Session 7 / Part 2

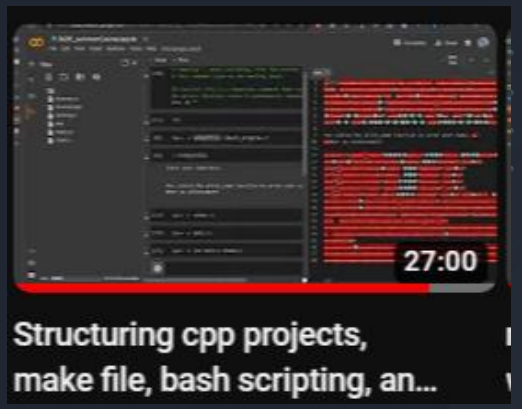
Makefile tool

As usual we discuss "Concept", "Necessity", & "Syntax"

1. Concept of Makefile

+ Makefile is a tool for automating the build process for your applications
+ It provides many options and flexibility to you, like not generating everything from Scratch, only the necessary files.

Read more



Based on C++ language

.CC file(s)

Using C++language compiler (called g++) And manually using **many *g++*** commands

final executable(s)

Based on "make" language

Makefile

.CC file(s)

Using make language compiler and using **very few *make*** commands

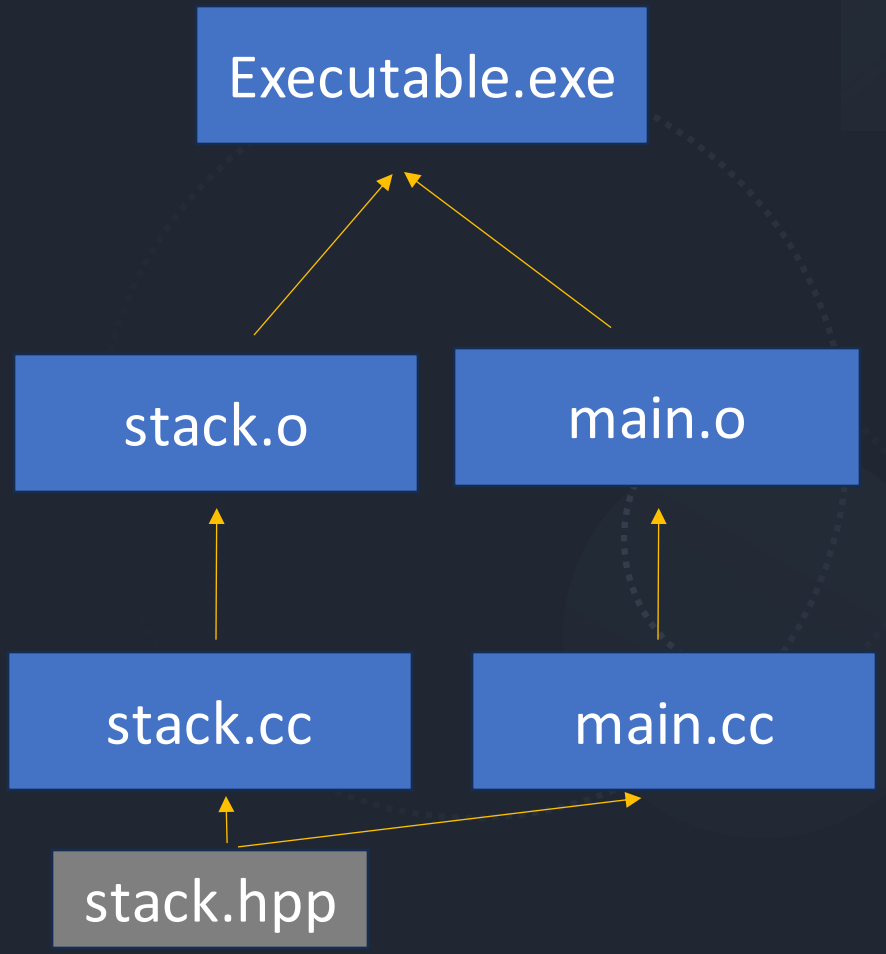
final executable(s)

2 Necessity of Makefile

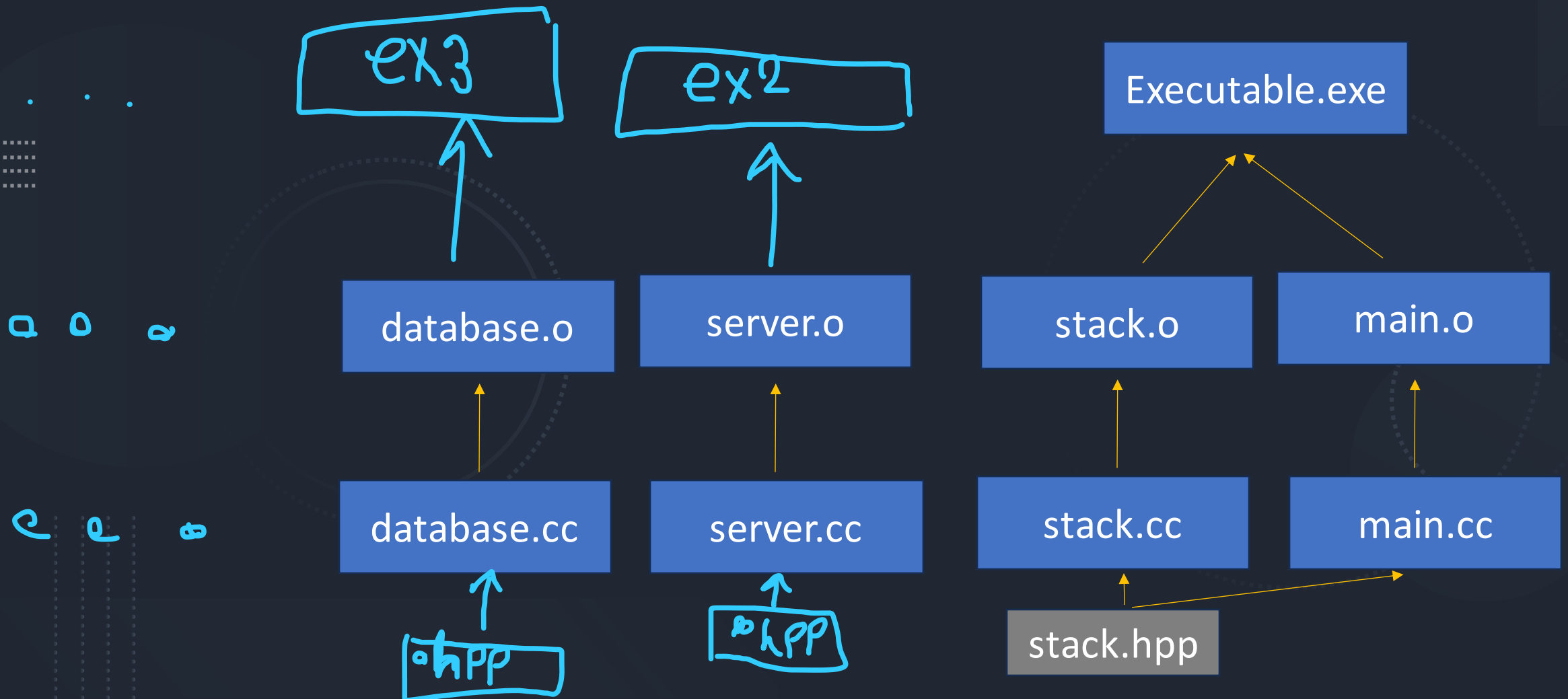
Linking process …
Generating the final executable file

Compilation process …
Generating binary file for each part of the whole project
(need to use – c flag after g++)

For header files you only need to **include** them in any file you need to use them. Nothing else!



2 Necessity of Makefile



1) Syntax

Makefile essential Components:

+ Variables

+ Target

+ Dependency

+ Recipe (command)

Notes:

- A set of Target, Dependency, and command is called "Rule"
- The convention is having a Rule called "all", you can change it, which generate all **target(s)**
- We usually have one rule called "clean" which cleans up all the intermediary files
- Unlike C++ that you could choose any name for your file(s), here we only have **one single** makefile which **should be named** Makefile or makefile
- You need to have one tab before each command!

1.Syntax

How does makefile work?
How does makefile know that it needs to regenerate a particular target, or the current version is up to date and there is no need to regenerate that?

The answer will be discussed on one example.

1.Syntax

Ok, What should you finally write in your Makefile?

Most of the time you need to write a Makefile, with the help of variables, which has the following elements:

- One rule named "all". This rule generates all the final executable files, possibly more than one, and does not have any recipe.
- One rule named "clean", which cleans all the targets.
- One rule that its target is one ".o" file for all the files that need compilation. In the recipe you should write the command for their compilation.
  (For example, in our stack program, we need to write two rules for .o files. One rule is associated with stack.o and one with main.o)