MODERN OPERATING SYSTEMS

Third Edition

ANDREW S. TANENBAUM

Chapter 1
Introduction

What Is An Operating System (1)

A modern computer consists of:

- One or more processors
- Main memory
- Disks
- Printers
- Various input/output devices

Managing all these components requires a layer of software – the **operating system**

What Is An Operating System (2)

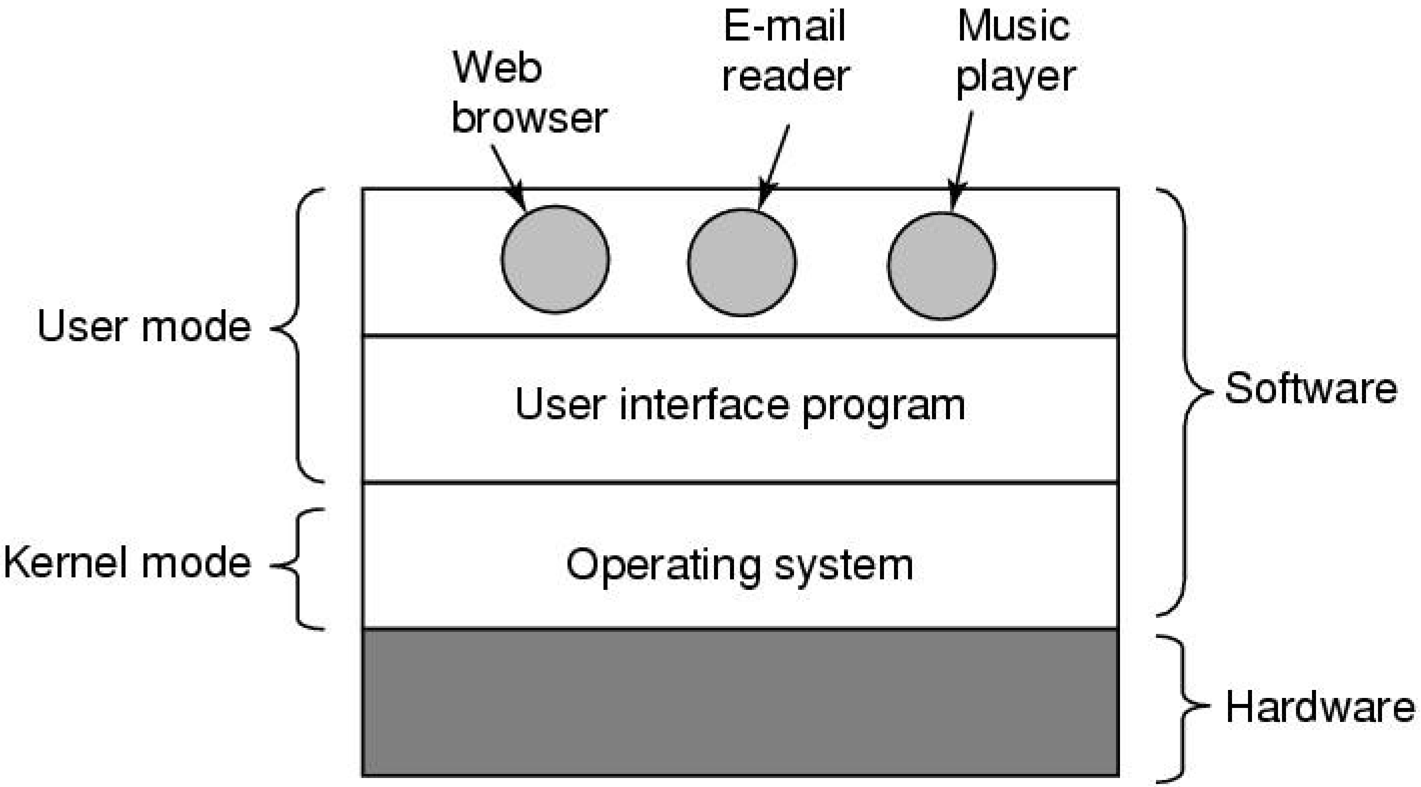


Figure 1-1. Where the operating system fits in.

The Operating System as an Extended Machine

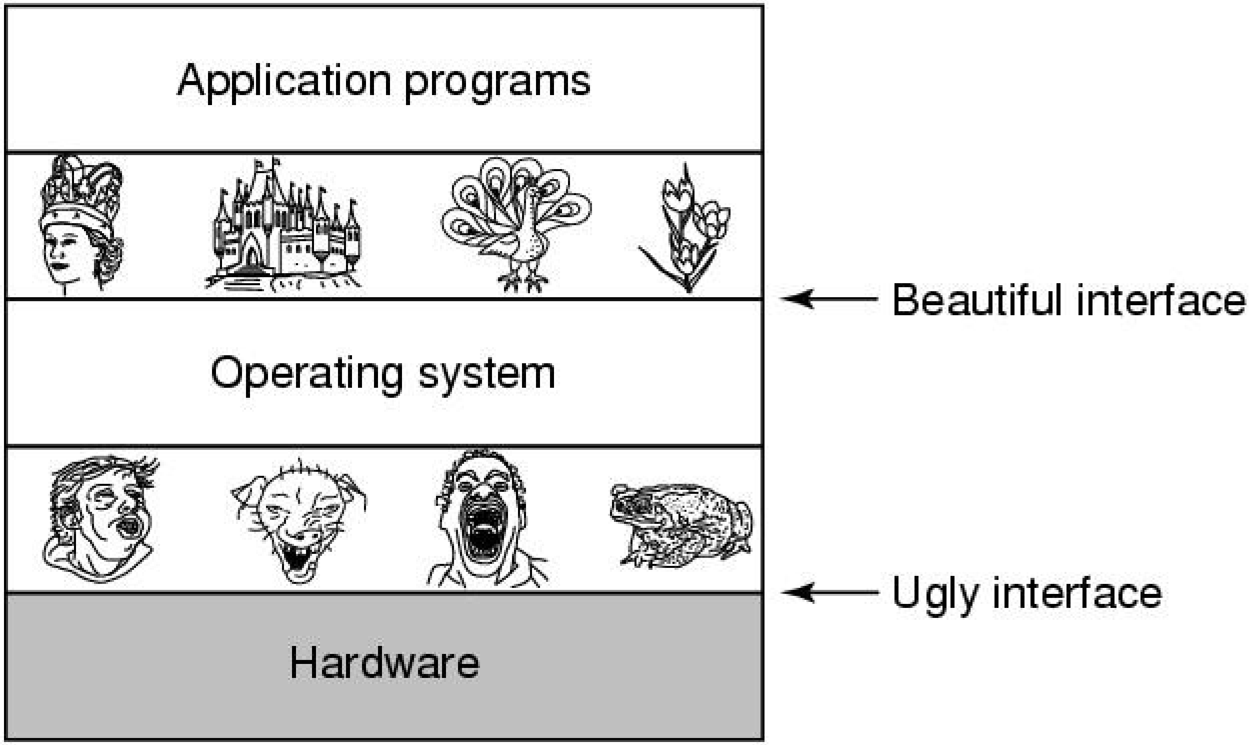


Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.

The Operating System as a Resource Manager

- Allow multiple programs to run at the same time
- Manage and protect memory, I/O devices, and other resources
- Includes multiplexing (sharing) resources in two different ways:
  - In time
  - In space

Computer Hardware Review

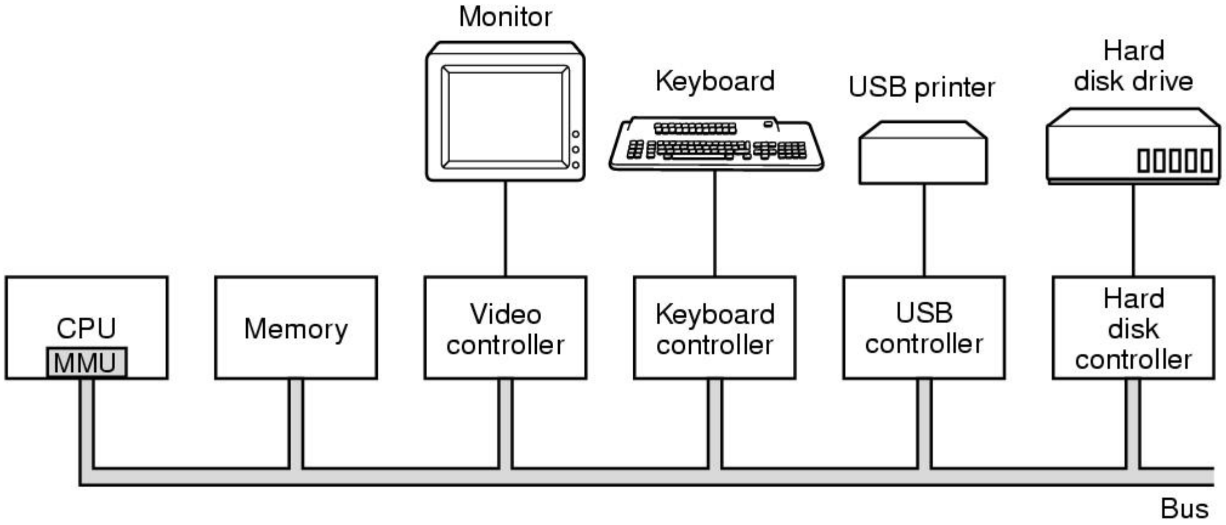


Figure 1-6. Some of the components of a simple personal computer.

CPU Pipelining

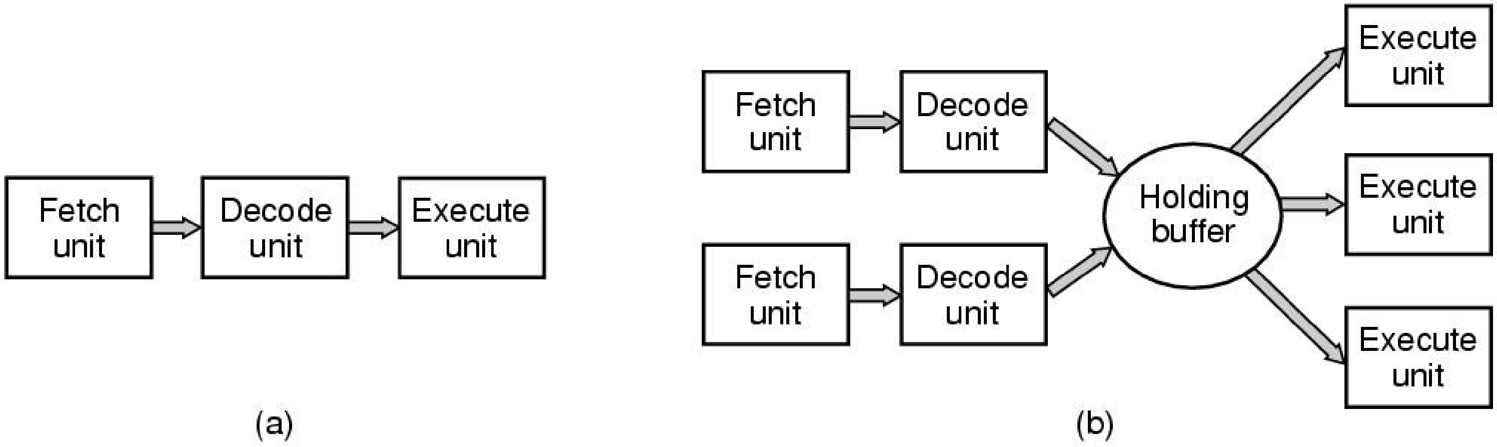


Figure 1-7. (a) A three-stage pipeline. (b) A superscalar CPU.

Multithreaded and Multicore Chips

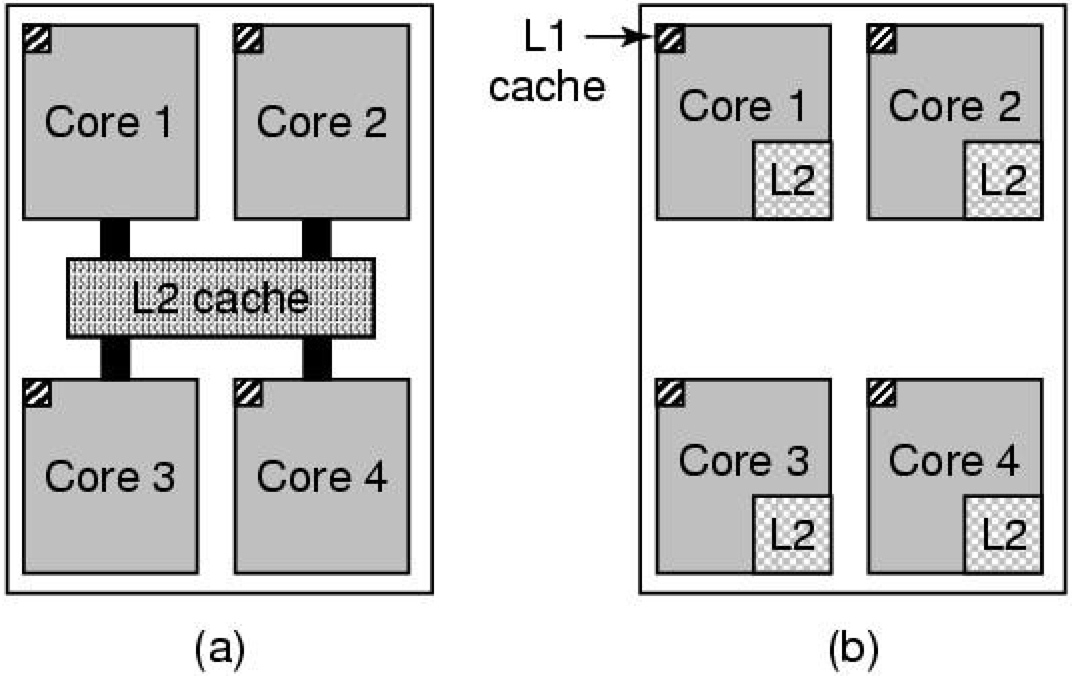
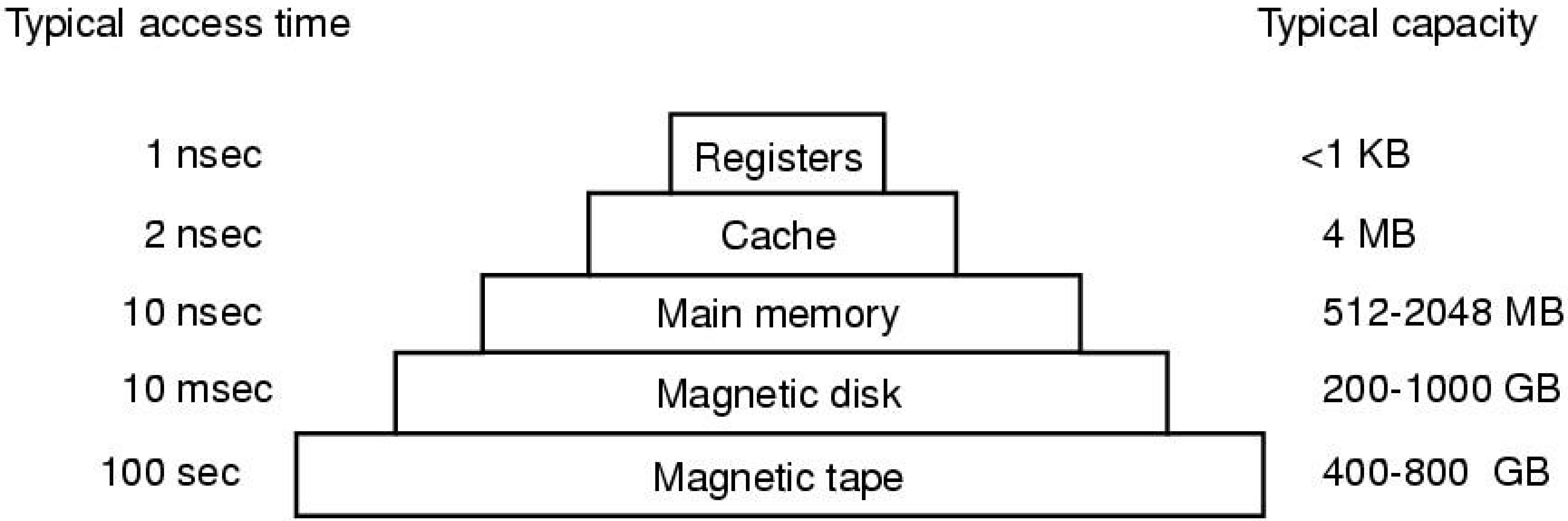


Figure 1-8. (a) A quad-core chip with a shared L2 cache.
(b) A quad-core chip with separate L2 caches.

Memory (1)



Typical access time		Typical capacity
1 nsec	Registers	<1 KB
2 nsec	Cache	4 MB
10 nsec	Main memory	512-2048 MB
10 msec	Magnetic disk	200-1000 GB
100 sec	Magnetic tape	400-800 GB

Figure 1-9. A typical memory hierarchy.
The numbers are very rough approximations.

Memory (2)

Questions when dealing with cache:

- When to put a new item into the cache.
- Which cache line to put the new item in.
- Which item to remove from the cache when a slot is needed.
- Where to put a newly evicted item in the larger memory.

Disks

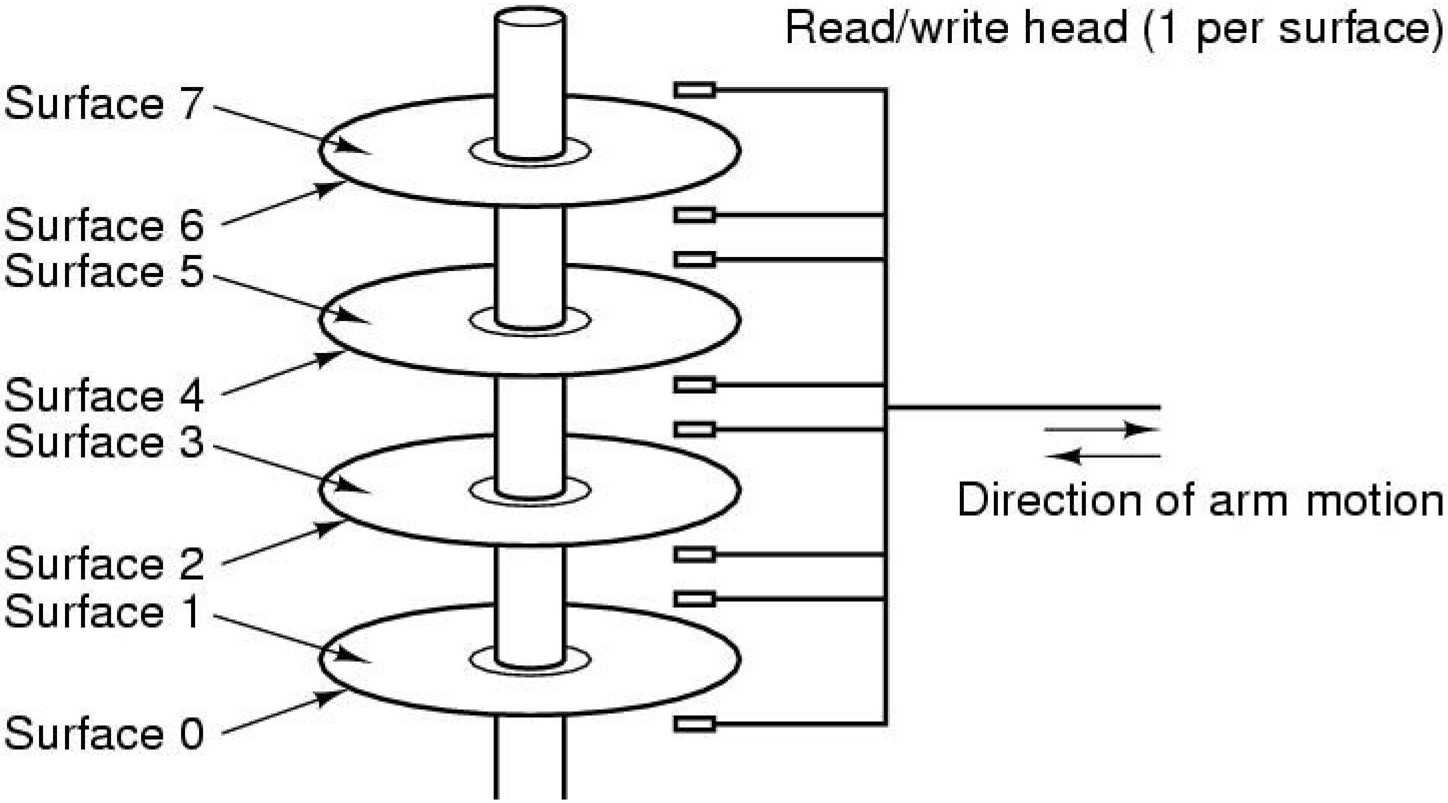


Figure 1-10. Structure of a disk drive.

I/O Devices

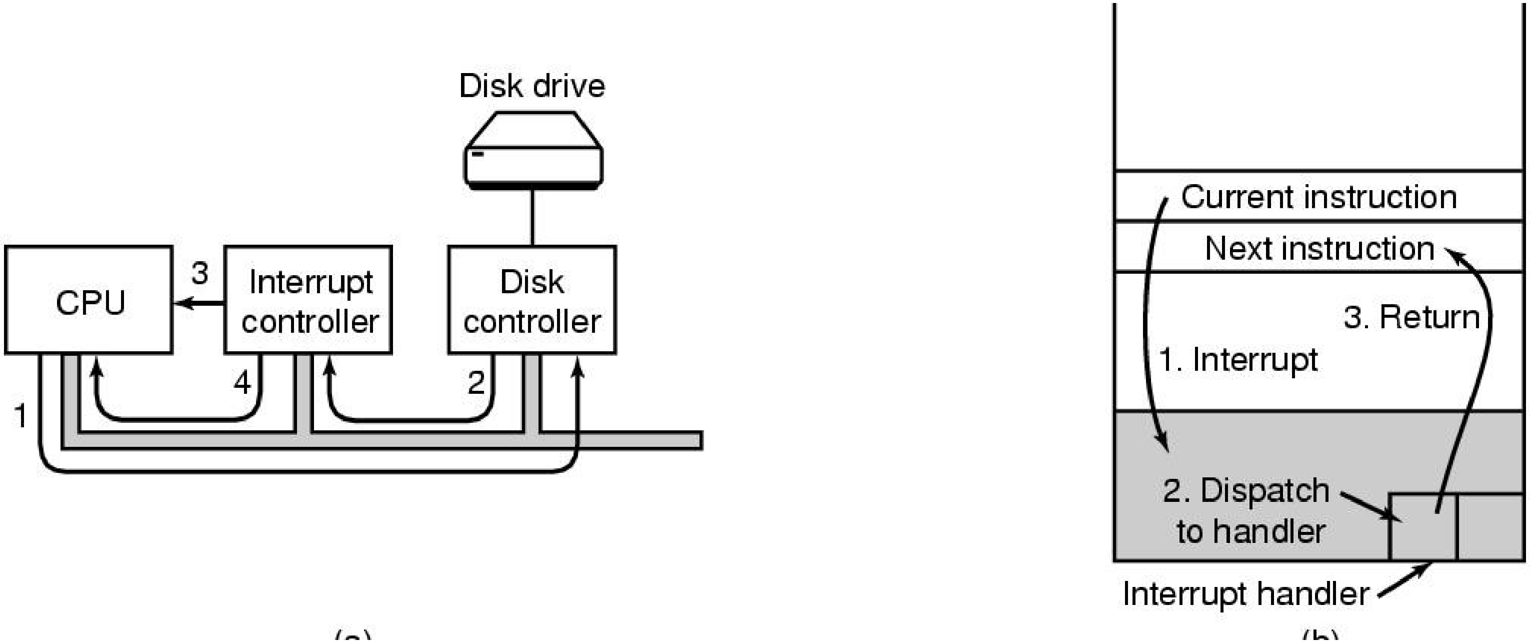


Figure 1-11. (a) The steps in starting an I/O device and getting an interrupt.

Buses

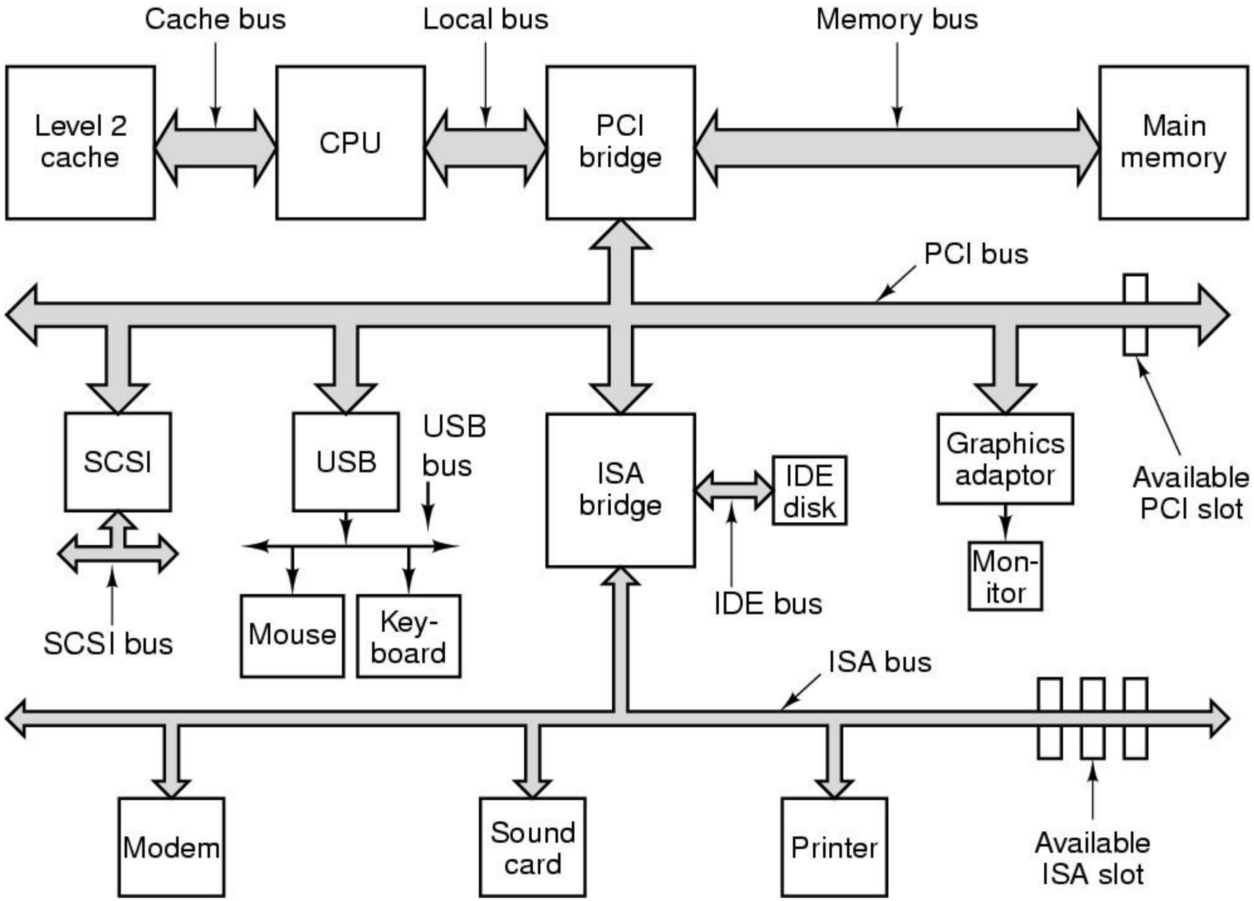


Figure 1-12. The structure of a large Pentium system

The Operating System Zoo

- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Handheld operating systems
- Embedded operating systems
- Real-time operating systems

Operating System Concepts

- Processes
- Address spaces
- Files
- Input/Output
- Protection
- The shell

Processes

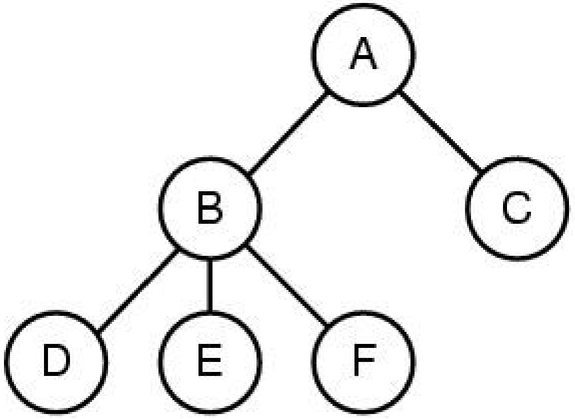


Figure 1-13. A process tree. Process A created two child processes, B and C. Process B created three child processes, D, E, and F.

Files (1)

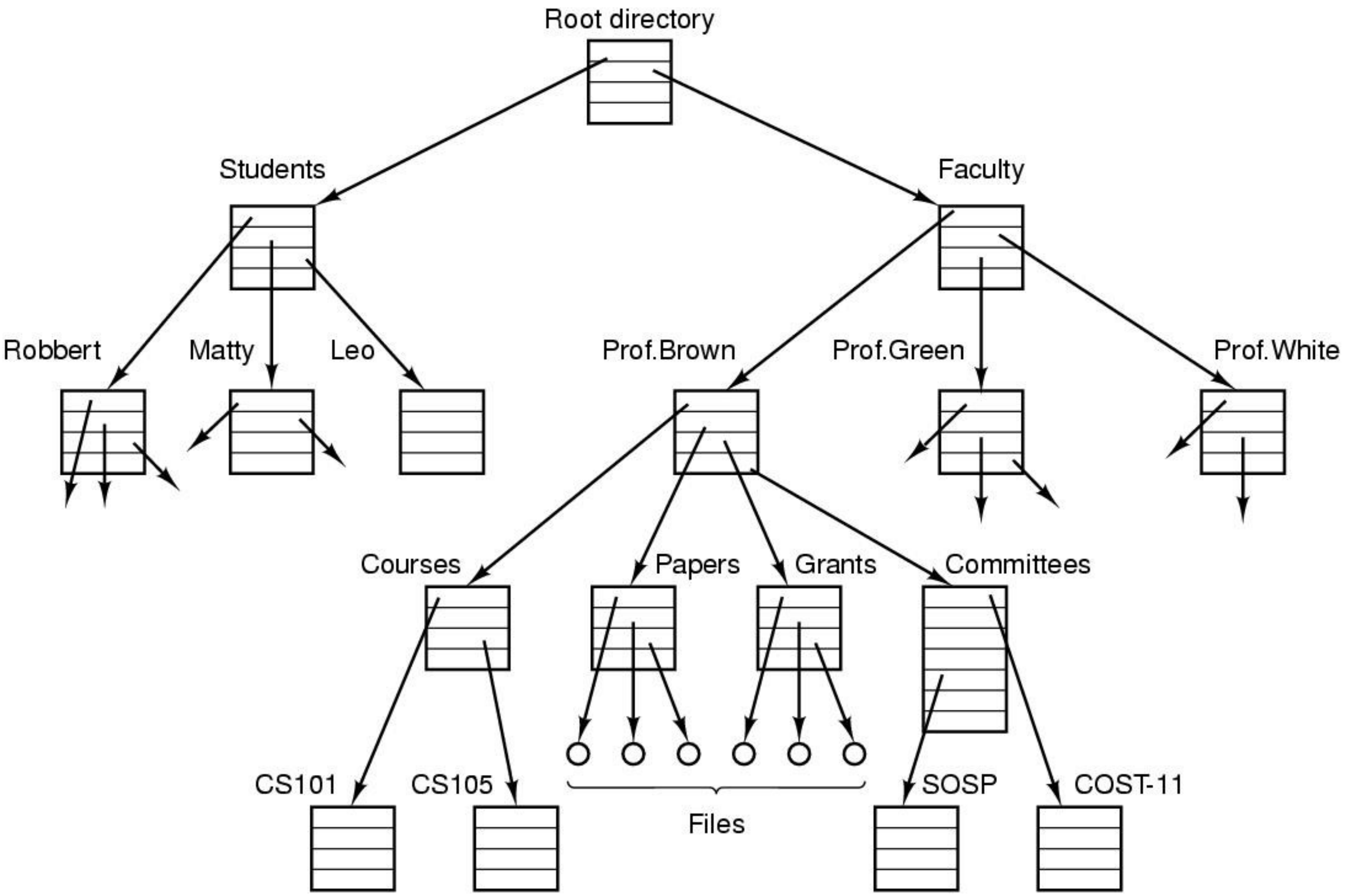


Figure 1-14. A file system for a university department.

Files (2)

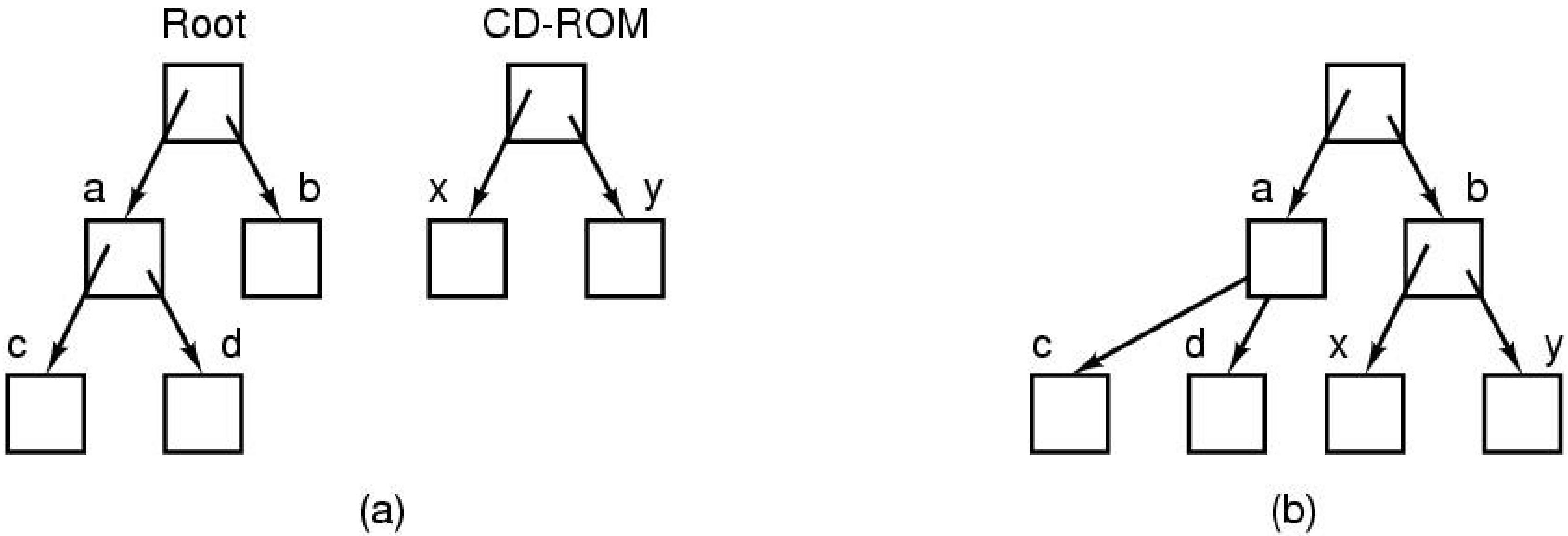


Figure 1-15. (a) Before mounting, the files on the CD-ROM are not accessible. (b) After mounting, they are part of the file hierarchy.

Files (3)

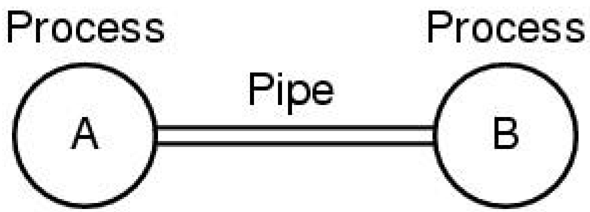


Figure 1-16. Two processes connected by a pipe.

System Calls

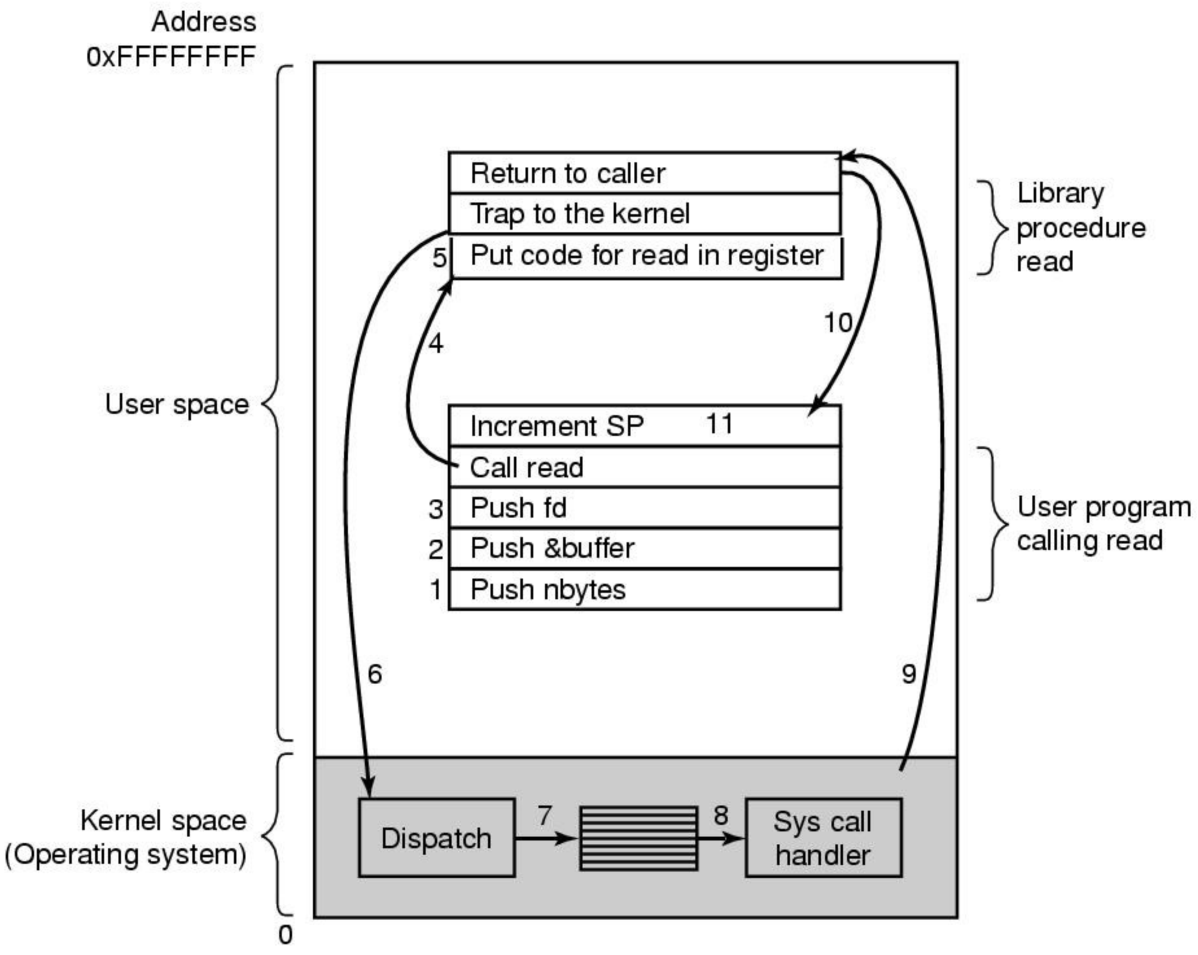
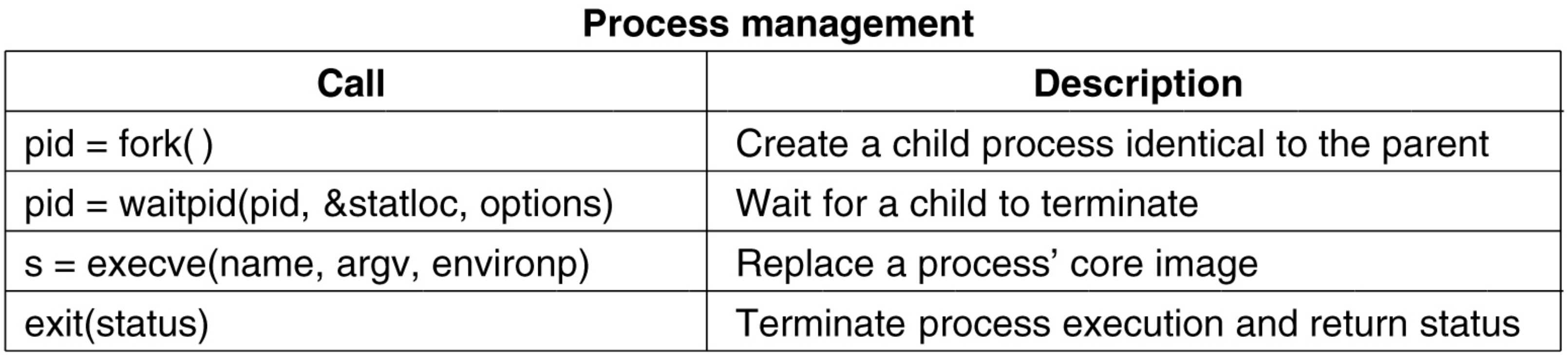


Figure 1-17. The 11 steps in making the system call read(fd, buffer, nbytes).

System Calls for Process Management

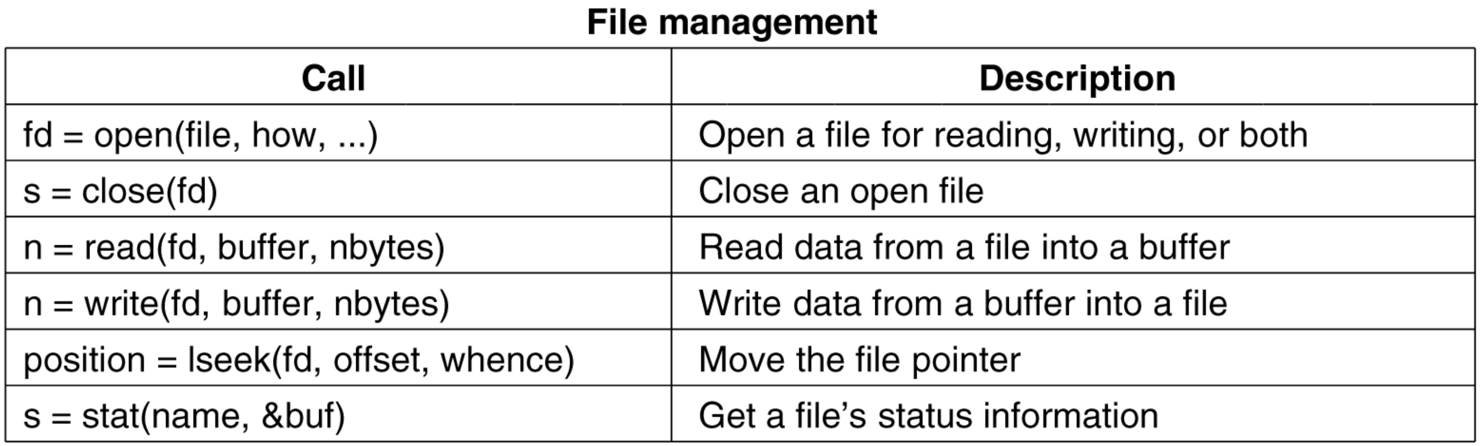


Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process’ core image
exit(status)	Terminate process execution and return status

Figure 1-18. Some of the major POSIX system calls.

System Calls for File Management (1)

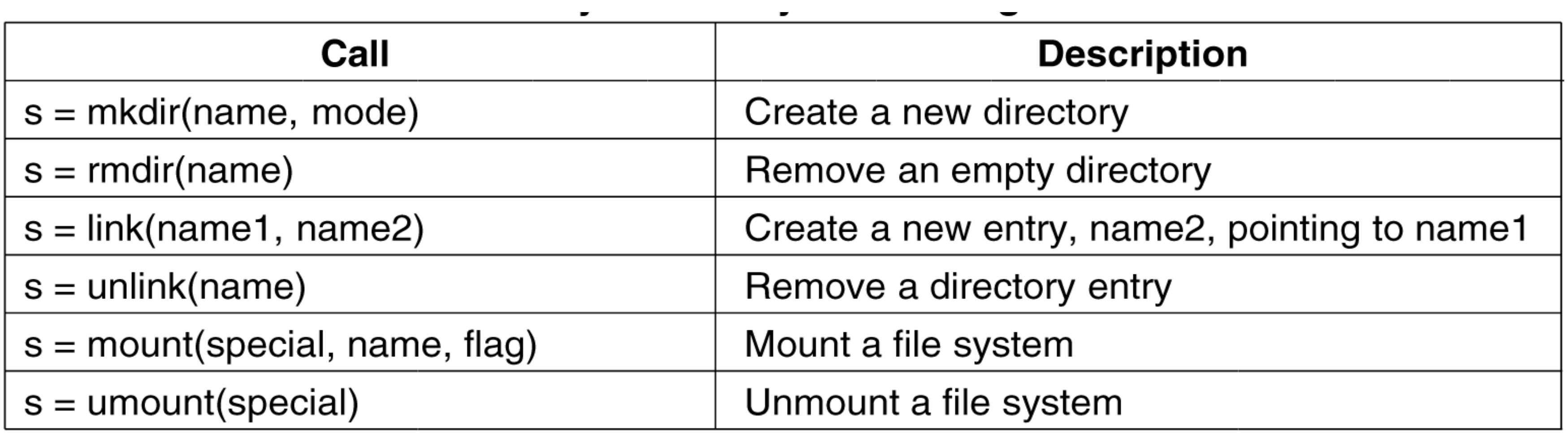


File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Figure 1-18. Some of the major POSIX system calls.

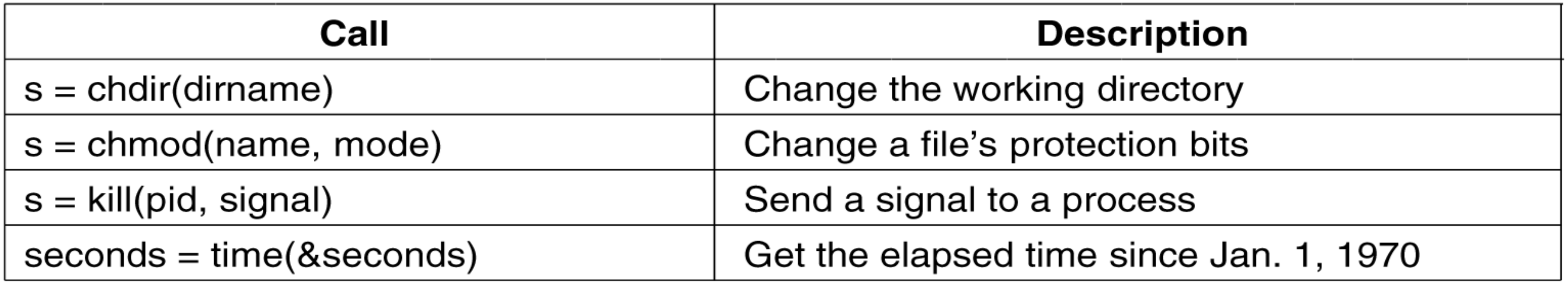
System Calls for File Management (2)



Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Figure 1-18. Some of the major POSIX system calls.

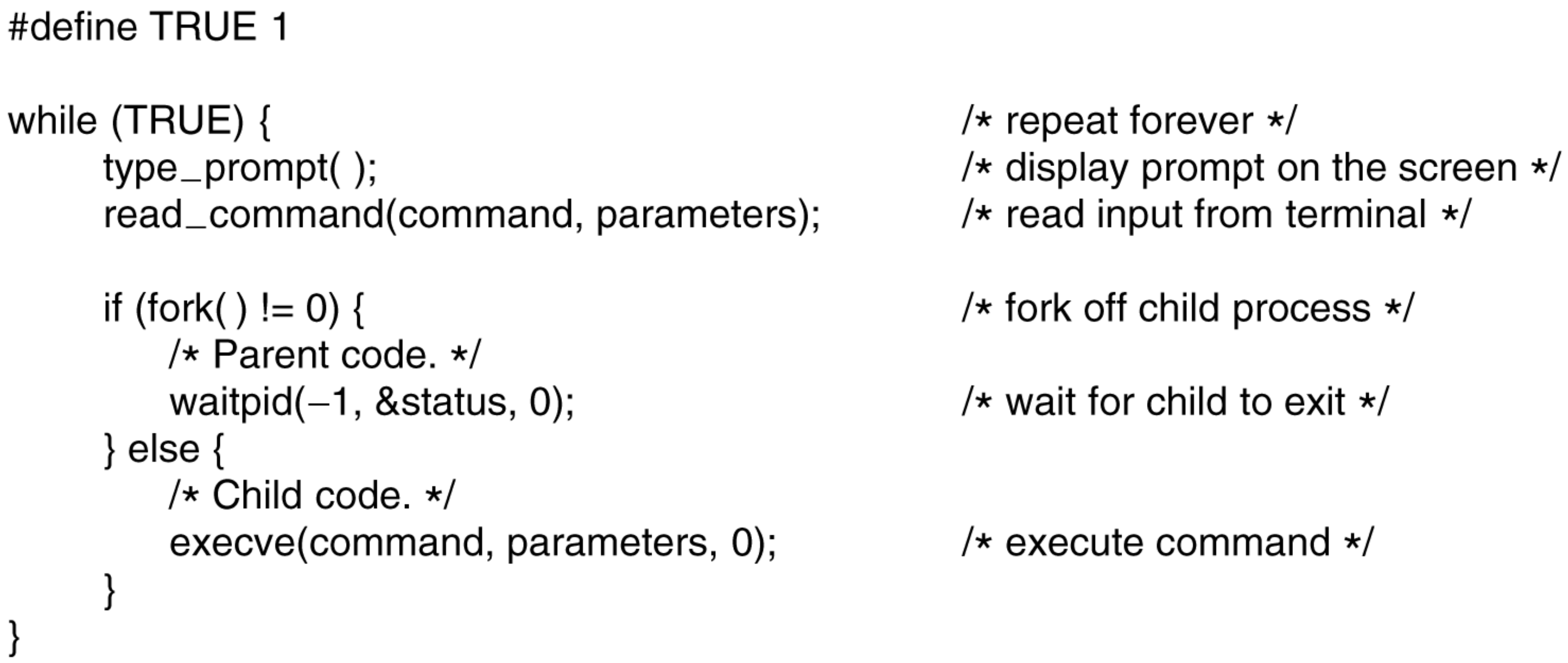
Miscellaneous System Calls



Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file’s protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Figure 1-18. Some of the major POSIX system calls.

A Simple Shell



```
#define TRUE 1

while (TRUE) {                                   /* repeat forever */
     type_prompt( );                             /* display prompt on the screen */
     read_command(command, parameters);          /* read input from terminal */

     if (fork( ) != 0) {                         /* fork off child process */
         /* Parent code. */
         waitpid(–1, &status, 0);                /* wait for child to exit */
     } else {
         /* Child code. */
         execve(command, parameters, 0);         /* execute command */
     }
}
```

Figure 1-19. A stripped-down shell.

Memory Layout

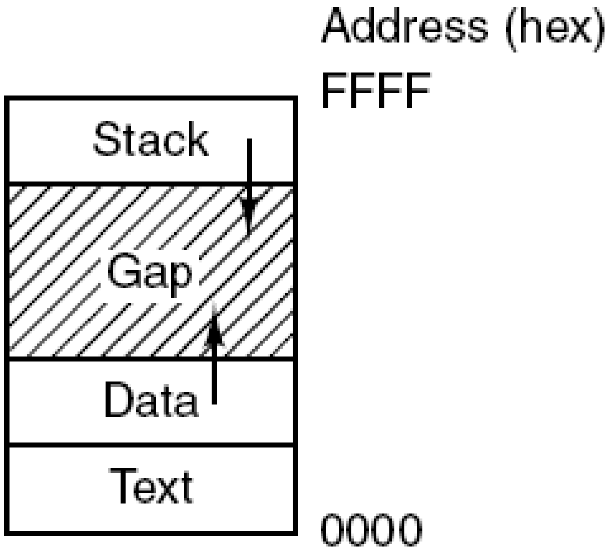
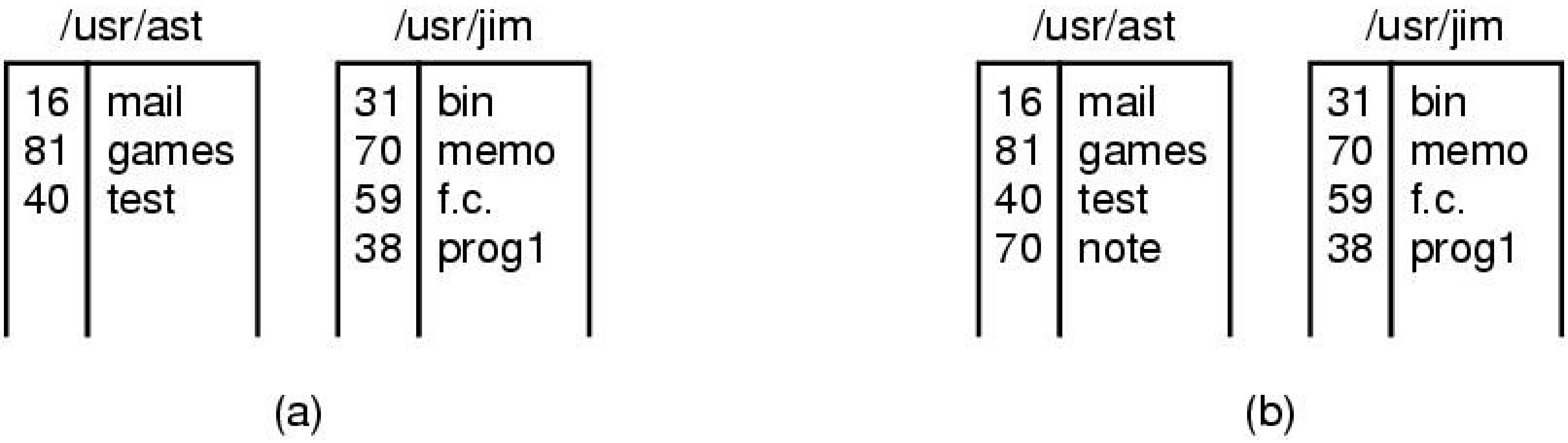


Figure 1-20. Processes have three segments: text, data, and stack.

Linking



/usr/ast

16	mail
81	games
40	test

/usr/jim

31	bin
70	memo
59	f.c.
38	prog1

(a)

/usr/ast

16	mail
81	games
40	test
70	note

/usr/jim

31	bin
70	memo
59	f.c.
38	prog1

(b)

Figure 1-21. (a) Two directories before linking */usr/jim/memo* to ast’s directory. (b) The same directories after linking.

Mounting

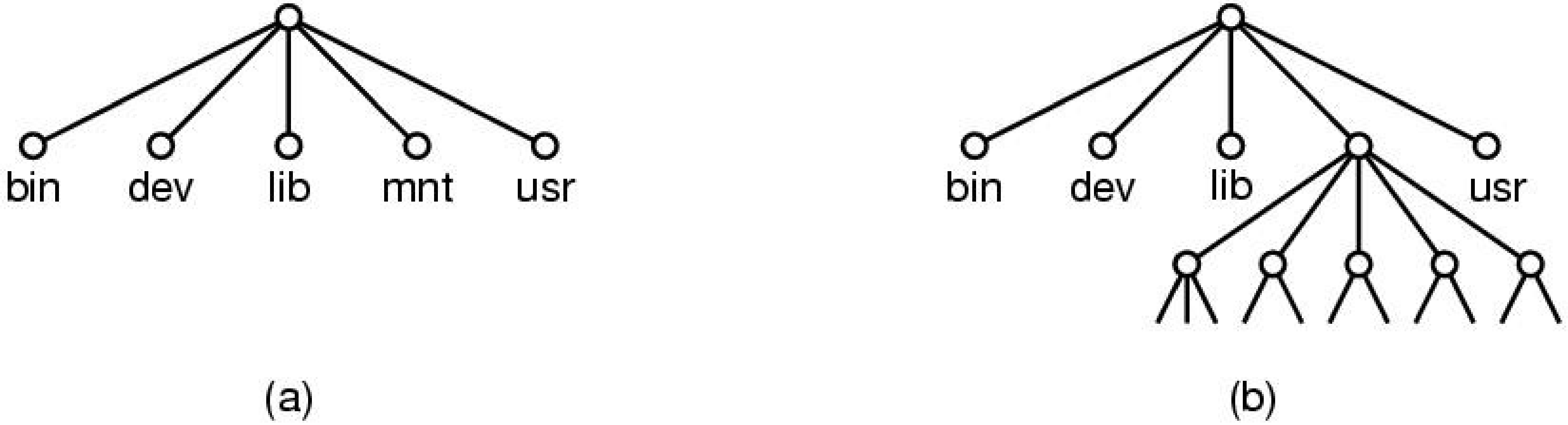


Figure 1-22. (a) File system before the mount.
(b) File system after the mount.

Operating Systems Structure

Monolithic systems – basic structure:

- A main program that invokes the requested service procedure.
- A set of service procedures that carry out the system calls.
- A set of utility procedures that help the service procedures.

Microkernels

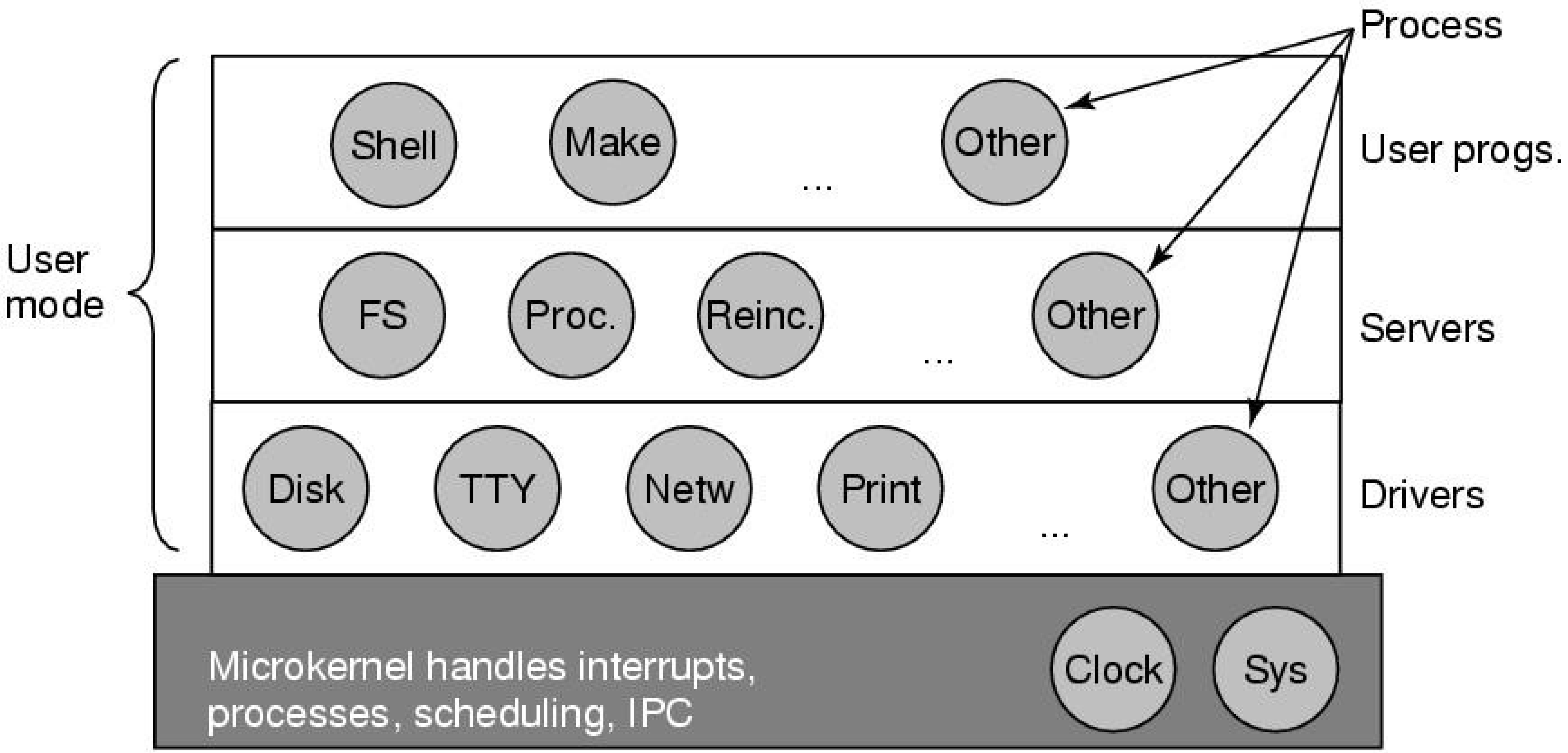


Figure 1-26. Structure of the MINIX 3 system.

Client-Server Model

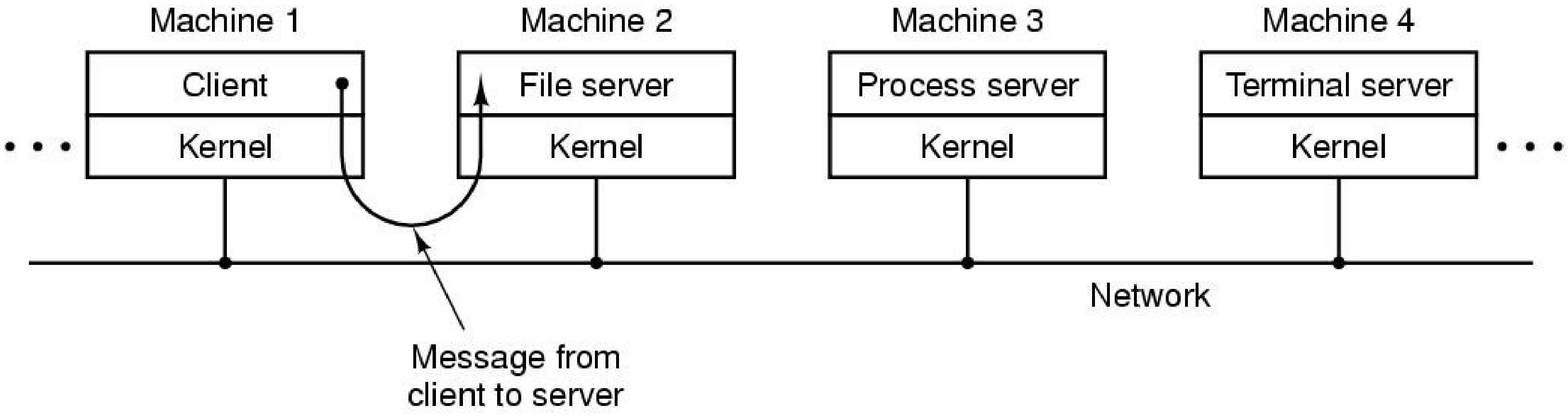


Figure 1-27. The client-server model over a network.

Virtual Machines

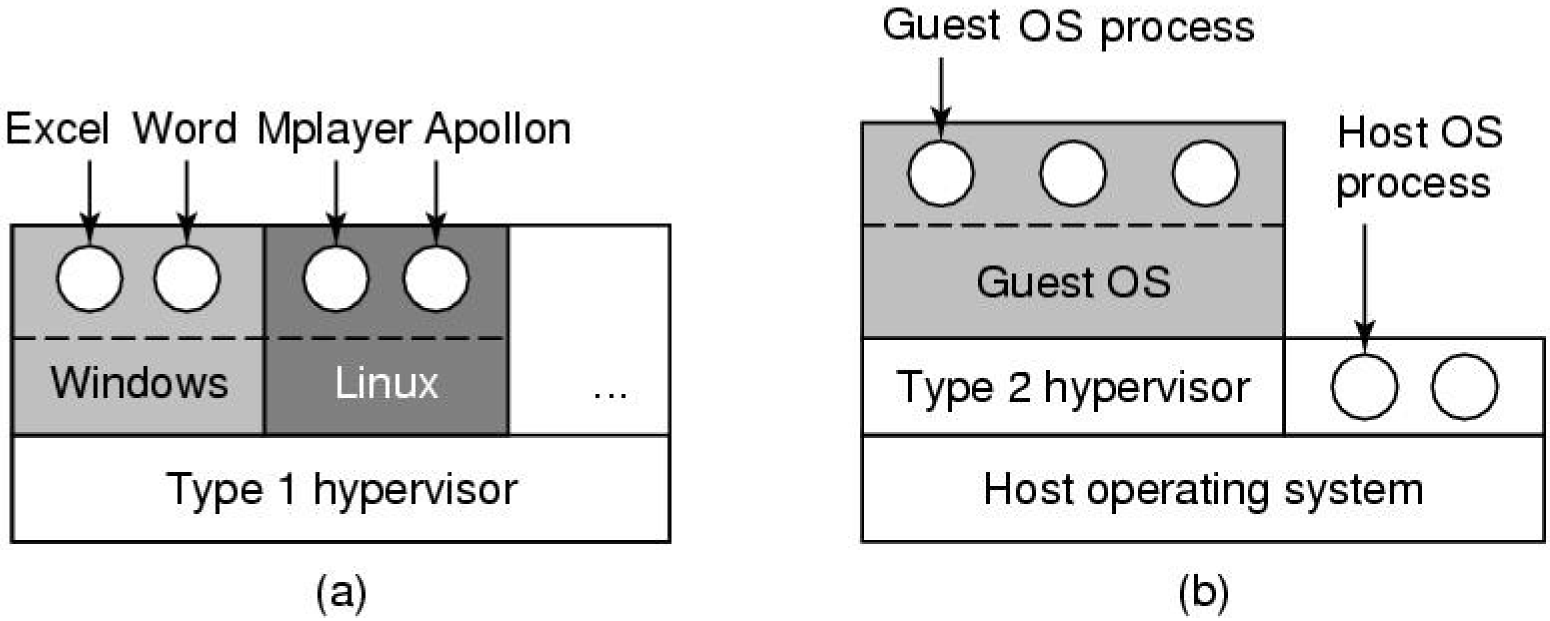


Figure 1-29. (a) A type 1 hypervisor. (b) A type 2 hypervisor.

The Model of Run Time

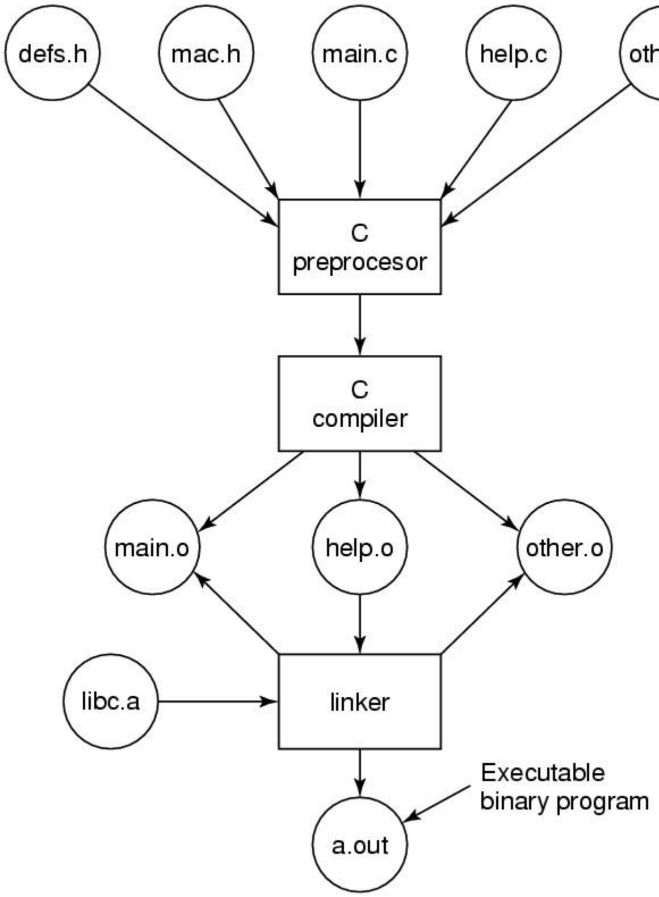


Figure 1-30. The process of compiling C and header files to make an executable.