CS 35L

LAB 8,

TA: Sucharitha Prabhakar

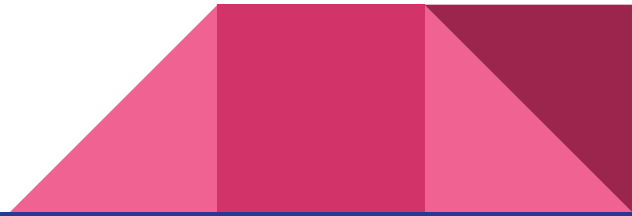
EMAIL ID: prabhakarsucharitha@gmail.com

Outline

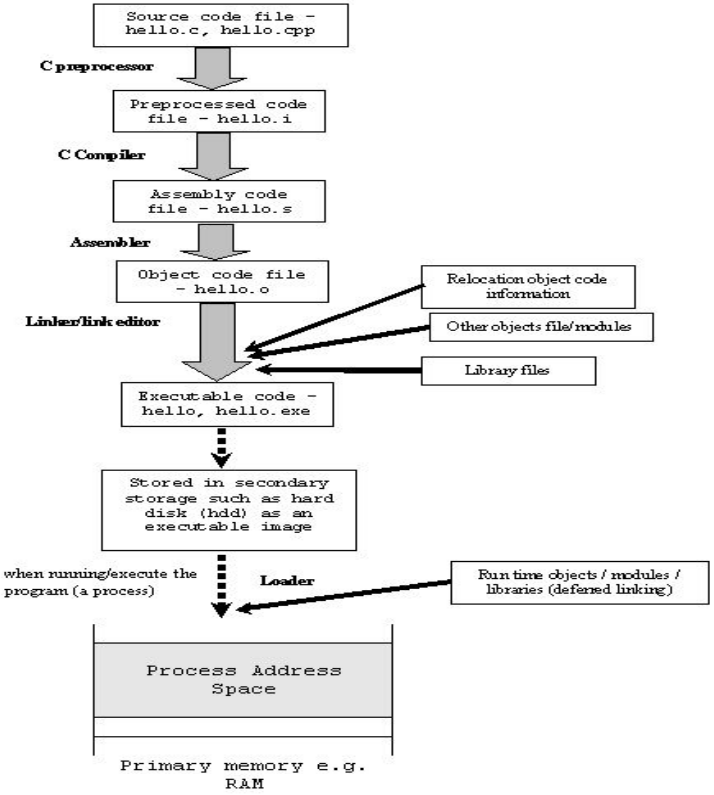
Linking and loading

Static linking

Dynamic linking



Building an executable file

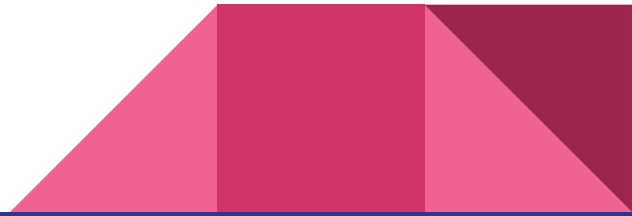


Linking and loading

Linker collects procedures and links object modules together 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.
- Multiple-language programs
- Other reasons?

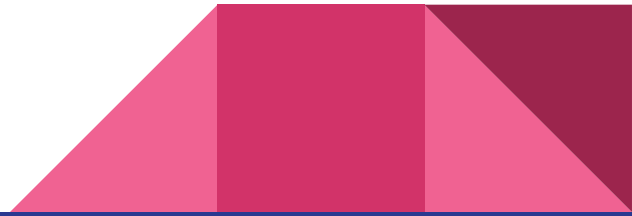


Static linking

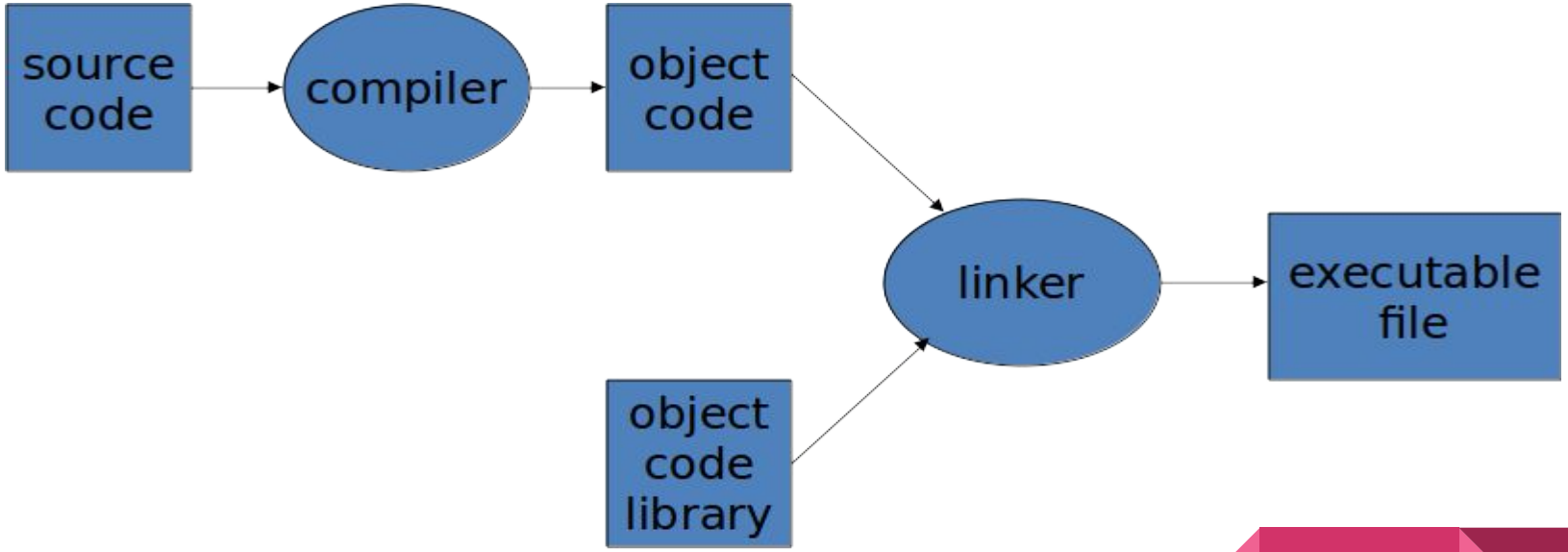
Carried out only once to produce an executable file

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



Static linking



Dynamic linking

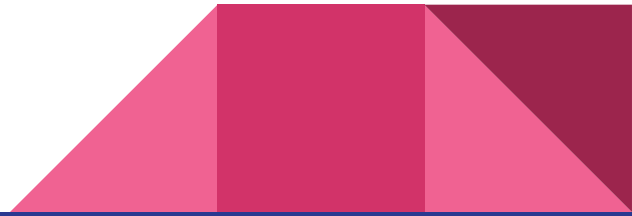
Allows a process to add, remove, replace or relocate object modules during its execution.

If shared libraries are called:

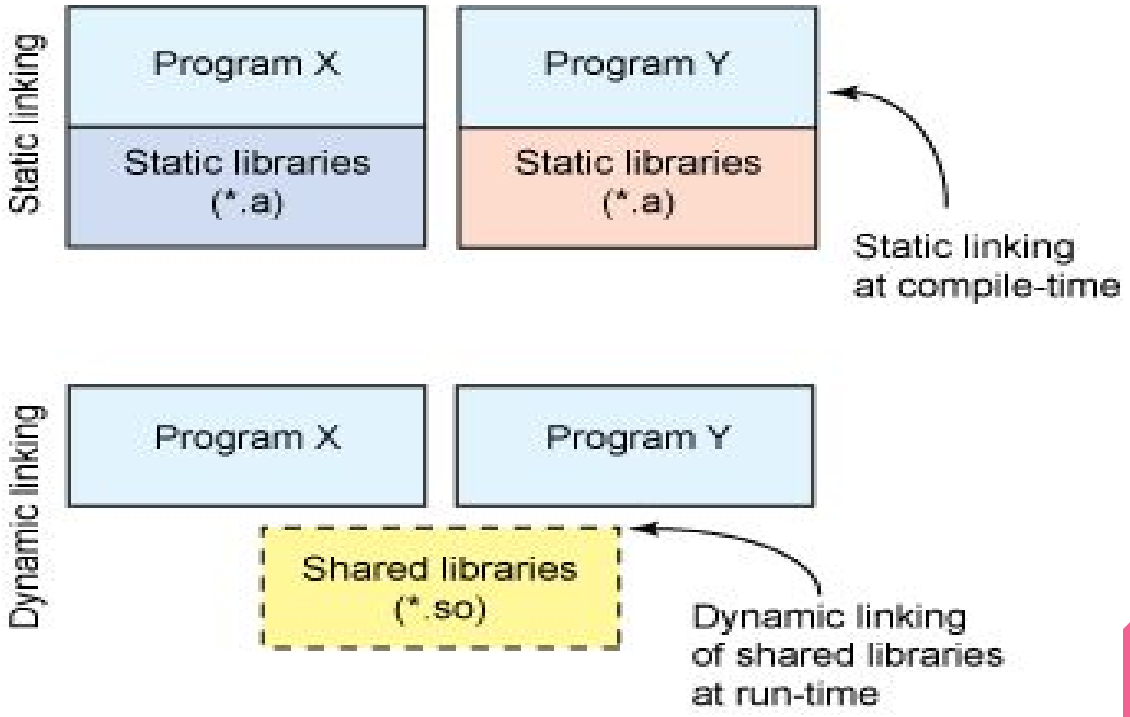
- Only copy a little reference information when the executable file is created
- Complete the linking during loading time or running time

Dynamic libraries are typically denoted by the .so file extension

- .dll on Windows



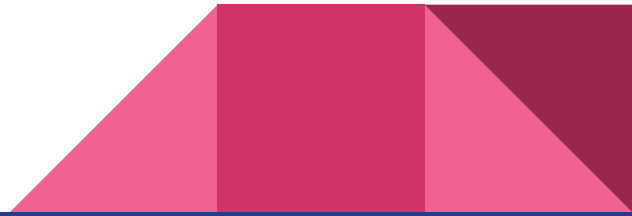
Dynamic linking



Dynamic linking

Dynamic vs. static linking resulting size

If you are the sysadmin, which do you prefer?



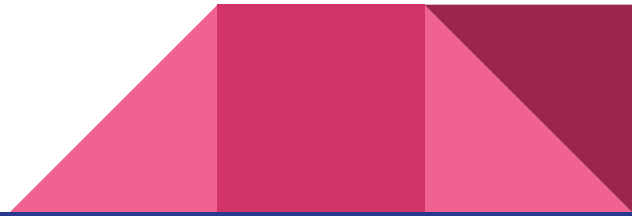
Advantages of dynamic linking

The executable is typically smaller

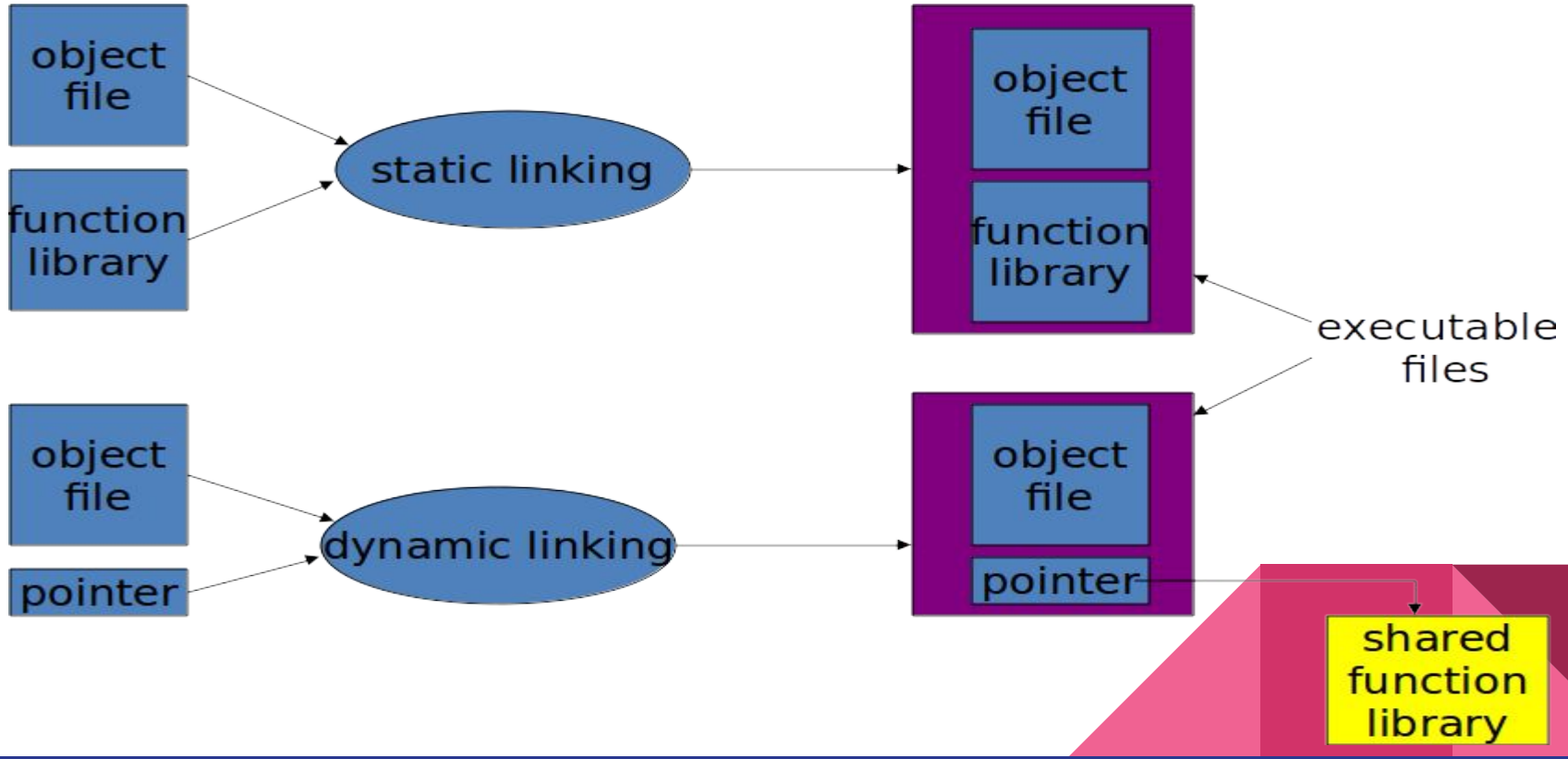
When the library is changed, the code that references it does not usually need to be recompiled.

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)

What if the necessary dynamic library is missing?

What if we have the library, but it is the wrong version?

Lab 9

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

Lab 9

Use “ls /usr/bin | awk ‘NR%101==SID%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 remove the duplicates!