CS 35L

Software Construction Laboratory

Lecture 7.2

21st February, 2019

Logistics

- Hardware requirement for Week 8
  - Seeed Studio BeagleBone Green Wireless Development Board
- Presentations for Assignment 10
  - https://docs.google.com/spreadsheets/d/1o6r6CKCaB2du3klPfIHiquymhBvbn7oP0wkHHMz_q1E/edit?usp=sharing
- Assignment 6 is due on 24th Feb, 2018 at 11:55pm

Review - Previous Lab

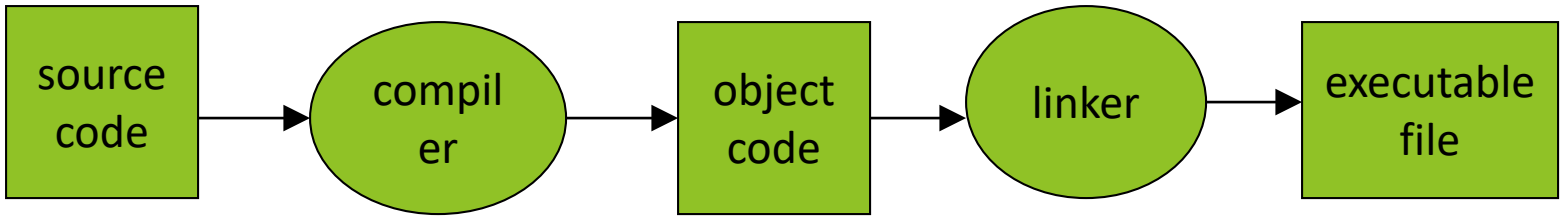
- Handling Race Condition
- Mutex and Semaphores
- Pthread Library
  - Pthread create and join functions

Dynamic Linking

Lifecycle of a C program

- The following entities help in getting a C program to work
  - Compiler
  - Assembler
  - Linker
  - Loader

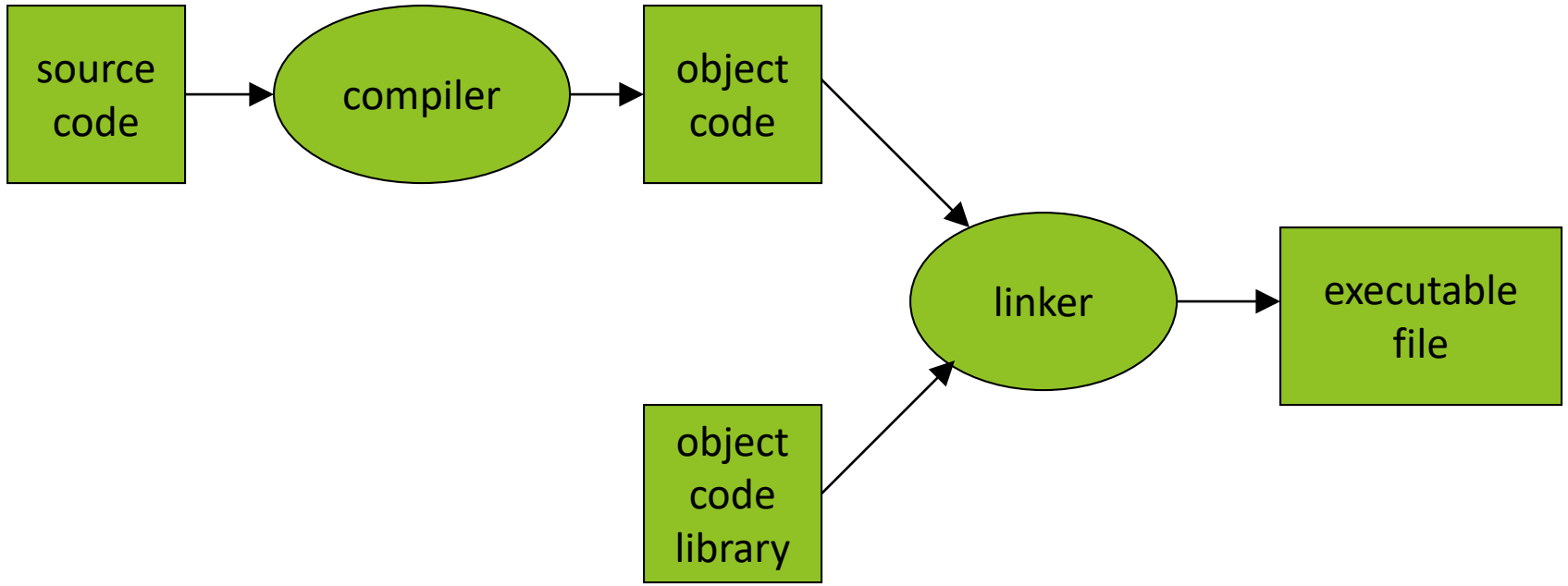
Building an Executable File



Translates programming language statements into cpu's machine-language instructions

Takes one or more object files generated by a compiler and combines them into a single executable file

Linking Libraries



A previously compiled collection of standard program functions

Static Linking

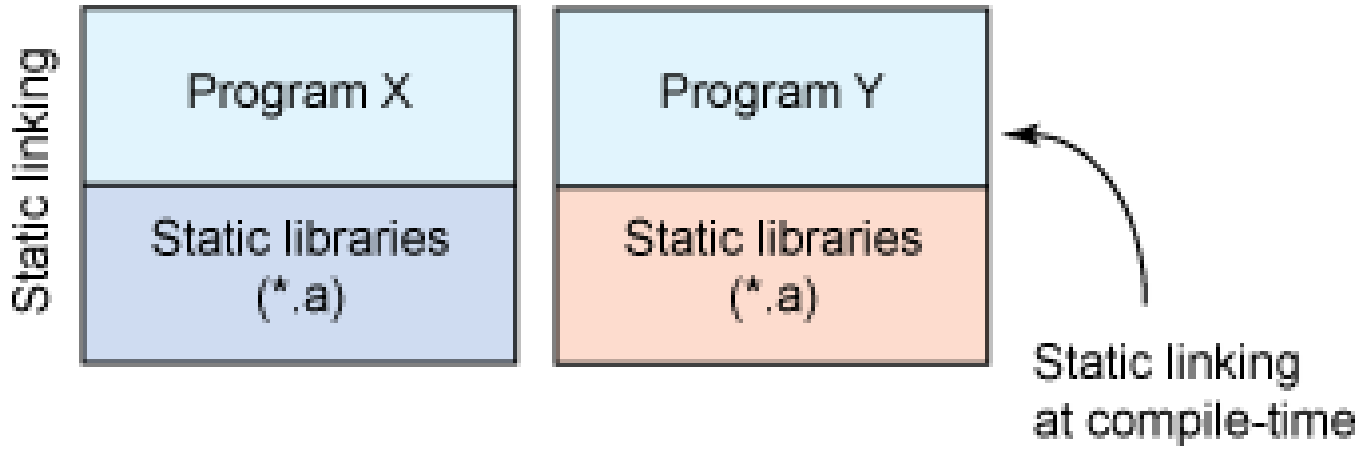
- Carried out only once to produce an executable file
- If static libraries are called, the linker will copy all the modules referenced by the program to the executable
- Static libraries are typically denoted by the .a file extension

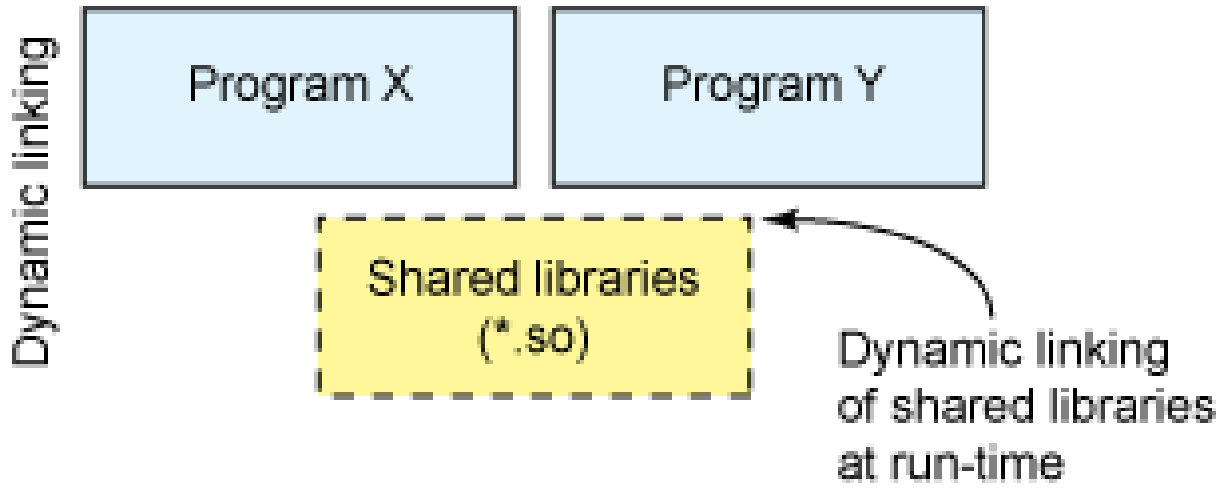
Linking and Loading

- Linker collects procedures and links together the object modules into one executable program
- Why isn't everything written as just one big program, saving the necessity of linking?
- Efficiency: if just one function is changed in a 100K line program, why recompile the whole program? Just recompile the one function and relink.

Dynamic Linking

- Allows a process to add, remove, replace or
- relocate object modules during its execution.
- If shared libraries are called:
  - Only copy a little reference information when the executable file is created
  - Complete the linking during loading time or running time
- Dynamic libraries are typically denoted by the .so file extension
  - .dll on Windows


Static linking
Program X
Static libraries
(*.a)
Program Y
Static libraries
(*.a)
Static linking
at compile-time


Dynamic linking
Program X
Program Y
Shared libraries
(*.so)
Dynamic linking
of shared libraries
at run-time

Dynamic Linking

- Unix systems: Code is typically compiled as a dynamic shared object (DSO)
- Dynamic vs. static linking resulting size

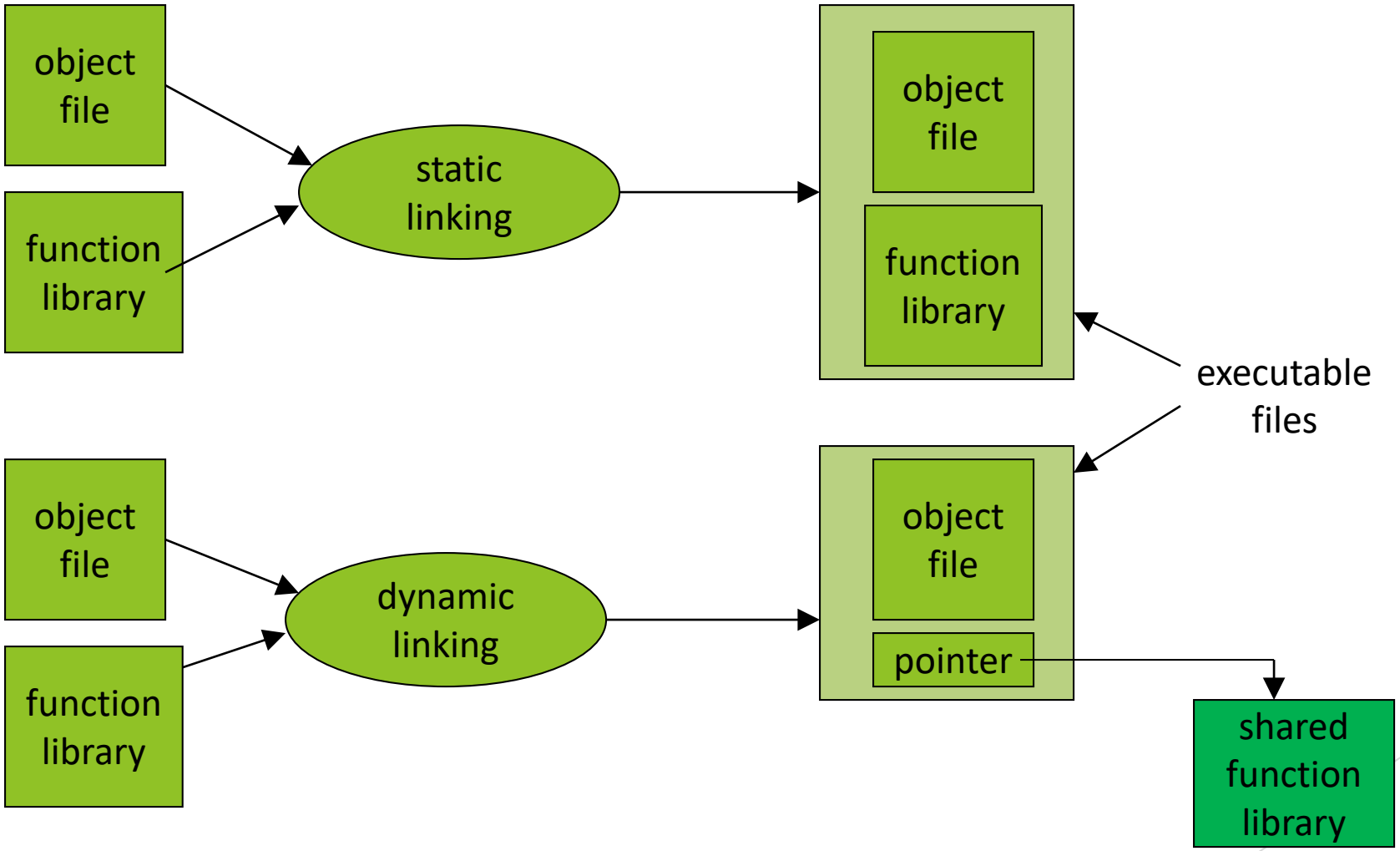
```
$ gcc -static hello.c -o hello-static
$ gcc hello.c -o hello-dynamic
$ ls -l hello
      80 hello.c
   13724 hello-dynamic
 1688756 hello-static
```

- If you are the sysadmin, which do you prefer?

Advantages of Dynamic Linking

- The executable is typically smaller
- When the library is changed, the code that references it does not usually need to be recompiled
- The executable accesses the .so at run time; therefore, multiple programs can access the same .so at the same time
- Memory footprint amortized across all programs using the same .so

Smaller is more efficient



Disadvantages of dynamic linking

- Performance hit
- Need to load shared objects (at least once)
- Need to resolve addresses (once or every time)
- Remember back to the system call assignment…
- What if the necessary dynamic library is missing?
- What if we have the library, but it is the wrong version?

Assignment 7 - Laboratory

- Write and build simple cos(sqrt(3.0)) program in C
  - Use ldd to investigate which dynamic libraries your cos program loads
  - Use strace to investigate which system calls your cos program makes
- Use “ls /usr/bin | awk ‘NR%101==nnnnnnnnn%101’” to find ~25 linux commands to use ldd on
  - Record output for each one in your log and investigate any errors you might see
  - From all dynamic libraries you find, create a sorted list
    - Remember to omit the duplicates!

Assignment 6 – Homework

- Download the single-threaded ray tracer implementation
- Run it to get output image
- Multithread ray tracing
  - Modify main.c and Makefile
- Run the multithreaded version and compare resulting image with single-threaded one

Assignment 6 – Homework

- Build a multi-threaded version of Ray tracer
- Modify “main.c” & “Makefile”
  - Include <pthread.h> in “main.c”
  - Use “pthread_create” & “pthread_join” in “main.c”
  - Link with –lpthread flag (LDLIBS target)
- make clean check
  - Outputs “1-test.ppm”
  - Can’t see “1-test.ppm”
  - sudo apt-get install gimp (Ubuntu)
  - X forwarding (lnxsrv)
    - ssh –X username@lnxsrv.seas.ucla.edu
  - gimp 1-test.ppm

Assignment 6 – Homework

- Ensure no compile error exists!
- Read the source code to understand the task
- **Don’t modify other functions in the original code**
- How to divide the task to run multiple threads?
- Difficulty: the 3rd and 4th arguments of pthread_create function
  - Argument 3: a function that divides the input by threads
  - Argument 4: an array to hold data for each thread

1-test.ppm

Figure. 1-test.ppm & baseline.ppm

Presentations

- Today's Presentation:
  - Yufei Wang
  - Calvin Chen
- Next up:
  - Renee Hsu
  - Atharv

Questions?