Department of Computer Science and Information Engineering

*Object Oriented Programming*

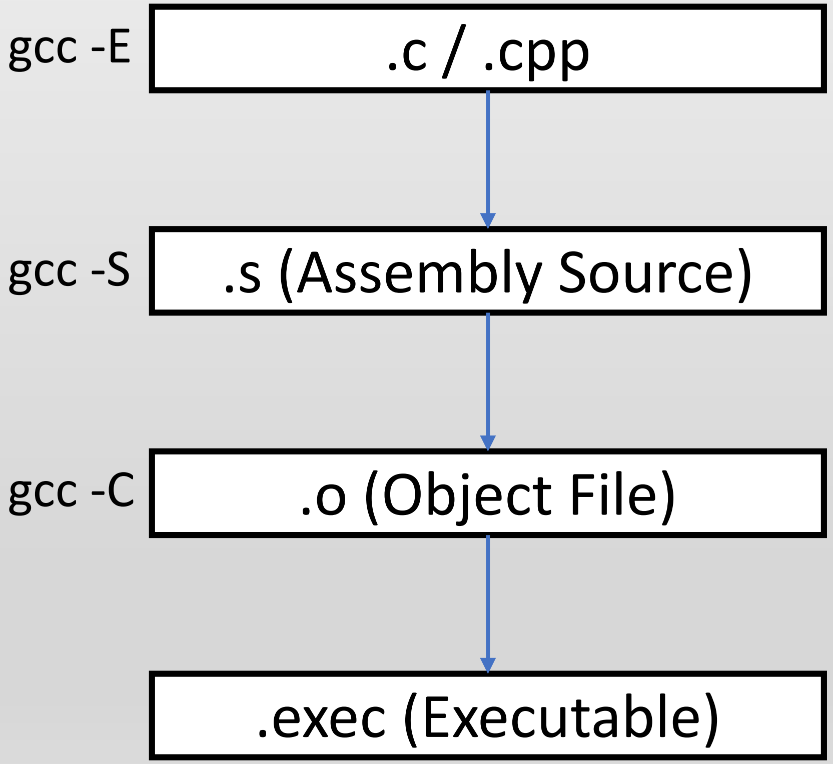
# Lecture 02: makefile

Shuo-Han Chen (陳碩漢),
*shchen@csie.ntut.edu.tw*

The Sixth Teaching Building 327
M 15:10 - 16:00 & F 10:10 - 12:00

# How your code is translated into a program?

- Before we dig into the details, let’s get familiar with how your code turn into a program

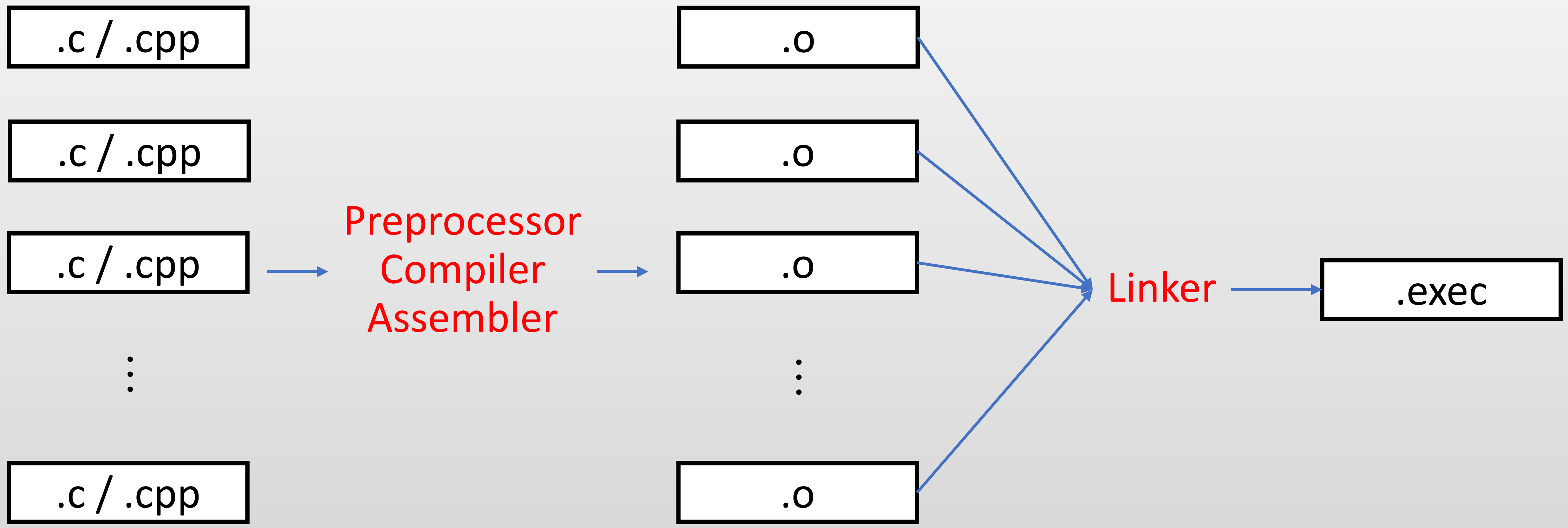


Preprocessor : Handle #include, #define, Remove comments
Compiler : Translate into Assembly

Assembler : Translate assembly into object file

Linker : Bring together object files to produce the executable

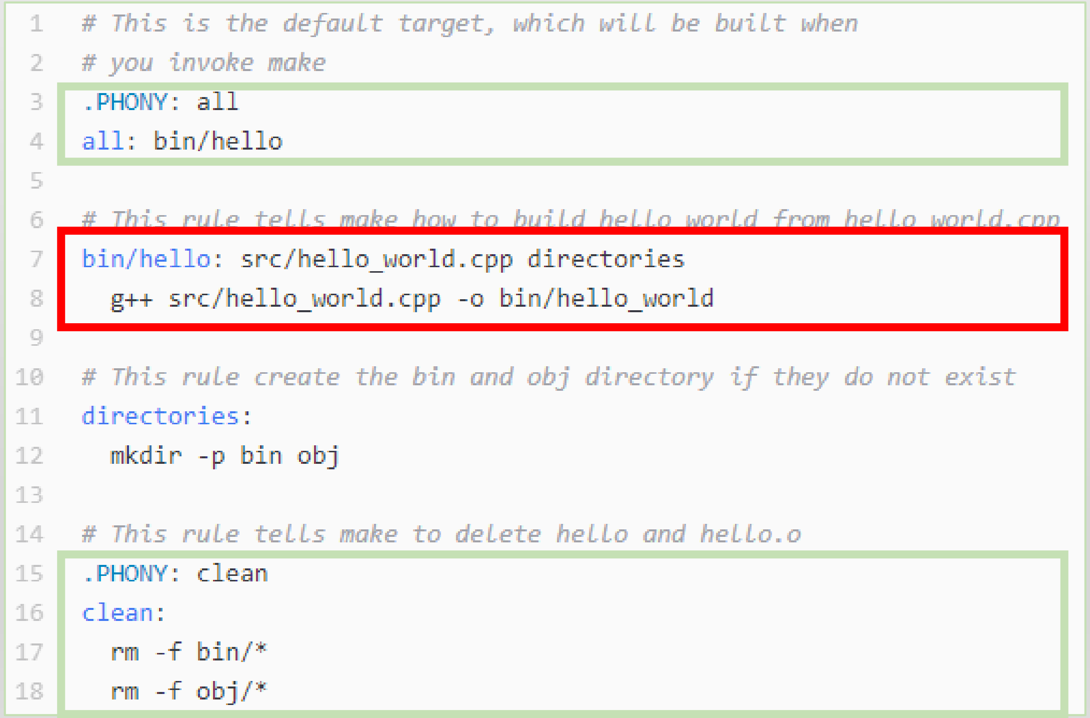
# Why use makefile during compiling?



- makefile can avoid all the gcc commands for each c/cpp file
- Specify the dependency (相依性) between each .c/.cpp
- Producing the final executable by entering “make”

# makefile Format

- makefile consists of a collection of rules
  - targets: prerequisites files or targets
    - command-script to complete for this target
  - Lower command-script can be referred to by upper script



```
# This is the default target, which will be built when
# you invoke make
.PHONY: all
all: bin/hello

# This rule tells make how to build hello world from hello_world.cpp
bin/hello: src/hello_world.cpp directories
  g++ src/hello_world.cpp -o bin/hello_world

# This rule create the bin and obj directory if they do not exist
directories:
  mkdir -p bin obj

# This rule tells make to delete hello and hello.o
.PHONY: clean
clean:
  rm -f bin/*
  rm -f obj/*
```

- This is the basic makefile, we will learn more in the future

Reference: https://www.oreilly.com/library/view/c-cookbook/0596007612/ch01s16.html

## Basic g++ Options

```
# Generate file.o
g++ -c file.cpp

# Generate Assembly : file.s
g++ -S file.cpp

# Generate file.exec
g++ file.cpp  -o file

# Generate file.exec from object and cpp files
g++ file.cpp  obj.o -o file
```

```
# Show all the warning
gcc -Wall -o main main.c

# Warning as error
gcc -Wall -Werror -o main main.c
```