CS 6015: Software Engineering

Spring 2024

Lecture 12: Include files - Libraries

One way to organize source files

- Split header files from cpp files

- Can we compile?

- What happens to .cpp files?
  - One way to update all cpp files with the new header location
  - Could be time consuming for large project

- Solution?

One way to organize source files

- Split header files from cpp files

- Can we compile?

- What happens to .cpp files?
  - One way to update all cpp files with the new header location
  - Could be time consuming for large project

- Solution: include the new path while compiling (using `-I` flag)
 - Update only the `Makefile`

Libraries

- Code that can be reused by programs

- Two types:
  - Static library (or **archive**)
  - Dynamic library (or **shared library**)

- Libraries vs. namespaces

Static library vs Dynamic library

- Static library: code linked **at compile time**. The executable file generated keeps its own copy of the library code.

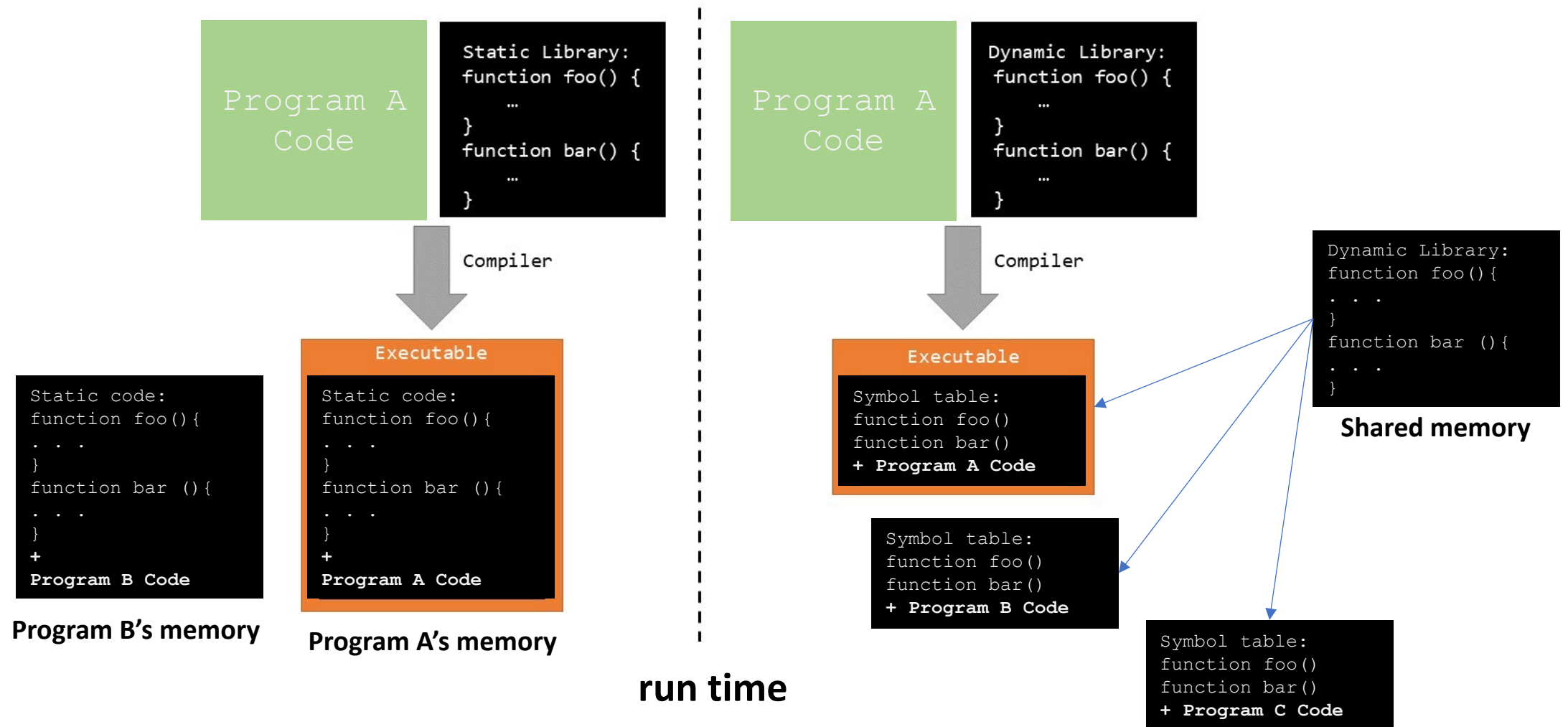- Dynamic library: code shared by multiple programs and loaded to memory **at runtime.**

Static library vs Dynamic library: Illustration

Compile time
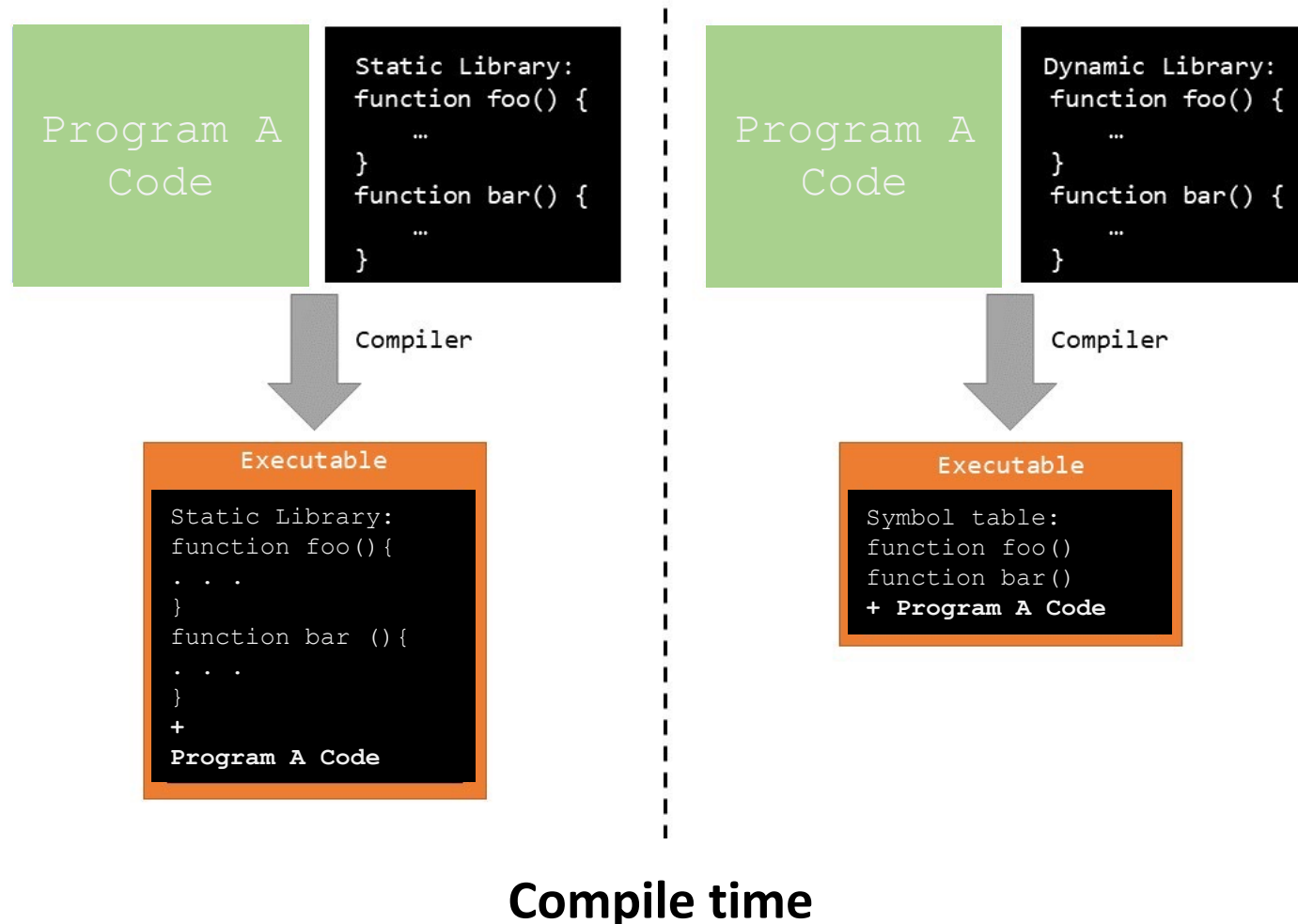
Static library vs Dynamic library: Illustration
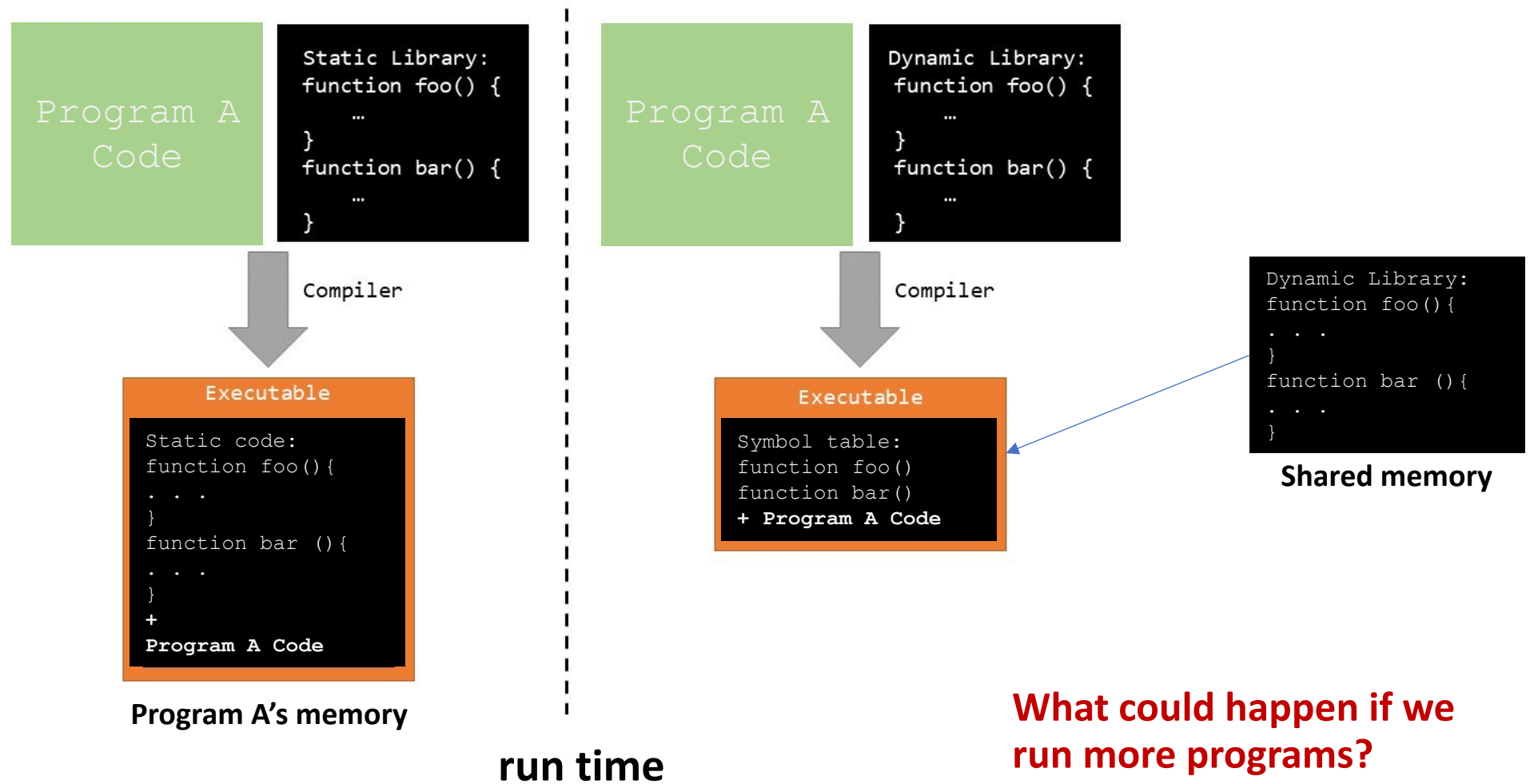
Static library vs Dynamic library: Illustration

Drawbacks

- Static library
  - Increases the size of the program
  - Modifying the library requires recompiling and reloading into the program.

- Dynamic library
  - Extra concern on installing

Extensions

- Static library
  - .a on Unix/Linux/Mac machines
  - .lib on windows machines

- Dynamic library
  - .so on Unix/Linux/Mac machines
  - .dll on windows machines

Static library: Adding and Linking

- Create the library

  - Generate the object files

  - Use ar (a Linux **ar**chive utility tool) to create the library file

- Linking the library

  - Specify library path using: `-L flag`

  - `-lname` is equivalent to `libname.a`

- With Cmake:

  - **add_library**(LibraryName **STATIC** simple_lib.cpp)

Dynamic library: Adding and Linking

- Linux

  - Create the library

    - Generate the object files

    - Use -shared flag with clang++: *clang++ -shared -o libmy_library.so my_library.o*

  - Linking the library

    - Specify library path using: `-L flag` with `-lname` as in static library

    - Use `-rpath` flag to specify the shared library path when building the executable.

- With Cmake:

  - **add_library**(LibraryName **SHARED** simple_lib.cpp)