CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Linking and Loading

We have assumed that an entire program must be in memory to execute, but that is not necessarily the case

Programs can have large sections of code that are rarely used, such as error handling routines

Code that is not actually run does not need to be loaded

Can save space in memory by using **dynamic loading**

Idea: do not load a routine until it is actually called

All function calls will first check whether code has been loaded into memory

If not, code is loaded (loader must make adjustmants to relocatable code)

Program proceeds as usual from there

Just as loading can be dynamic (i.e., at runtime), so can linking

Dynamically linked libraries are system libraries linked to user code at runtime

Contrast with **static linking**, where libraries (or at least the routines used) are included by the linker directly into the executable

Instead of including linked code, a **stub** is stored for each call to a library routine

Stub is small piece of code indicating

- where to find memory-resident library routine, and
- how to load routine into memory if not already there

Once stub has ensured routine is present in memory, it replaces itself with call to routine, so particular stub is executed at most once

With dynamic linking, only one copy of library code need be in memory regardless of how many programs use it

Benefits:

- Conserves disk space (smaller executables)
- Conserves main memory (only one copy of routine need be loaded)
- Updates to library do not require relinking of user code

One drawback to dynamic linking is that shared library must exist on system, whereas static linking includes everything necessary to run

Incompatible changes to libraries can break formerly working code

Similarly, different programs on same system may require different versions of shared library (Windows users used to call this problem DLL hell)

Straightforward enough to avoid some of these problems if program and library include version information

More than one version of library can be in memory at once, and each program will know which one it needs

Dynamic loading can be done without OS support

Dynamic linking requires OS intervention because multiple processes need to read memory where library resides