Background on Binary Analysis

Introduction to Security (184.783, 192.082)
S&P Research Division

slides are adapted from “CyberChallenge.IT Software Security I” by Lorenzo Veronese

Compilation

From C Code to Executables

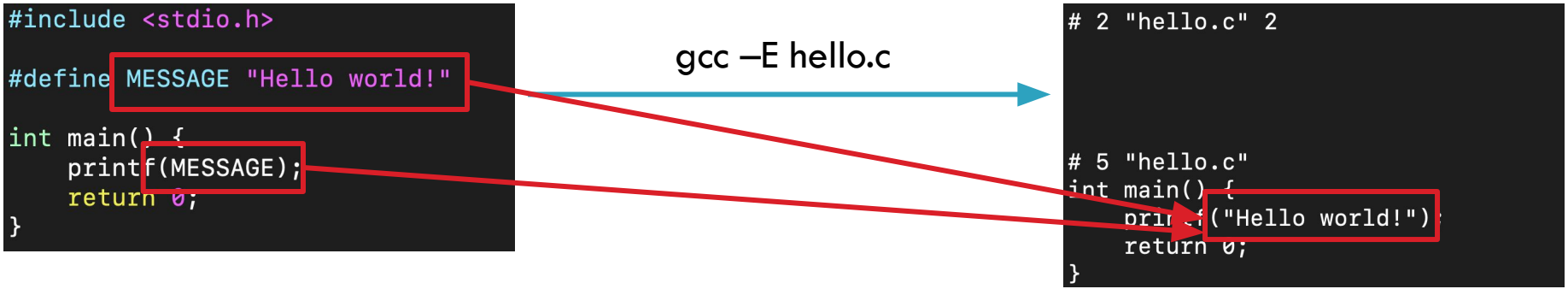
Compiling a C program is a multi-stage process composed by 4 steps

- **preprocessing**
- **compilation**
- **assembly**
- **linking**

see first 10 minutes of https://www.youtube.com/watch?v=8PrOp9t0PyQ

Preprocessing

In the first phase, **preprocessor** commands (in C they start with '#') are interpreted

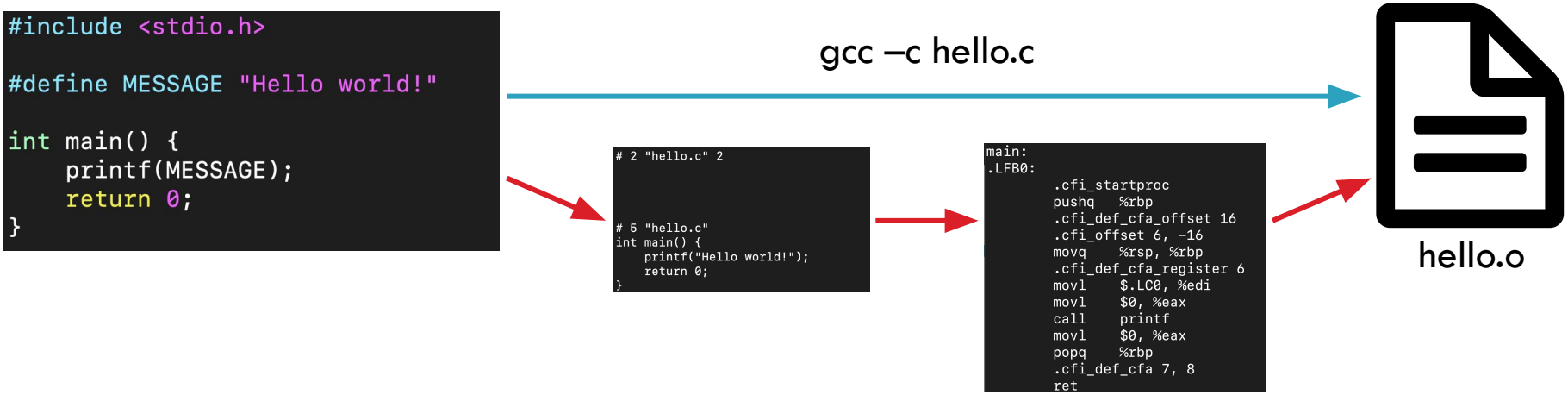


Compilation

In the second phase, preprocessed code is translated into **assembly instructions**

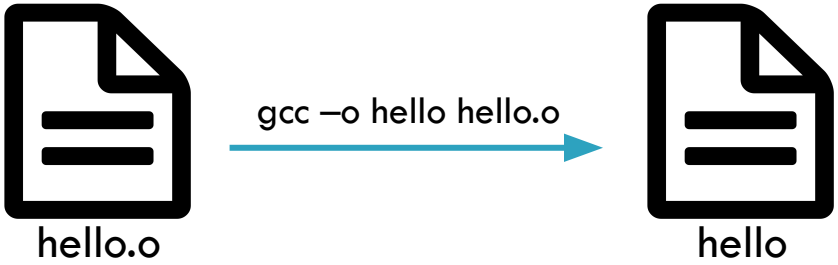
Assembly

In the assembly phase assembly instructions are translated into **machine or object code**



Linking

- In the last phase (multiple) object files are combined in a single executable
- In the generated file references (links) to the used library are added.



Two approaches can be used in the linking phase

Static Link

- Binaries are self-contained and do not depend on any external libraries

Dynamic Link

- Binaries rely on system libraries that are loaded when needed
- Mechanisms are needed to dynamically relocate code

Executable and Linkable Format

The Executable and Linkable Format (ELF) is a common file format for object files

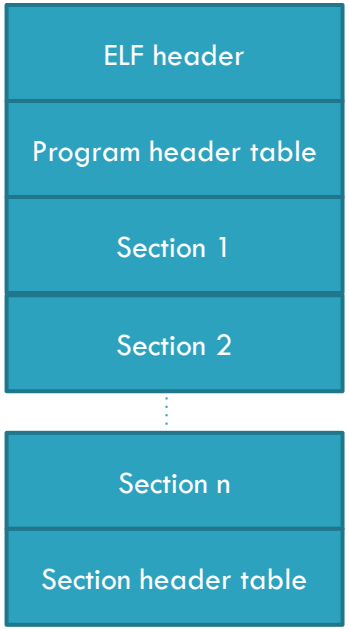
There are three types of object files

- **Relocatable file** containing code and data that can be linked with other object files to create an executable or a shared object file
- **Executable files** holding a program suitable for execution
- **Shared object files** that can be
  - linked with other relocatable and shared object files to obtain another object file
  - used by a **dynamic linker** together with other executable files and object files to create a **process image**

Executable and Linkable Format

Any ELF file is composed by

- **ELF header** describing the file content
- **Program header table** providing informations on how to create a process image
- sequence of **Sections** containing what is needed for linking (instructions, data, symbol table, relocation information, …)
- **Section header table** with a description of previous sections



ELF: Relevant Sections

.text	contains the executable instructions of a program
.bss	contains uninitialised data that contribute to the program's memory image
.data **.data1**	contain initialized data that contribute to the program's memory image
.rodata **.rodata1**	are similar to .data and .data1, but refer to read only data
.symtab	contains the program's symbol table
.dynamic	provides linking information