
```

- Gives a name for the project
- Required
- Recommended near the top, but after CMake min version

CMake Hello World

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)
add_executable(HelloWorld main.cpp)
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)
```

- Specifies an executable target
- A single project may have multiple executable targets
- First parameter is executable name, do not put .exe or any other extension
- After first parameter, list of source files to associate with the target

CMake Hello World

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)
add_executable(HelloWorld main.cpp)
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)
```

- Set the C++ 20 standard for the target
- Depending upon the compiler, the appropriate setting is written to the platform build files

Generating Project Files

- Want to keep generated project files and original source files in separate folders

  assignment1
  assignment1/build

- Source files and `CMakeLists.txt` in *assignment1*
- Generated (by CMake) files in *assignment1/build*

Generating Project Files Demo

CMake – Compiler Options

- Best practice to enable high-level of compiler warnings
- CMake doesn't know everything about each compiler
- Need a way to specify compiler specific warnings

```
if (CMAKE_CXX_COMPILER_ID STREQUAL "MSVC")
    ...do MSVC specific stuff...
elseif (CMAKE_CXX_COMPILER_ID STREQUAL "GNU")
    ...do g++ specific stuff...
elseif (CMAKE_CXX_COMPILER_ID MATCHES "Clang")
    ...do llvm specific stuff...
endif()
```

CMake – Compiler Options

- MSVC
  - /W4
  - /permissive-
- g++
  - -Wall
  - -Wextra
  - -pedantic

CMake – Compiler Options

- MSVC
  - /W4
  - /permissive-
- g++
  - -Wall
  - -Wextra
  - -pedantic

```
if (CMAKE_CXX_COMPILER_ID STREQUAL "MSVC")
    target_compile_options(HelloWorld PRIVATE /W4 /permissive-)
elseif (CMAKE_CXX_COMPILER_ID STREQUAL "GNU")
    target_compile_options(HelloWorld PRIVATE -Wall -Wextra -pedantic)
elseif (CMAKE_CXX_COMPILER_ID MATCHES "Clang")
    target_compile_options(HelloWorld PRIVATE -Wall -Wextra -pedantic)
endif()
```

Multiple Source Files

```
add_executable(BigProject
    main.cpp
    utilities.hpp
    utilities.cpp
    simulation.hpp
    simulation.cpp)
```

Multiple Source Files – Cont.

```
set(HEADER_FILES
    utilities.hpp
    simulation.hpp)

set(SOURCE_FILES
    main.cpp
    utilities.cpp
    simulation.cpp)

add_executable(BigProject ${HEADER_FILES} ${SOURCE_FILES})
```