# CMake

## Cross platform build tool

CMake

Cross-platform Make


**Zhivko Petrov**

**A guy that knows C++**

Software University

SoftUni

Software University

https://about.softuni.bg

# Have a Question?

Software University

# sli.do

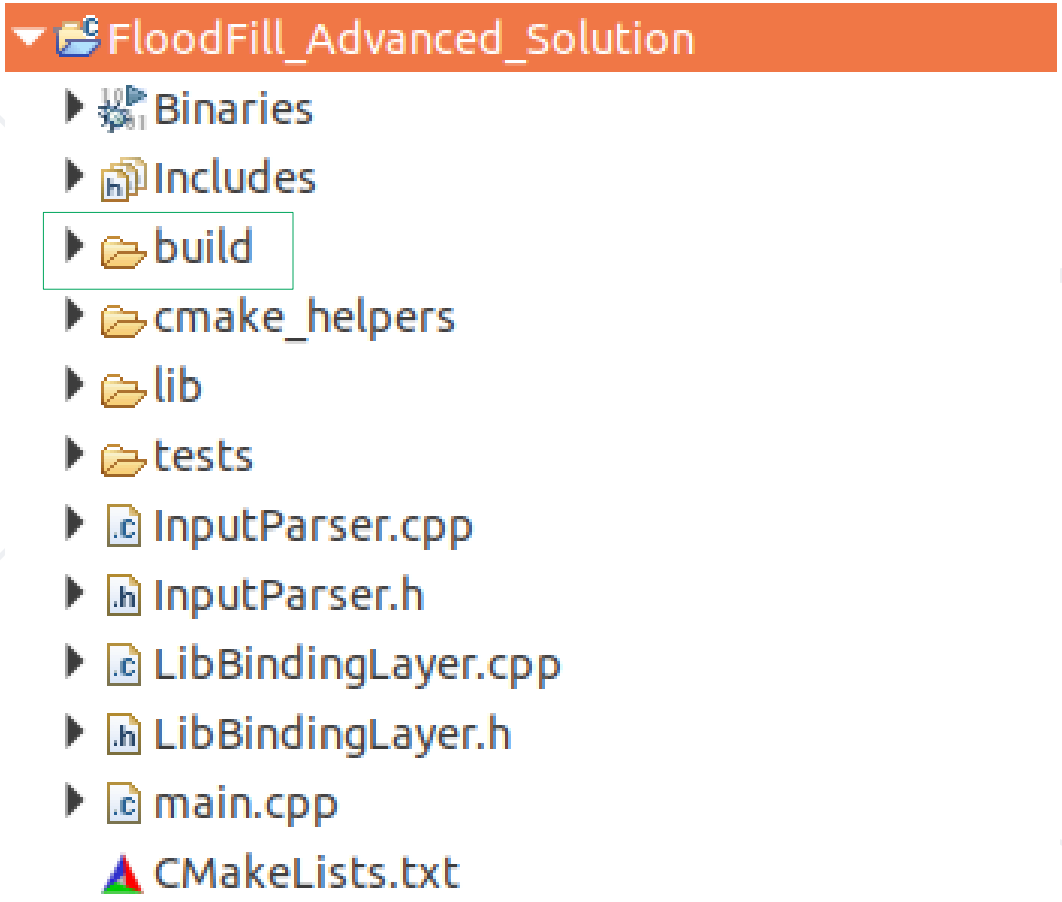
# #app-dev-cpp

# Overview

- CMake is a **cross-platform** tool that automates the building process of software C/C++ projects
- Main **Pros**:
  - Cross platform discovery of system libraries
  - Automatic discovery and configuration of the toolchain
  - Easier to compile your files into a shared library in a platform agnostic way, and in general easier to use than make
  - **Out of source build**

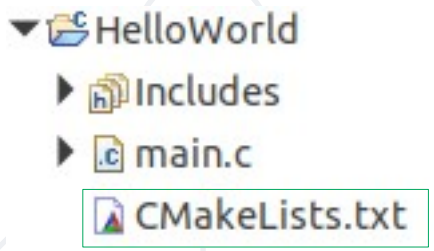
# Out of source build

- Don’t need to write ‘make clean’
- Simply **delete** the content of the build folder



# Hello, World!

- Every logical “level” of your file structure needs a “CMakeLists.txt”
- Run CMake from the **build**(external) folder



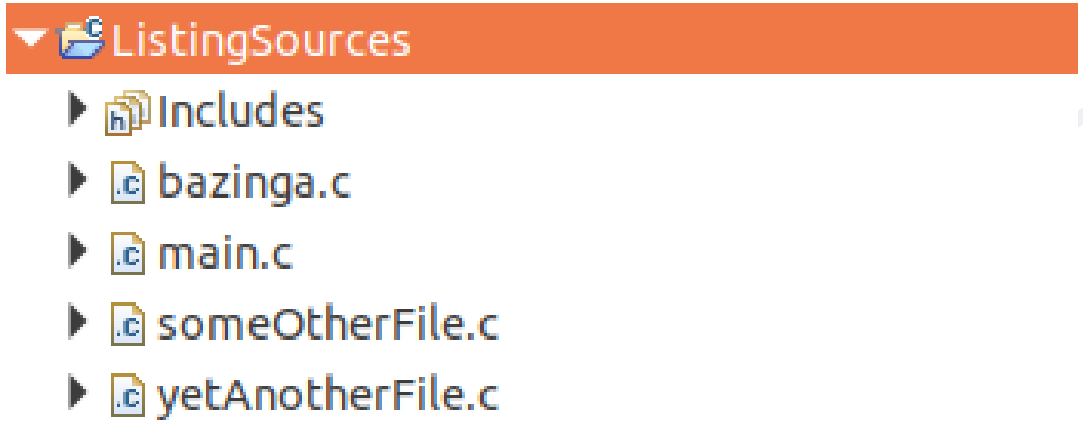

```
cmake_minimum_required(VERSION 3.5.1)

project(hello_world)

#generate project binary
add_executable(${PROJECT_NAME}
               ${CMAKE_CURRENT_SOURCE_DIR}/main.c)
```

# Listing your source files

- All compilation units part of the binary should be described
- Either glob (find all files) using some pattern
- Or Individually list every single file (preferred approach)




```
set(SRC_DIR ${CMAKE_CURRENT_SOURCE_DIR})

#generate project binary
add_executable(${PROJECT_NAME}
               ${SRC_DIR}/main.c
               ${SRC_DIR}/someOtherFile.c
               ${SRC_DIR}/yerAnotherFile.c
               ${SRC_DIR}/bazinga.c)
```

```
#file(GLOB...) allows for wildcard additions:
file(GLOB SOURCES ${CMAKE_CURRENT_SOURCE_DIR}/*c)
```

# Including directories

- Each project can add **directories** to it's include path

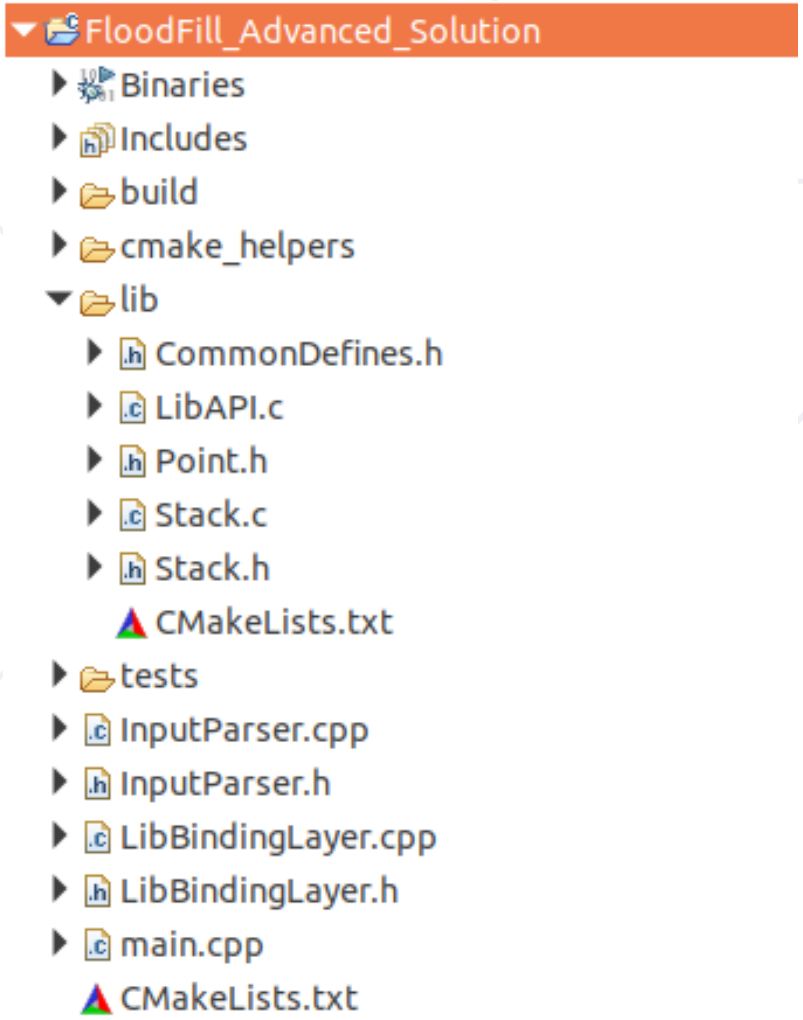
```
target_include_directories(${PROJECT_NAME} PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
```

- Access levels are:
- **PRIVATE** – only included for the project itself
- **PUBLIC** – included for itself and everyone, which links with that target (which "inherits it")
- **INTERFACE** – not included for current target but only for those, which links against it

- *include_directories()* - can be used, but considered **bad** practice

# Add subdirectory

- Each CMakeLists.txt file could invoke a child one (subproject)




```
#invoke child Cmake files
add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/lib)


cmake_minimum_required(VERSION 3.5.1)

project(solution)
```

# Linking with target

- Each project can link against other targets and directories

```
target_link_libraries(${PROJECT_NAME} PRIVATE solution)
```

- Access levels are the same: PRIVATE, PUBLIC and INTERFACE
- When a target links against other target:
- Cmake automatically handles the dependencies for you.
  solution **will be build before** ${PROJECT_NAME}
- The ${PROJECT_NAME} inherits all of “solution” PUBLIC/INTERFACE includes
- *link_directories()* - can be used, but considered **bad** practice

- NOTE: dependencies between target could be
  explicitly added
- *add_dependencies(${PROJECT_NAME} solution)*

# Functions

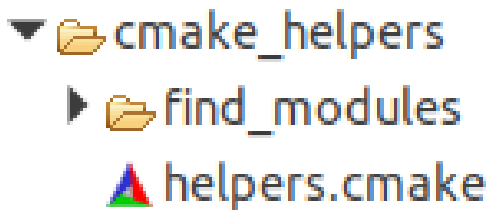
- Function can be created and used like any programming language

```
function(enable_target_warnings target)
    target_compile_options(
        ${target}
        PRIVATE
          -Wall
          -Wextra
          -Werror
          -Wuninitialized
          -Wreorder
          -Wshadow
          -Wpointer-arith
          -Wcast-align
          -Wcast-qual
          -Wconversion
          -Wunused-parameter
          -Wlogical-op
          -Wdouble-promotion
          -Wuseless-cast
          -Wnon-virtual-dtor
          -Woverloaded-virtual
          -Wduplicated-cond
          -Wduplicated-branches
          -Wnull-dereference
    )
endfunction()
```

```
function(set_target_cpp_standard target standard)
    set_target_properties(
        ${target}
        PROPERTIES
            CXX_STANDARD ${standard}
            CXX_STANDARD_REQUIRED YES
            CXX_EXTENSIONS NO
    )
endfunction()
```

# Helper files and includes

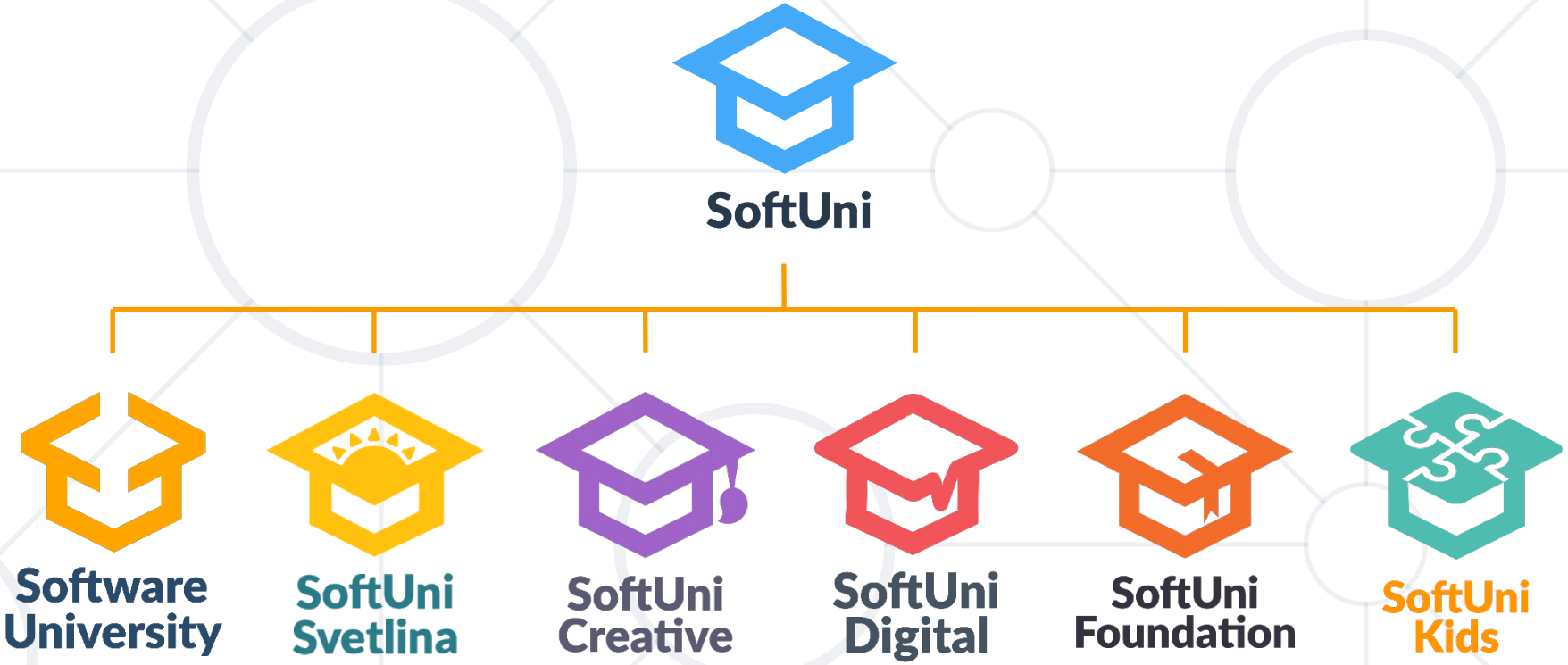
- Common functions are stored into a helpers file
- That file can be included in other CmakeLists.txt files
- From there on included methods can be reused




```
include(${CMAKE_CURRENT_SOURCE_DIR}/cmake_helpers/helpers.cmake)


set_target_cpp_standard(${PROJECT_NAME} 17)
enable_target_warnings(${PROJECT_NAME})
```

# Questions?



# Diamond Partners

Software University

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

SmartIT

XS SOFTWARE

zühlke
empowering ideas

INFRAGISTICS®

Coca-Cola HBC Bulgaria

DRAFT KINGS

Postbank
Решения за твоето утре

SOFTWARE GROUP

# Educational Partners

Software University

eee.bg

VIRTUAL RACING SCHOOL

VRS

codexio

# License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

# Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg