Lecture-4

BTS of a C++ Build

Agenda

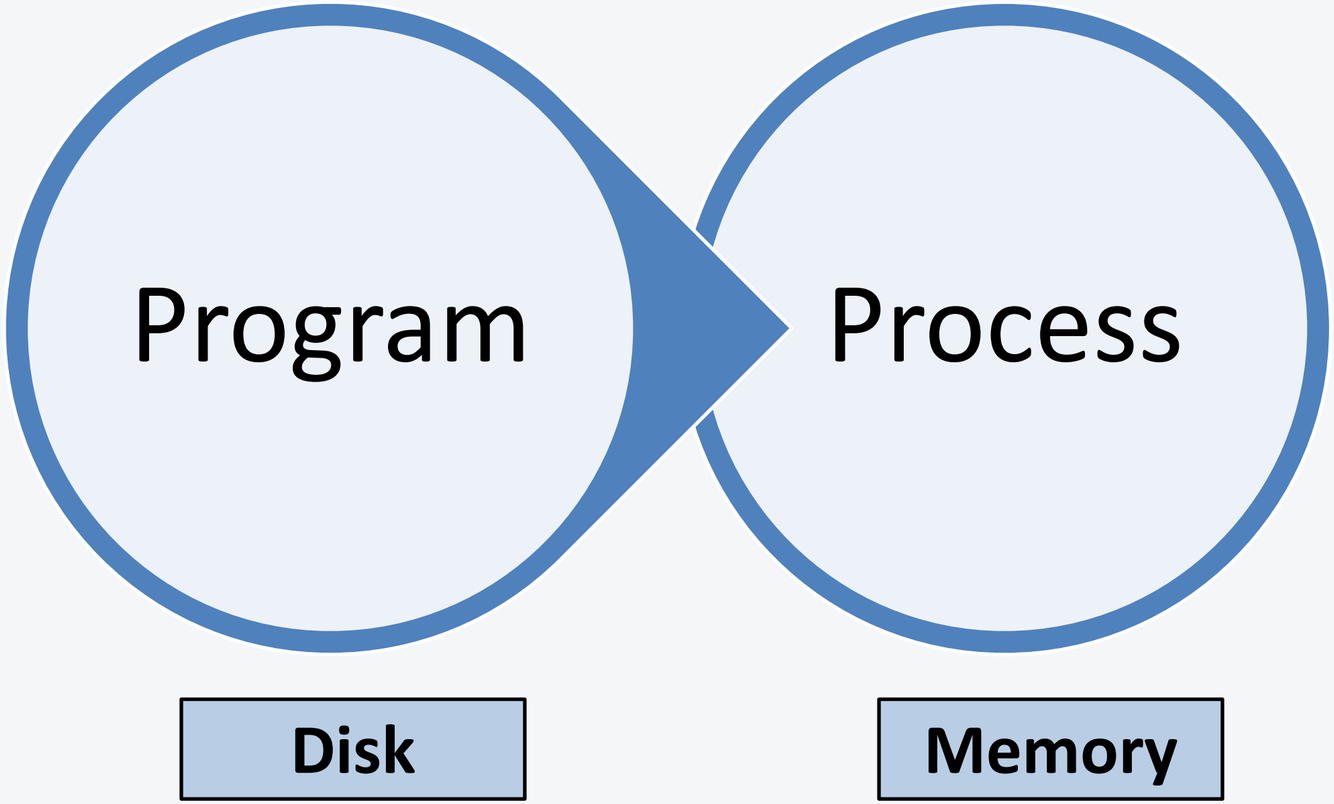
01. Permanent vs Ephemeral
02. Memory and Pointers
03. Important Files in CPP Project
04. Build Process
05. Loading the EXE

01.

Permanent vs Ephemeral

What's the difference?

What's the difference?



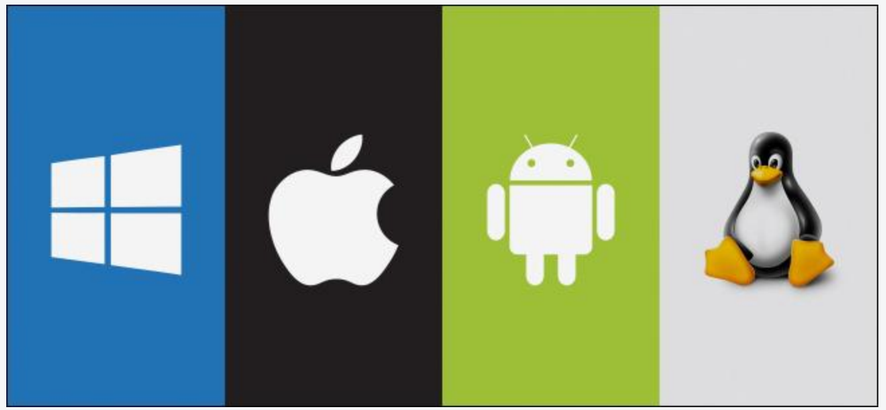
02.

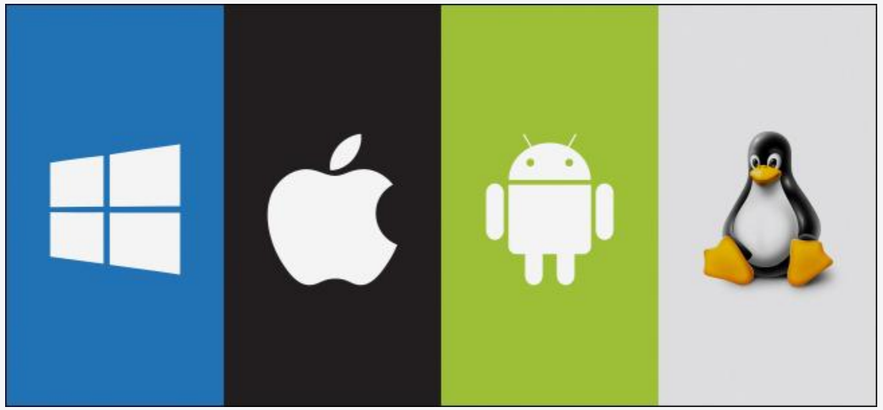
Memory and Pointers

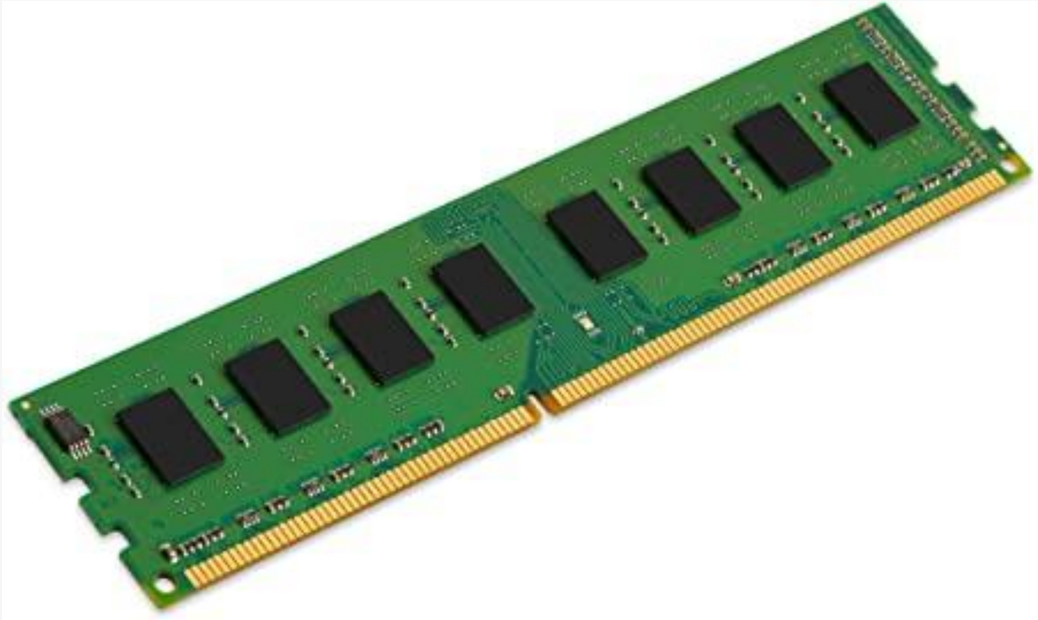
What's this?

Have you seen this?


LIBRARIAN



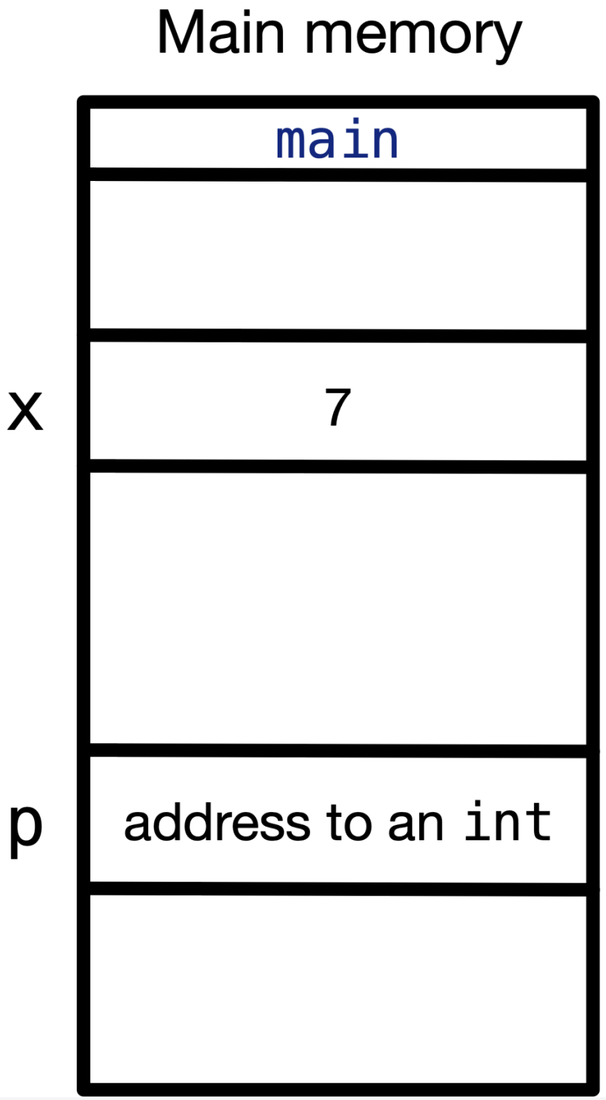




Memory Block and Address

Computer		Programmers		
Address	Content	Name	Type	Value
90000000	00			
90000001	00	sum	int (4 bytes)	000000FF(255_{10})
90000002	00			
90000003	FF			
90000004	FF	age	short (2 bytes)	FFFF(-1_{10})
90000005	FF			
90000006	1F			
90000007	FF			
90000008	FF			
90000009	FF	averge	double (8 bytes)	1FFFFFFFFFFFFFF ($4.45015E-308_{10}$)
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			
9000000E	90			
9000000F	00	ptrSum	int* (4 bytes)	90000000
90000010	00			
90000011	00			

Note: All numbers in hexadecimal




```c
#include <stdio.h>

int main(void) {
    int x = 7;
    int *p;

    return 0;
}
```

① data type of x is int

② data type of p is int*

03.

Important Files in CPP Project

CPP File

- A C++ file is a source code file that contains the program's instructions.
- It includes data definitions, program logic, functions, classes, and other constructs.
- The file is written in a human-readable form.
- C++ source files can be used in open-source projects, personal projects, or any project where you have the right to modify the code.

Header File

- A header file contains declarations of functions, classes, and constants.
- It does not usually contain full program logic; that is implemented in source (.cpp) files.
- Header files are written in human-readable form.
- They are included in source files using the #include directive to share declarations across multiple files.

Binary File

- A binary file is the compiled version of source code.
- It contains machine-readable code that the computer can execute directly.
- Binary files are not human-readable.
- They are generated by a compiler from source (.cpp) and header (.h) files.

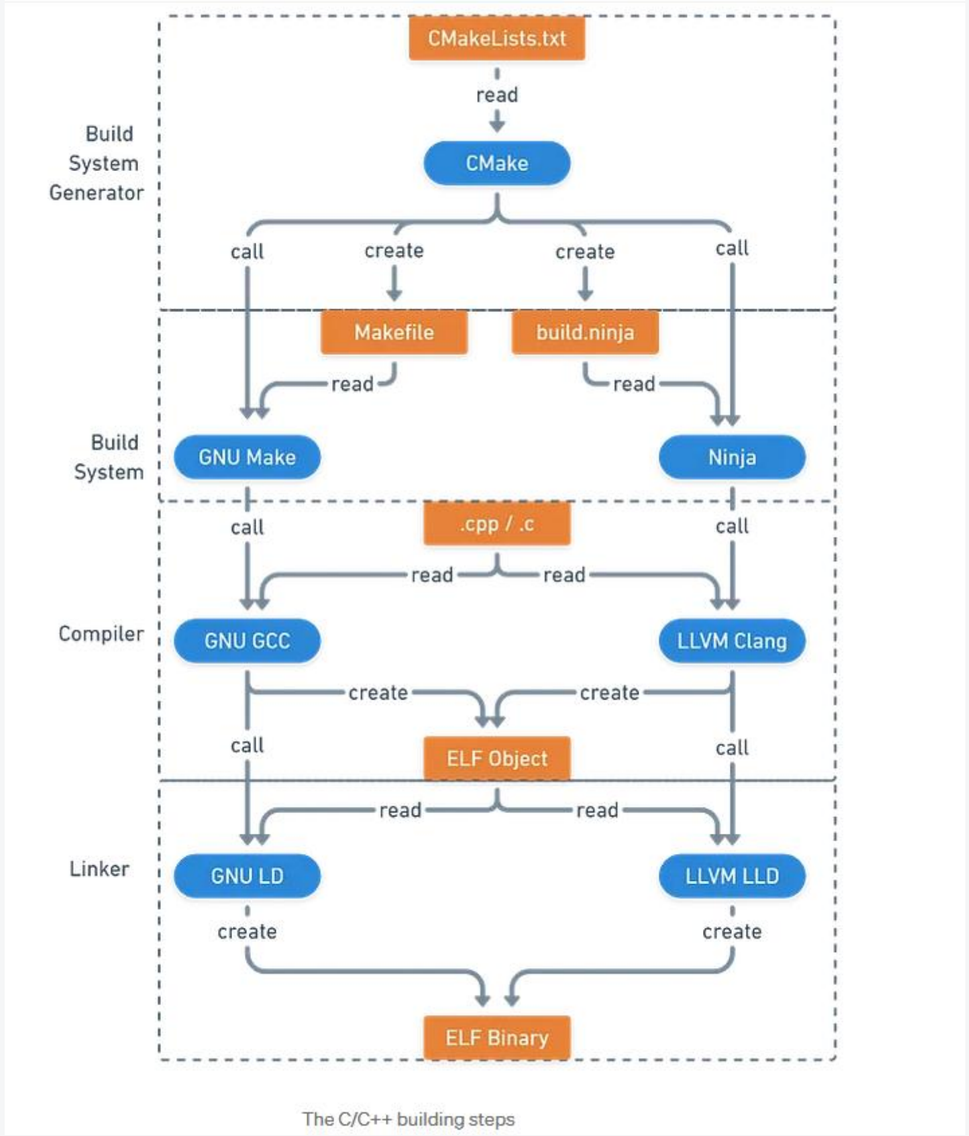
EXE File

- An EXE file is a type of binary file that contains a program ready to run.
- It is the result of compiling and linking source code and libraries.
- EXE files are machine-readable and cannot be directly read by humans.
- Running an EXE file executes the program without needing the original source code.

04.

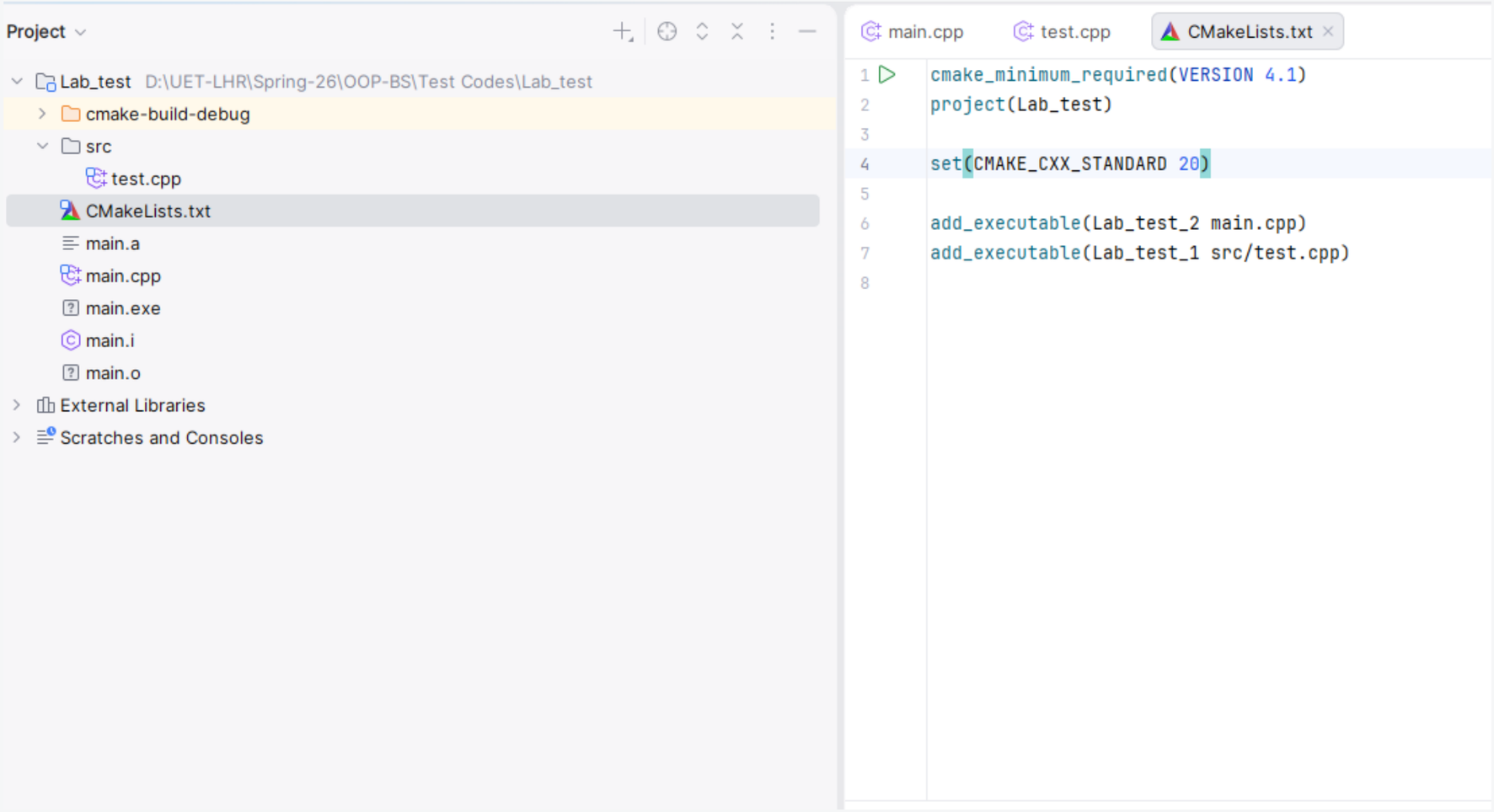
Build Process

Build Process



The C/C++ building steps

Step-1

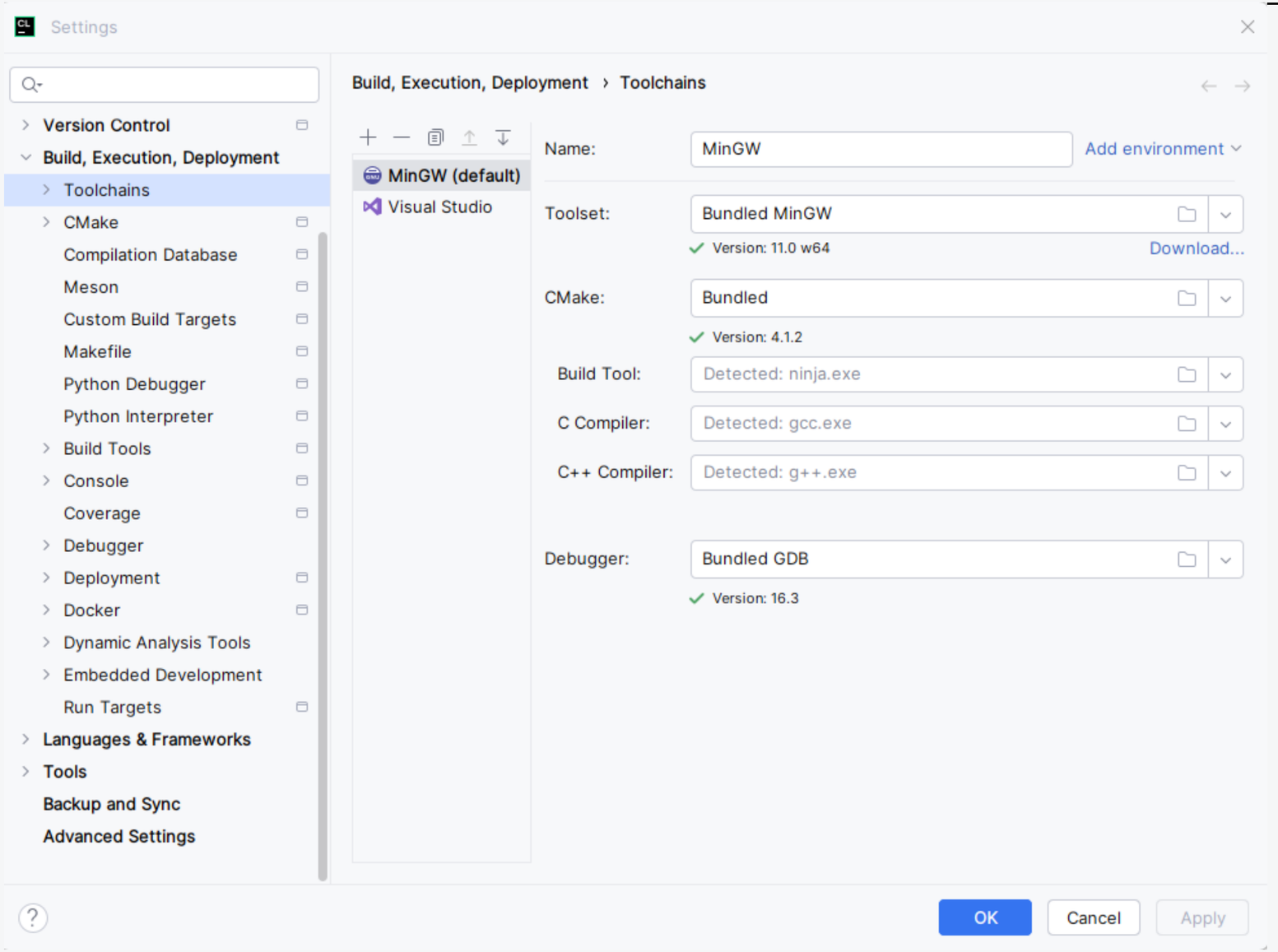

Project
Lab_test D:\UET-LHR\Spring-26\OOP-BS\Test Codes\Lab_test
cmake-build-debug
src
test.cpp
CMakeLists.txt
main.a
main.cpp
main.exe
main.i
main.o
External Libraries
Scratches and Consoles
main.cpp
test.cpp
CMakeLists.txt
1 cmake_minimum_required(VERSION 4.1)
2 project(Lab_test)
3
4 set(CMAKE_CXX_STANDARD 20)
5
6 add_executable(Lab_test_2 main.cpp)
7 add_executable(Lab_test_1 src/test.cpp)
8

Step-1

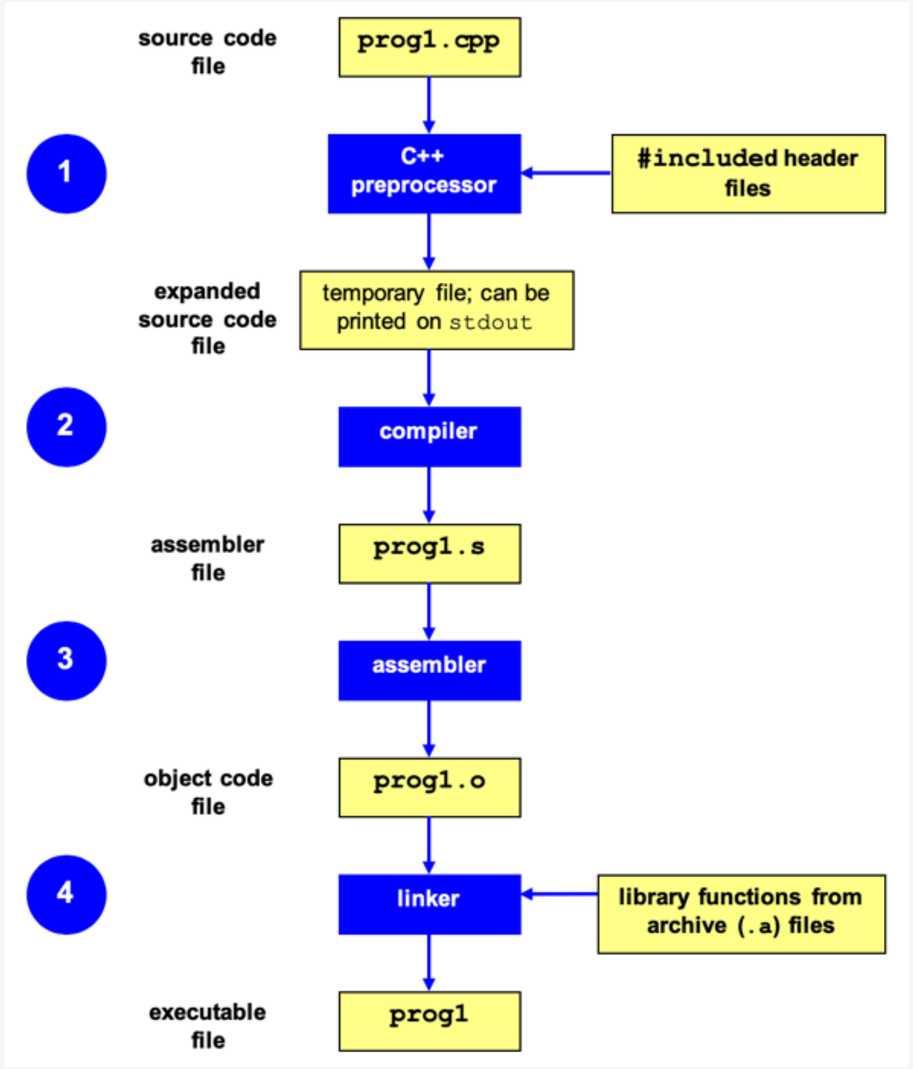
README.md | autograding.yml | student_code.cpp | test_student_code.cpp | CMakeLists.txt | tests\CMakeLists.txt

```
cmake_minimum_required(VERSION 3.14)
project(CPP_AUTOGRADED LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

include(FetchContent)

FetchContent_Declare(
        googletest
        URL https://github.com/google/googletest/archive/refs/tags/v1.14.0.zip
)

FetchContent_MakeAvailable(googletest)

enable_testing()

add_library(student_code
        src/student_code.cpp
)

target_include_directories(student_code PUBLIC
        ${PROJECT_SOURCE_DIR}/include
)

add_subdirectory(tests)

```

Step-1


Settings
Build, Execution, Deployment › Toolchains
Version Control
Build, Execution, Deployment
Toolchains
CMake
Compilation Database
Meson
Custom Build Targets
Makefile
Python Debugger
Python Interpreter
Build Tools
Console
Coverage
Debugger
Deployment
Docker
Dynamic Analysis Tools
Embedded Development
Run Targets
Languages & Frameworks
Tools
Backup and Sync
Advanced Settings
MinGW (default)
Visual Studio
Name:
MinGW
Add environment
Toolset:
Bundled MinGW
Version: 11.0 w64
Download...
CMake:
Bundled
Version: 4.1.2
Build Tool:
Detected: ninja.exe
C Compiler:
Detected: gcc.exe
C++ Compiler:
Detected: g++.exe
Debugger:
Bundled GDB
Version: 16.3
OK
Cancel
Apply

Step II, III and IV



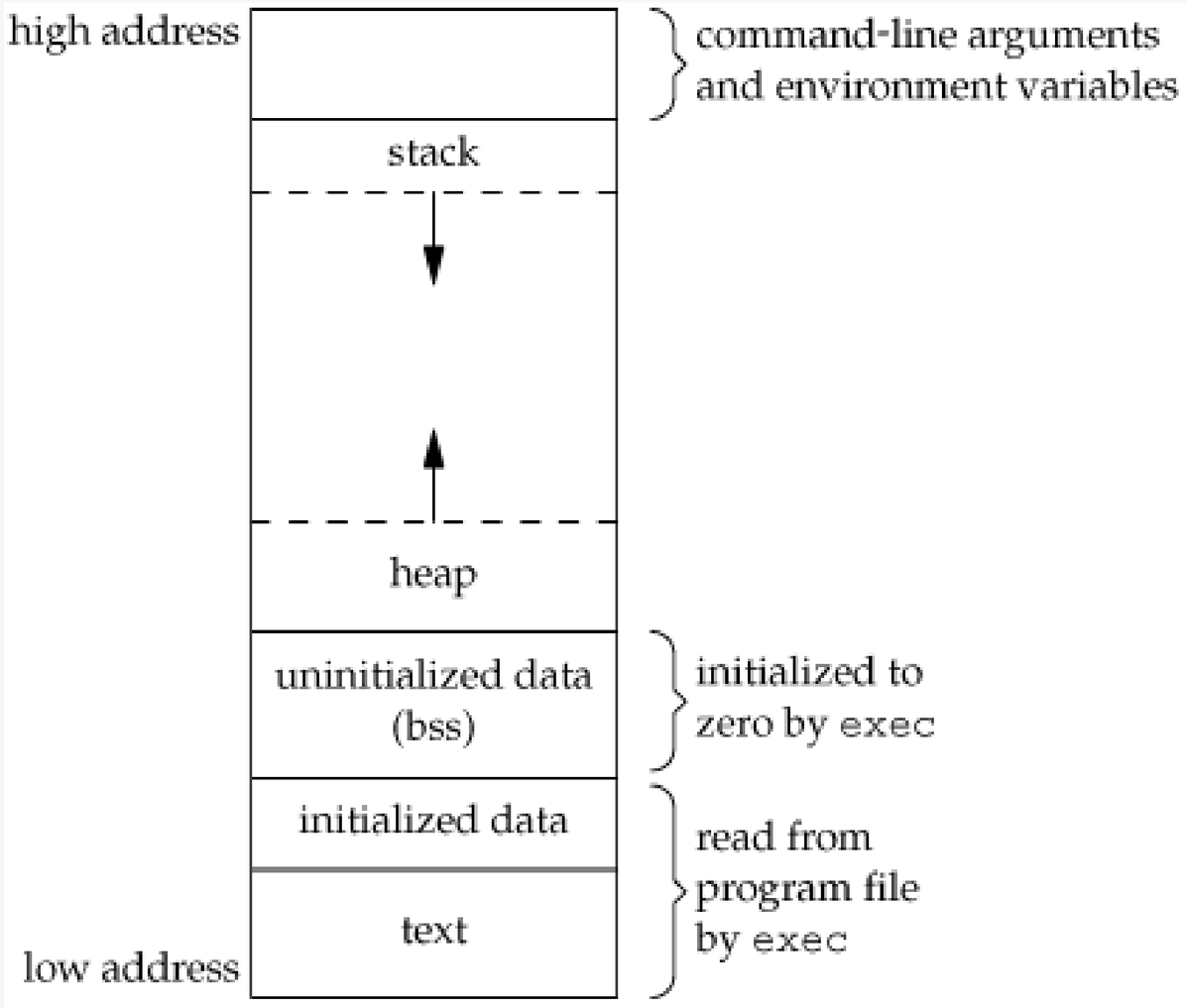
Step II, III and IV

/d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test

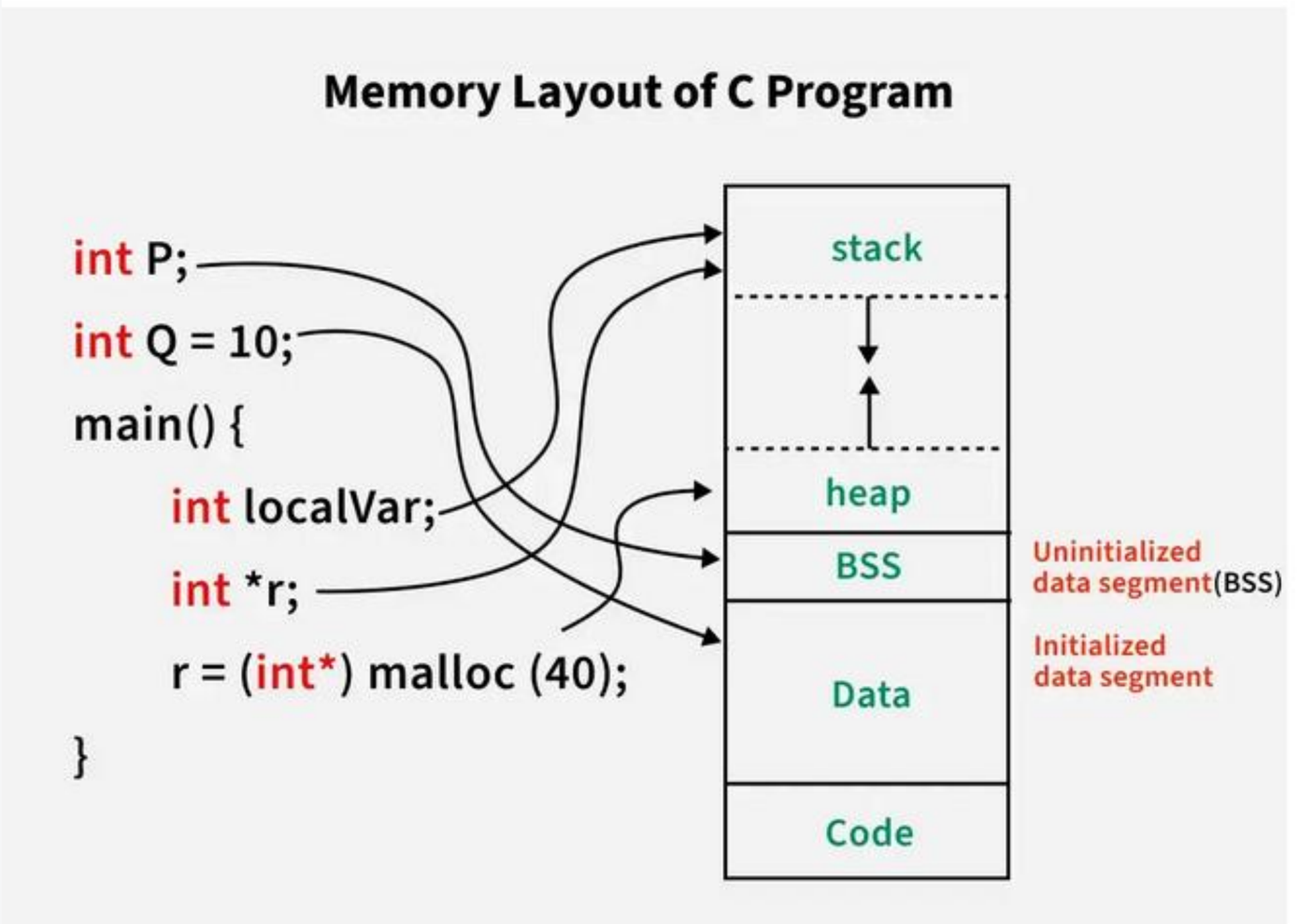
```
shaba@DESKTOP-KMUINPG MINGW64 ~
$ cd /d/UET-LHR/Spring-26/OOP-BS/Test\ Codes/Lab_test

shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$ g++ -E main.cpp -o main.i

shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$ g++ -S main.cpp -o main.s

shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$ g++ -c main.cpp -o main.o

shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$ g++ main.cpp -o main

shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$ ./main
32
shaba@DESKTOP-KMUINPG MINGW64 /d/UET-LHR/Spring-26/OOP-BS/Test Codes/Lab_test
$
```

05. Loading the EXE

PROCESS


high address
command-line arguments
and environment variables
stack
heap
uninitialized data
(bss)
initialized to
zero by exec
initialized data
text
read from
program file
by exec
low address

PROCESS


Memory Layout of C Program
int P;
int Q = 10;
main() {
int localVar;
int *r;
r = (int*) malloc (40);
}
stack
heap
BSS
Data
Code
Uninitialized data segment(BSS)
Initialized data segment

Thank You!