南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Advanced Programming

Lab 10

1 CONTENTS

- Learn CMake

2 Knowledge Points

2.1 CMake

2.1 CMake

What is CMake?

Cmake is an open-source, cross-platform family of tools designed to build, test and package software. **Cmake** is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice.

Reference:
https://en.wikipedia.org/wiki/CMake
https://cmake.org/

CMake needs **CMakeLists.txt** to run properly.

A CMakeLists.txt consists of **commands** , **comments** and **spaces**.

- The **commands** include command name, brackets and parameters ,

the parameters are separated by spaces. Commands are not case sensitive.

- **Comments** begins with '#'.

Steps for generating a makefile and compiling on Linux using Cmake:

Step1: Writes the CMake configuration file **CMakeLists.txt**.

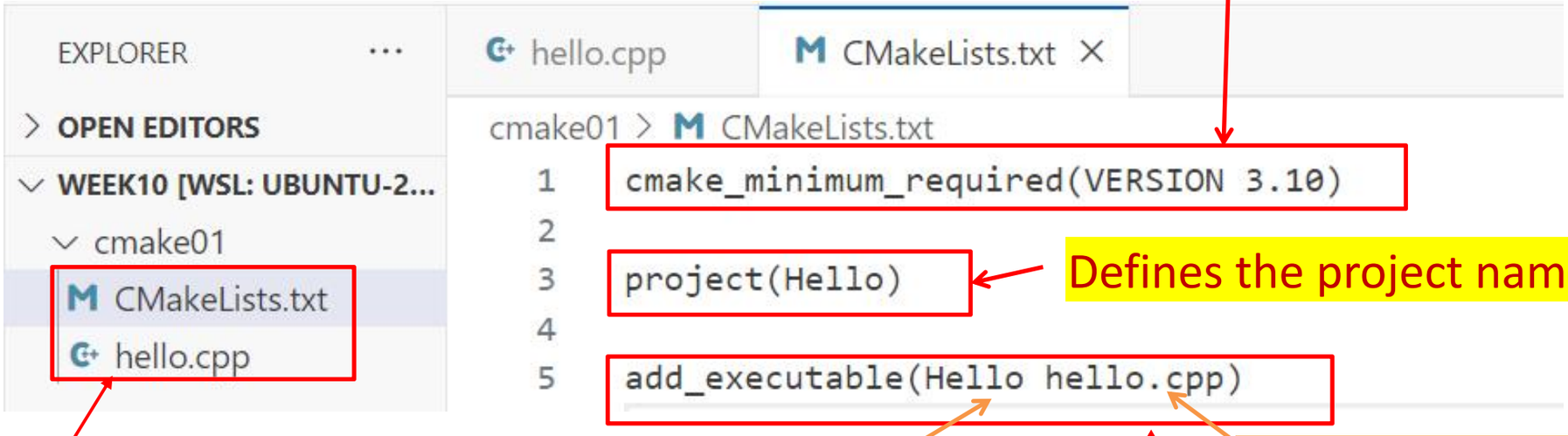
Step2: Executes the command **cmake PATH** to generate the **Makefile**.
(PATH is the directory where the CMakeLists.txt resides.)

Step3: Compiles using the **make** command.

1. A single source file in a project

The most basic project is an executable built from source code files. For simple projects, a three-line **CMakeLists.txt** file is all that is required.

Specifies the minimum required version of CMake.Use **cmake --version** in Vscode terminal window to check the cmake version in your computer.



Defines the project name.

The first parameter indicates the filename of executable file.

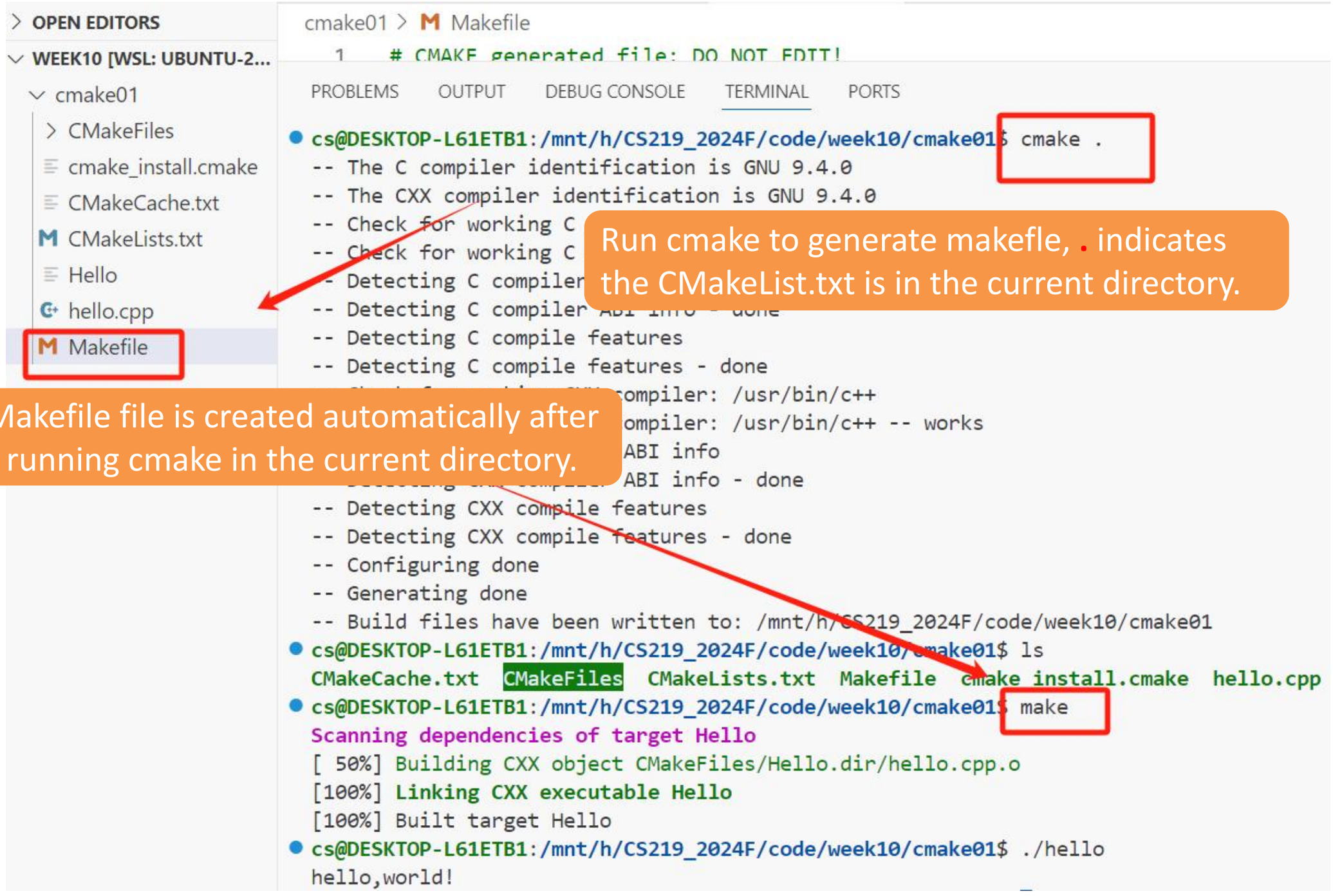
The second parameter indicates the source file.

Store the CMakeLists.txt file in the same directory as the hello.cpp file.

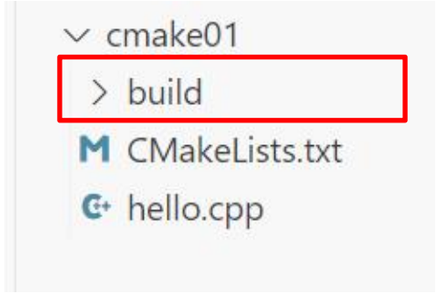
Adds the Hello executable target which will be built from hello.cpp.

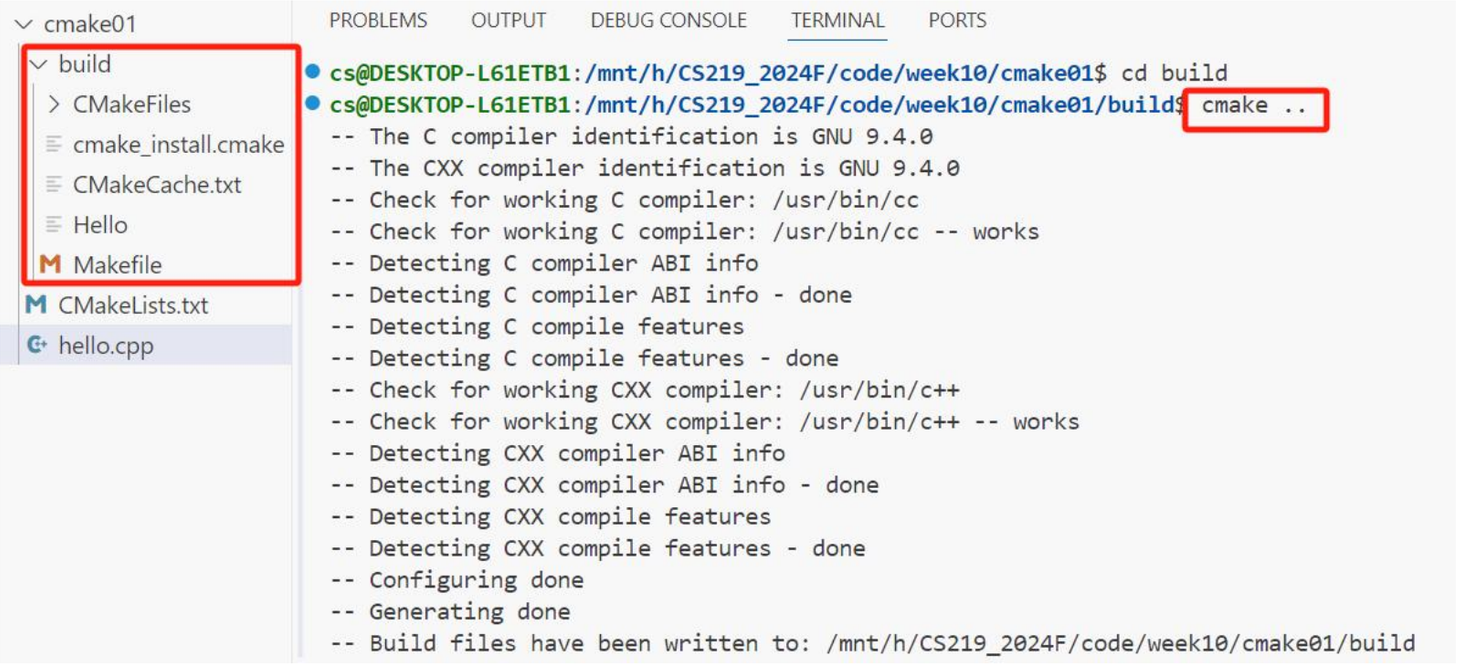
In current directory, type **cmake .** to generate makefile. If cmake does not be installed, follow the instruction to intall cmake.

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ cmake .

Command 'cmake' not found, but can be installed with:

sudo apt install cmake          # version 3.16.3-1ubuntu1.20.04.1, or
sudo apt install cmake-mozilla  # version 3.27.9-0ubuntu1~20.04

cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ sudo apt install cmake
[sudo] password for cs:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cmake-data libjsoncpp1 librhash0
Suggested packages:
  cmake-doc ninja-build
The following NEW packages will be installed:
  cmake cmake-data libjsoncpp1 librhash0
0 upgraded, 4 newly installed, 0 to remove and 146 not upgraded.
Need to get 5470 kB of archives.
After this operation, 28.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Waiting for headers]
```


OPEN EDITORS
WEEK10 [WSL: UBUNTU-2...
cmake01
CMakeFiles
cmake_install.cmake
CMakeCache.txt
CMakeLists.txt
Hello
hello.cpp
Makefile
cmake01 > Makefile
1 # CMAKE generated file: DO NOT EDIT!
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ cmake .
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C
-- Check for working C
-- Detecting C compiler
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
compiler: /usr/bin/c++
ompiler: /usr/bin/c++ -- works
ABI info
ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/h/CS219_2024F/code/week10/cmake01
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ ls
CMakeCache.txt CMakeFiles CMakeLists.txt Makefile cmake_install.cmake hello.cpp
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ make
Scanning dependencies of target Hello
[50%] Building CXX object CMakeFiles/Hello.dir/hello.cpp.o
[100%] Linking CXX executable Hello
[100%] Built target Hello
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01$ ./hello
hello,world!
Run cmake to generate makefle, . indicates the CMakeList.txt is in the current directory.
Makefile file is created automatically after running cmake in the current directory.

- Delete all the building files and directory by CMake.
- Create an empty folder to store the building files and directory by CMake.






```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01/build$ make
Scanning dependencies of target Hello
[ 50%] Building CXX object CMakeFiles/Hello.dir/hello.cpp.o
[100%] Linking CXX executable Hello
[100%] Built target Hello
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01/build$ ./hello
hello,world!
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01/build$
```

or

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01/build$ cmake --build .
Scanning dependencies of target Hello
[ 50%] Building CXX object CMakeFiles/Hello.dir/hello.cpp.o
[100%] Linking CXX executable Hello
[100%] Built target Hello
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake01/build$ ./hello
hello, world!
```

2. Multi-source files in a project

There are three files in the same directory.

```
sudo apt-get install tree
```

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10$ tree ./cmake02
./cmake02
├── factorial.cpp
├── functions.h
└── main.cpp
```

CMakeLists.txt

cmake02 > CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.10)
2
3  project(function)
4
5  add_executable(function main.cpp function.cpp)
6
```

List all the source files using space as the separator.

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10$ cd cmake02
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02$ mkdir build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02$ cd build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/h/CS219_2024F/code/week10/cmake02/build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ make
Scanning dependencies of target function
[ 33%] Building CXX object CMakeFiles/function.dir/main.cpp.o
[ 66%] Building CXX object CMakeFiles/function.dir/function.cpp.o
[100%] Linking CXX executable function
[100%] Built target function
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ ./function
The factorial off 5 is: 120
```

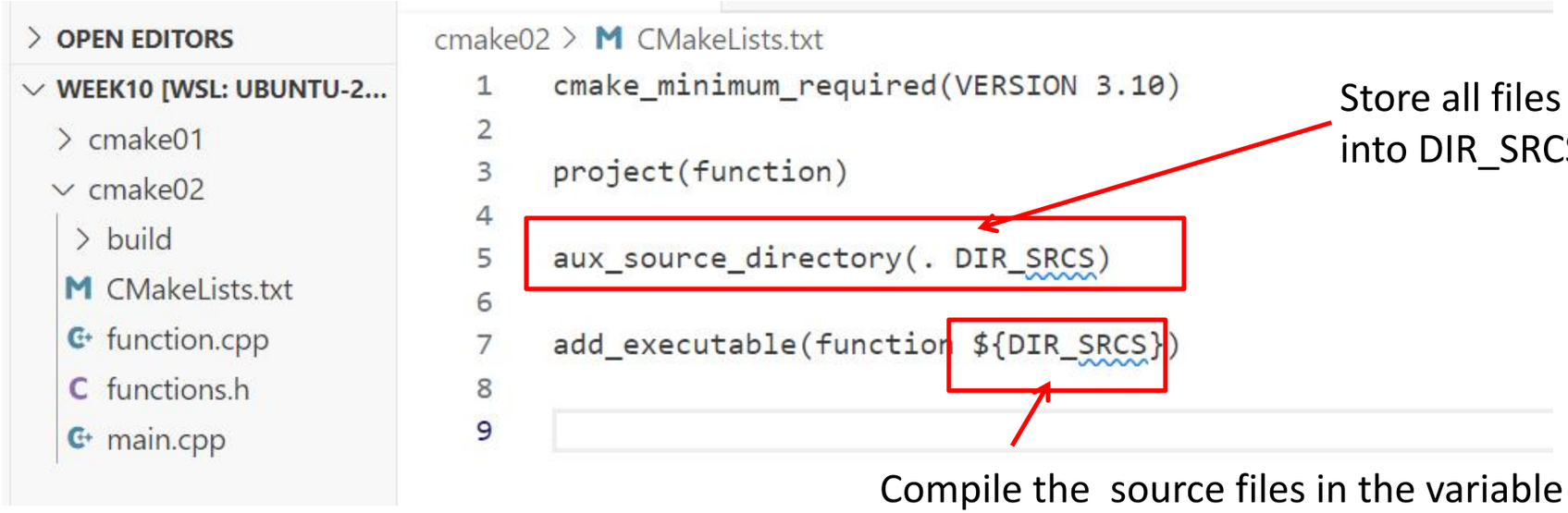
2. Multi-source files in a project

If there are several files in directory, put each file into the **add_executable** command is not recommended. The better way is using **aux_source_directory** command.

aux_source_directory (<dir> <variable>)

The command finds all the source files in the specified directory indicated by <dir> and stores the results in the specified variable indicated by <variable>.

2. Multi-source files in a project



Store all files in the current directory into DIR_SRCS variable.

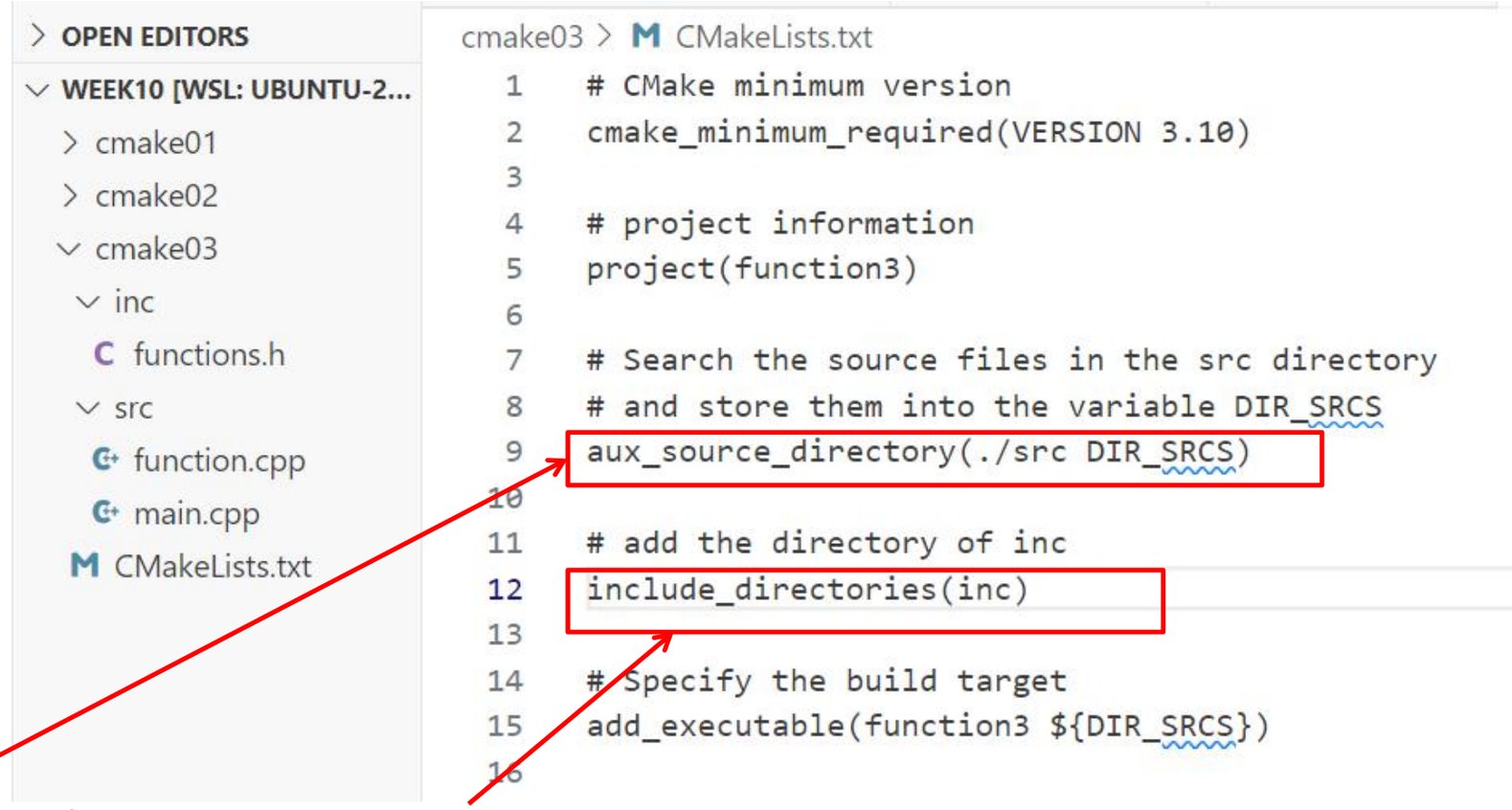
Compile the source files in the variable by **${ }** into an executable file named CmakeDemo2

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
^[[A-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/h/CS219_2024F/code/week10/cmake02/build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ make
Scanning dependencies of target function
[ 33%] Building CXX object CMakeFiles/function.dir/function.cpp.o
[ 66%] Building CXX object CMakeFiles/function.dir/main.cpp.o
[100%] Linking CXX executable function
[100%] Built target function
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake02/build$ ./function
```

3. Multi-source files in a project in different directories

```
./cmake03
├── CMakeLists.txt
├── inc
│   └── functions.h
└── src
    ├── function.cpp
    └── main.cpp
```

We write CMakeLists.txt in CmakeDemo3 folder.



All .cpp files are in the **src** directory

Include the header file which is stored in **inc** directory.

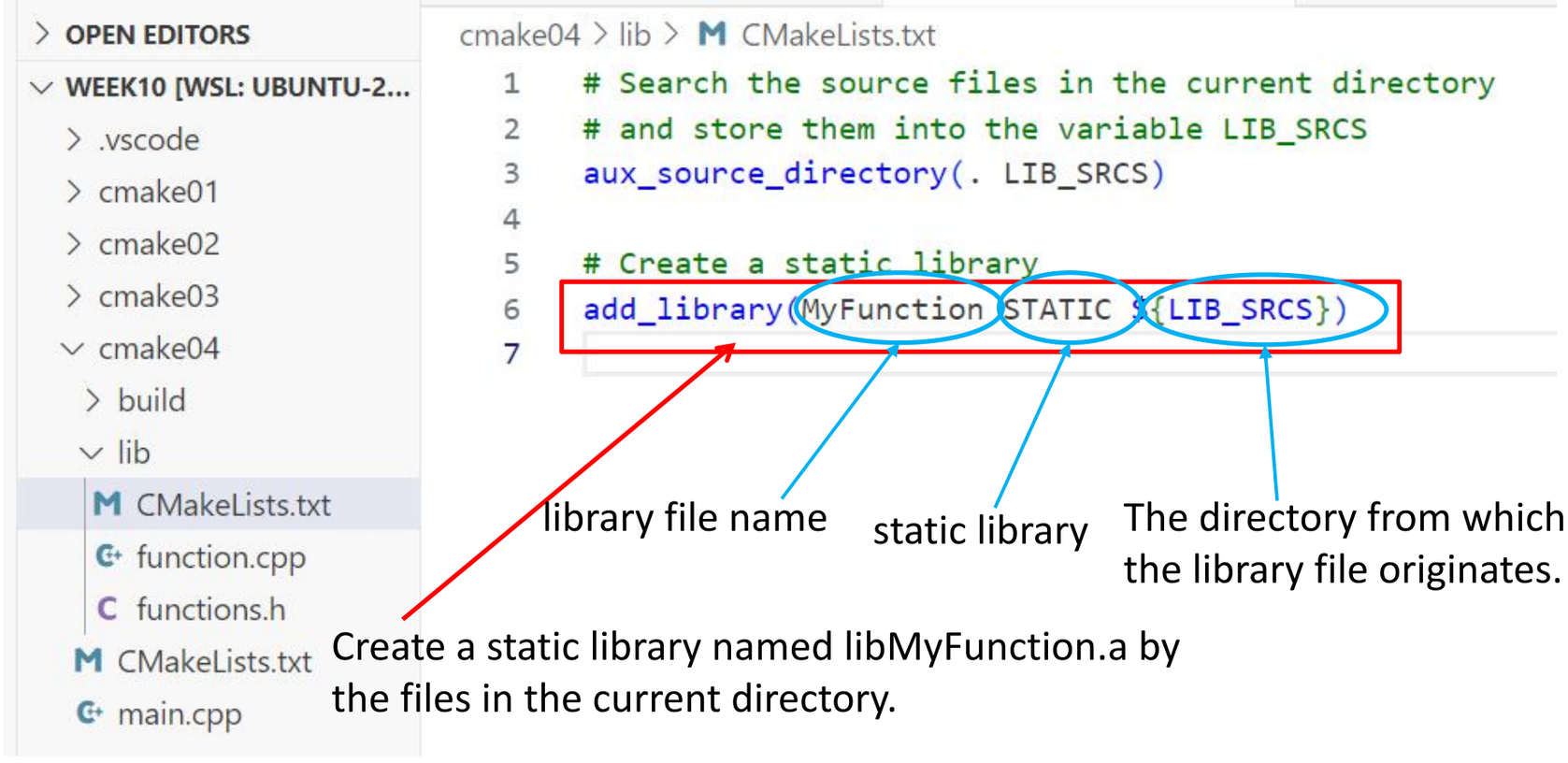
```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake03$ mkdir build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake03$ cd build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake03/build$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/h/CS219_2024F/code/week10/cmake03/build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake03/build$ make
Scanning dependencies of target function3
[ 33%] Building CXX object CMakeFiles/function3.dir/src/function.cpp.o
[ 66%] Building CXX object CMakeFiles/function3.dir/src/main.cpp.o
[100%] Linking CXX executable function3
[100%] Built target function3
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake03/build$ ./function3
The factorial of 5 is: 120
```

4. Static library Dynamic library

We want to create a static(or dynamic) library by function.cpp and call the static(or dynamic) library in main.cpp. This time we write two CMakeLists.txt files, one in CmakeDemo4 folder and another in lib folder.

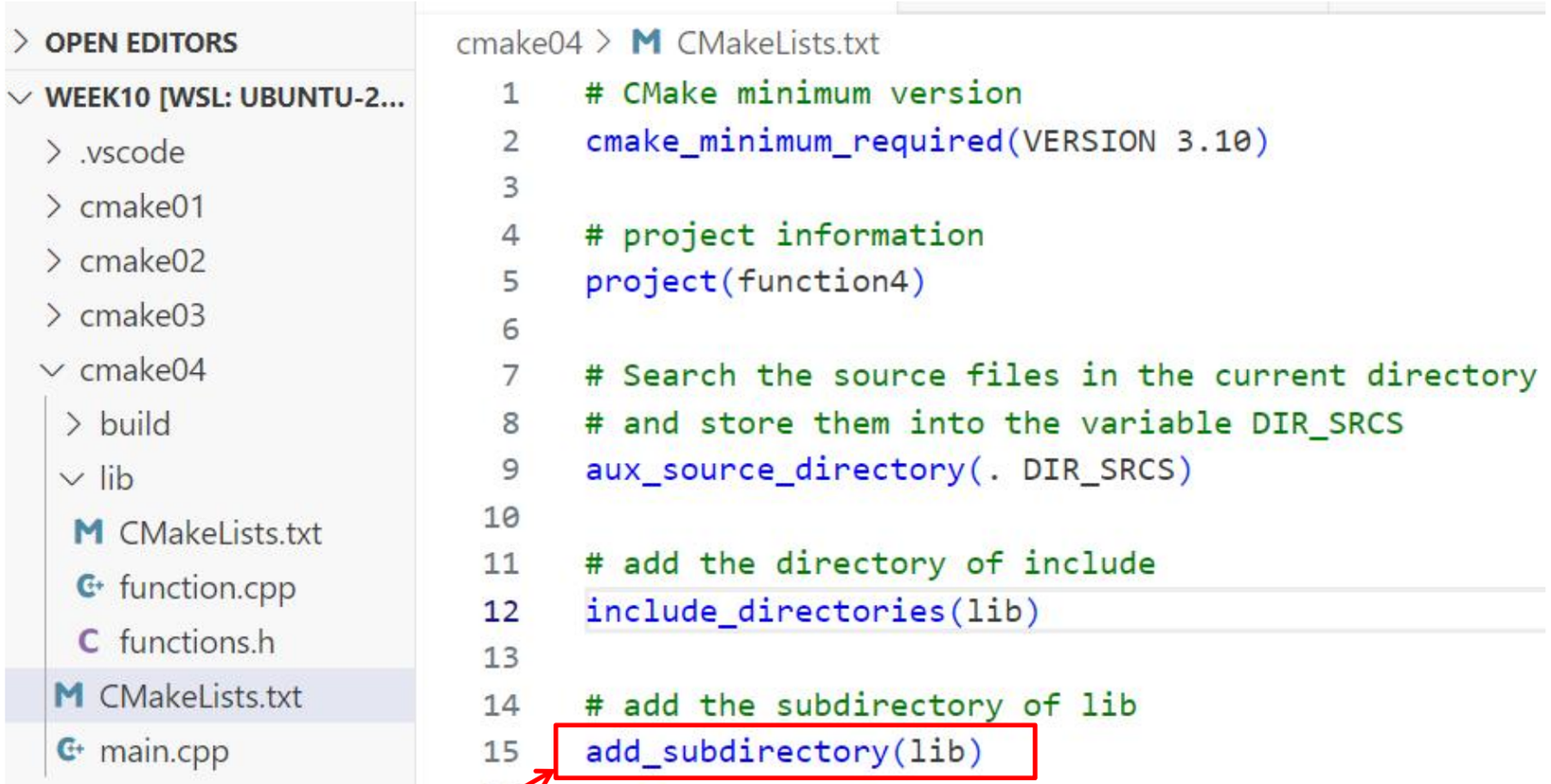
The CMakeLists.txt in lib folder creates a static library.

```
./CMakeDemo4
      |
      +--- main.cpp
      |
      +--- lib/
            |
            +--- function.h
            |
            +--- function.cpp
```



Note: If we use **SHARED** instead of STATIC in **add_library** command, it will create a shared(dynamic) library file.

The CMakeLists.txt in CMakeDemo4 folder creates the project.



add_subdirectory command indicates there is a subdirectory in the project. When running the command, it will execute the CMakeLists.txt in the subdirectory automatically.

Indicates that the project needs link a library named **MyFunction**, MyFunction can be a static library file or a dynamic library file.

project target

library file name
If there are more than one file, list them using space as the separator.

cmake04 > lib > M CMakeLists.txt

```
1   # Search the source files in the current directory
2   # and store them into the variable LIB_SRCS
3   aux_source_directory(. LIB_SRCS)
4
5   # Create a static library
6   add_library(MyFunction STATIC ${LIB_SRCS})
```

If we use **SHARED** in add_library command, there will creates a dynamic library named **libMyFunction.so** and link with it in main.

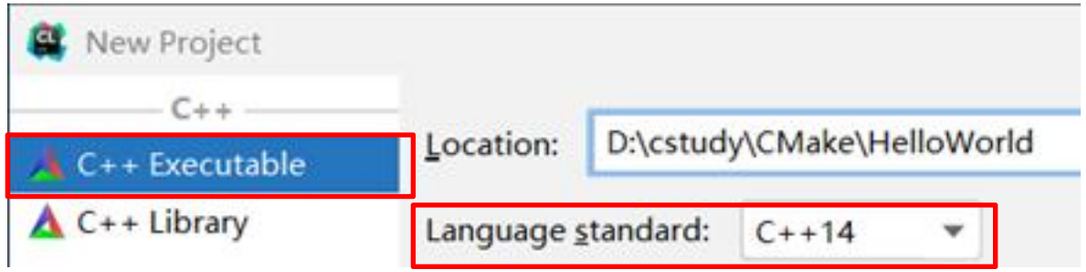
cmake04 > lib > M CMakeLists.txt

```
1   # Search the source files in the current directory
2   # and store them into the variable LIB_SRCS
3   aux_source_directory(. LIB_SRCS)
4
5   # Create a shared (dynamic) library
6   add_library(MyFunction SHARED ${LIB_SRCS})
7
```

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ make
Scanning dependencies of target MyFunction
[ 25%] Building CXX object lib/CMakeFiles/MyFunction.dir/function.cpp.o
[ 50%] Linking CXX shared library libMyFunction.so
[ 50%] Built target MyFunction
Scanning dependencies of target function4
[ 75%] Building CXX object CMakeFiles/function4.dir/main.cpp.o
[100%] Linking CXX executable function4
[100%] Built target function4
```

```
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10$ cd cmake04
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04$ mkdir build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04$ cd build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ cmake ..
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/h/CS219_2024F/code/week10/cmake04/build
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ ls
CMakeCache.txt  CMakeFiles  Makefile  cmake_install.cmake  lib
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ cd lib
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build/lib$ ls
CMakeFiles  Makefile  cmake_install.cmake
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build/lib$ cd ..
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ make
Scanning dependencies of target MyFunction
[ 25%] Building CXX object lib/CMakeFiles/MyFunction.dir/function.cpp.o
[ 50%] Linking CXX static library libMyFunction.a
[ 50%] Built target MyFunction
Scanning dependencies of target function4
[ 75%] Building CXX object CMakeFiles/function4.dir/main.cpp.o
[100%] Linking CXX executable function4
[100%] Built target function4
cs@DESKTOP-L61ETB1:/mnt/h/CS219_2024F/code/week10/cmake04/build$ ./function4
The factorial of 5 is: 120
```

Create a C++ project by CLion, the CMakeLists.txt is created automatically.



CMakeLists.txt | main.cpp

```
1  #include <iostream>
2
3  int main() {
4      std::cout << "Hello, World!" << std::endl;
5      return 0;
6  }
```

CMakeLists.txt | main.cpp

```
1  cmake_minimum_required(VERSION 3.16)
2  project(HelloWorld)
3
4  set(CMAKE_CXX_STANDARD 14)
5
6  add_executable(HelloWorld main.cpp)
```

Specifies the minimum required version of Cmake. It is set to the version of Cmake bundled in Clion (always one of the newest versions available).

Defines the project name according to what we provided during project creation.

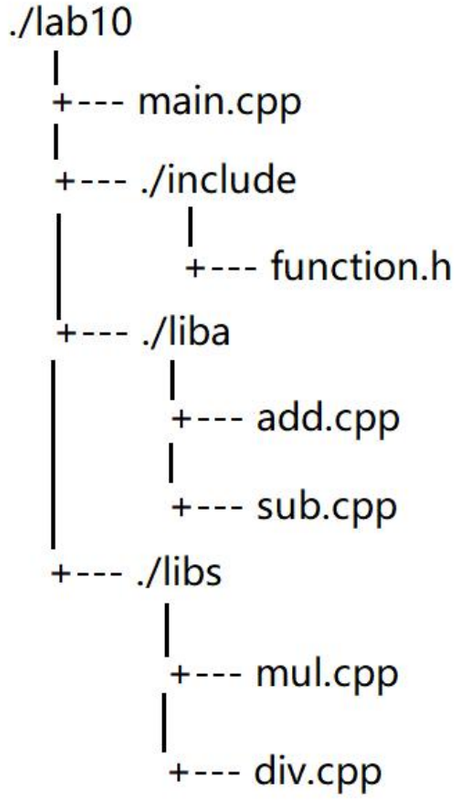
Sets the CMAKE_CXX_STANDARD variable to the value of 14, as we selected when creating the project.

Adds the HelloWorld executable target which will be built from main.cpp.

For more about Cmake(cmake tutorial):
https://cmake.org/cmake/help/latest/guide/tutorial/index.html
https://www.jetbrains.com/help/clion/2016.3/quick-cmake-tutorial.html

3 Exercises

1.Define four functions that implement the operations of addition, subtraction, multiplication and division respectively.(one function one .cpp file) Write a test program to test these functions.



According to the tree structure of the files, creates a static library with the two files in the liba directory and a dynamic library with two files in the libs directory. And then link with main.cpp. Using cmake command to compile and build your project. At last run the program.