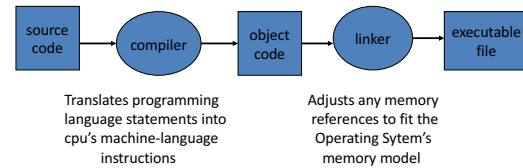


CS 35L Software Construction Lab

Week 8 – Dynamic Linking

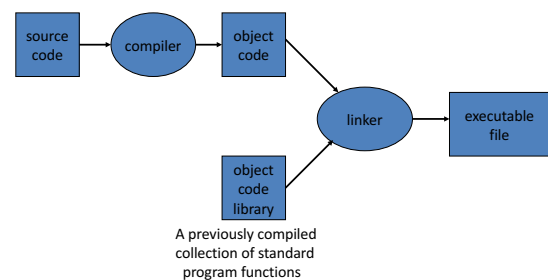
Building an executable file



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

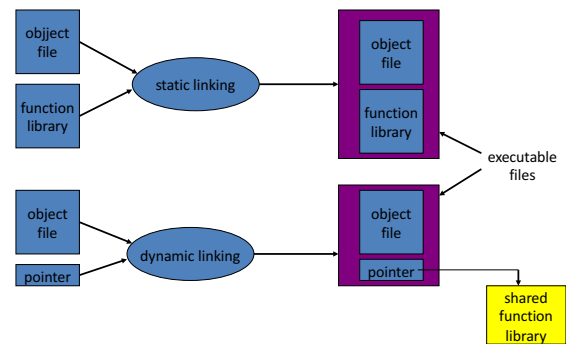
Linking libraries



Dynamic Linking

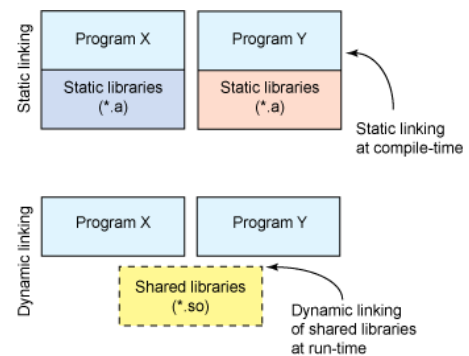
- If shared libraries are called:
 - Only copy a little reference information when the executable file is created
 - Complete the linking during loading time or running time
- Dynamic libraries are typically denoted by the .so file extension
 - .dll on Windows

Smaller is more efficient



Linking and Loading

- Linker collects procedures and links them together 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.
 - Multiple-language programs



Dynamic linking

- Dynamic vs. static linking resulting size

```
$ gcc -static hello.c -o hello-static
$ gcc hello.c -o hello-dynamic
$ ls -l hello
-rwxr-xr-x 1 root root 80 hello.c
-rwxr-xr-x 1 root root 13724 hello-dynamic
-rwxr-xr-x 1 root root 1688756 hello-static
```

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

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 8

- Write and build simple “cos(sqrt(3.0))” program in C
 - Use `ldd` to investigate which dynamic libraries your hello world program loads
 - Use `strace` to investigate which system calls your hello world program makes
- 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!