Building a Runnable Program

Chapter 15

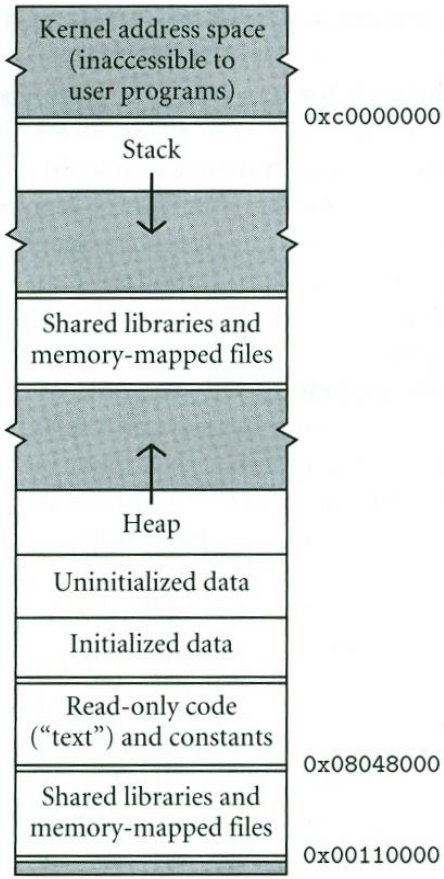
Programming Languages

Objective

Today's primary objective:

- Understand the format of runnable files
- Review the process of linking files together

Sample Layout of a Process Address Space



In early Unix systems with very limited memory, the stack grew downward from the bottom of the text segment; the number 0x08048000 is a legacy of these systems. The sections marked "Shared libraries and memory-mapped files" typically comprise multiple segments with varying permissions and addresses. (Modern Linux systems randomize the choice of addresses to discourage malware.) The top quarter of the address space belongs to the kernel. Just over 1 MB of space is left unmapped at the bottom of the address space to help catch program bugs in which small integer values are accidentally used as pointers.

Figure 14.8 Layout of process address space in x86 Linux (not to scale). Double lines separate regions with potentially different access permissions.

Linking Example

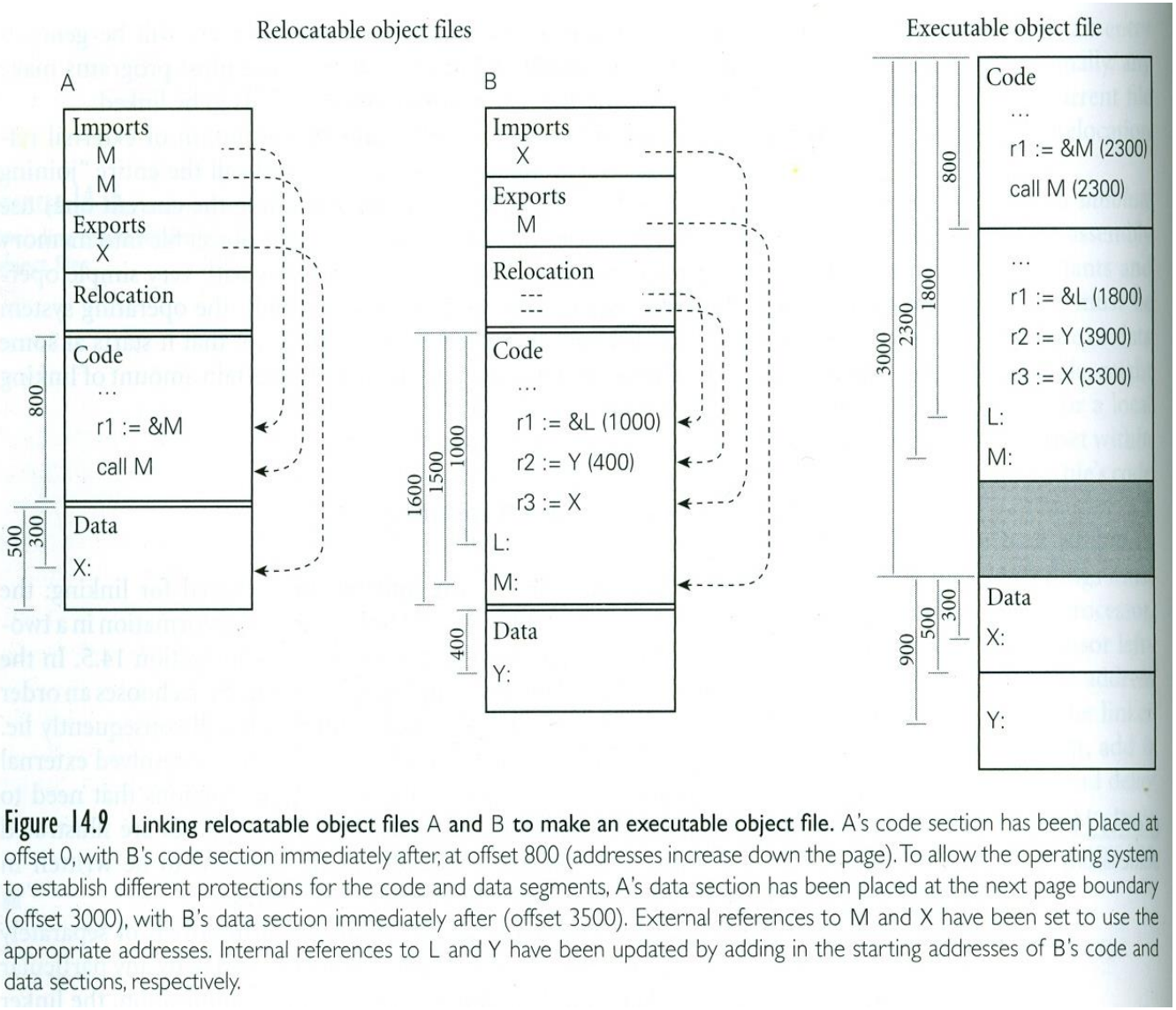


Figure 14.9 **Linking relocatable object files A and B to make an executable object file.** A's code section has been placed at offset 0, with B's code section immediately after, at offset 800 (addresses increase down the page). To allow the operating system to establish different protections for the code and data segments, A's data section has been placed at the next page boundary (offset 3000), with B's data section immediately after (offset 3500). External references to M and X have been set to use the appropriate addresses. Internal references to L and Y have been updated by adding in the starting addresses of B's code and data sections, respectively.