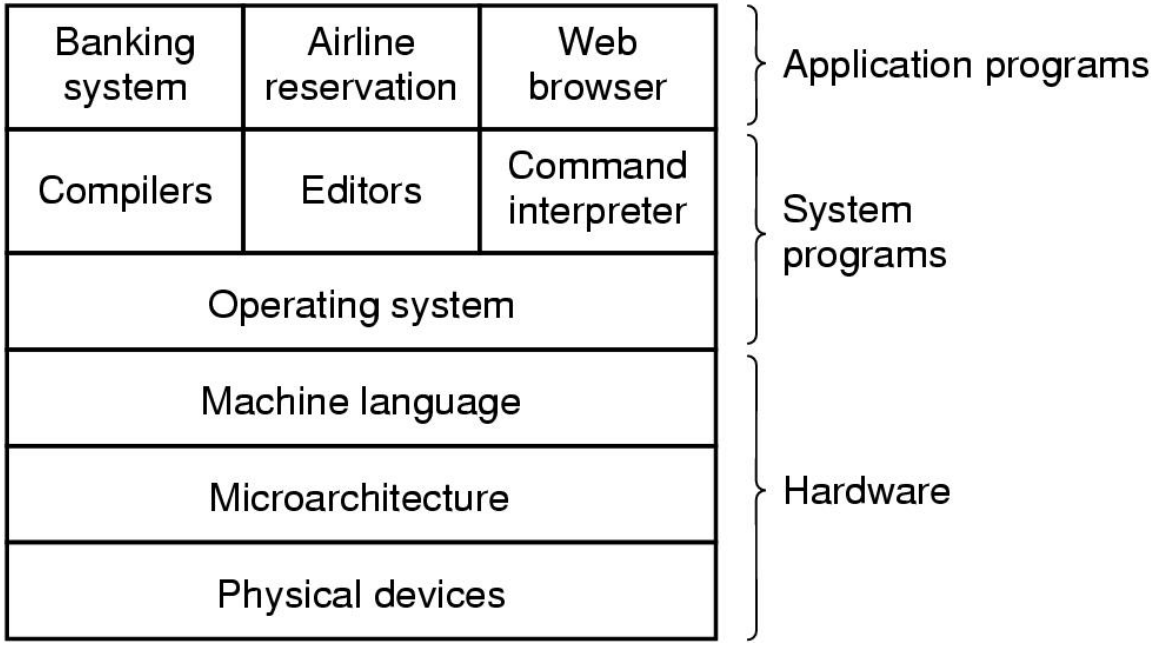
Introduction to Operating Systems

B. Ramamurthy

(adapted from C. Egert's and W. Stallings' slides)

Introduction



- A computer system consists of
  - hardware
  - system programs
  - application programs

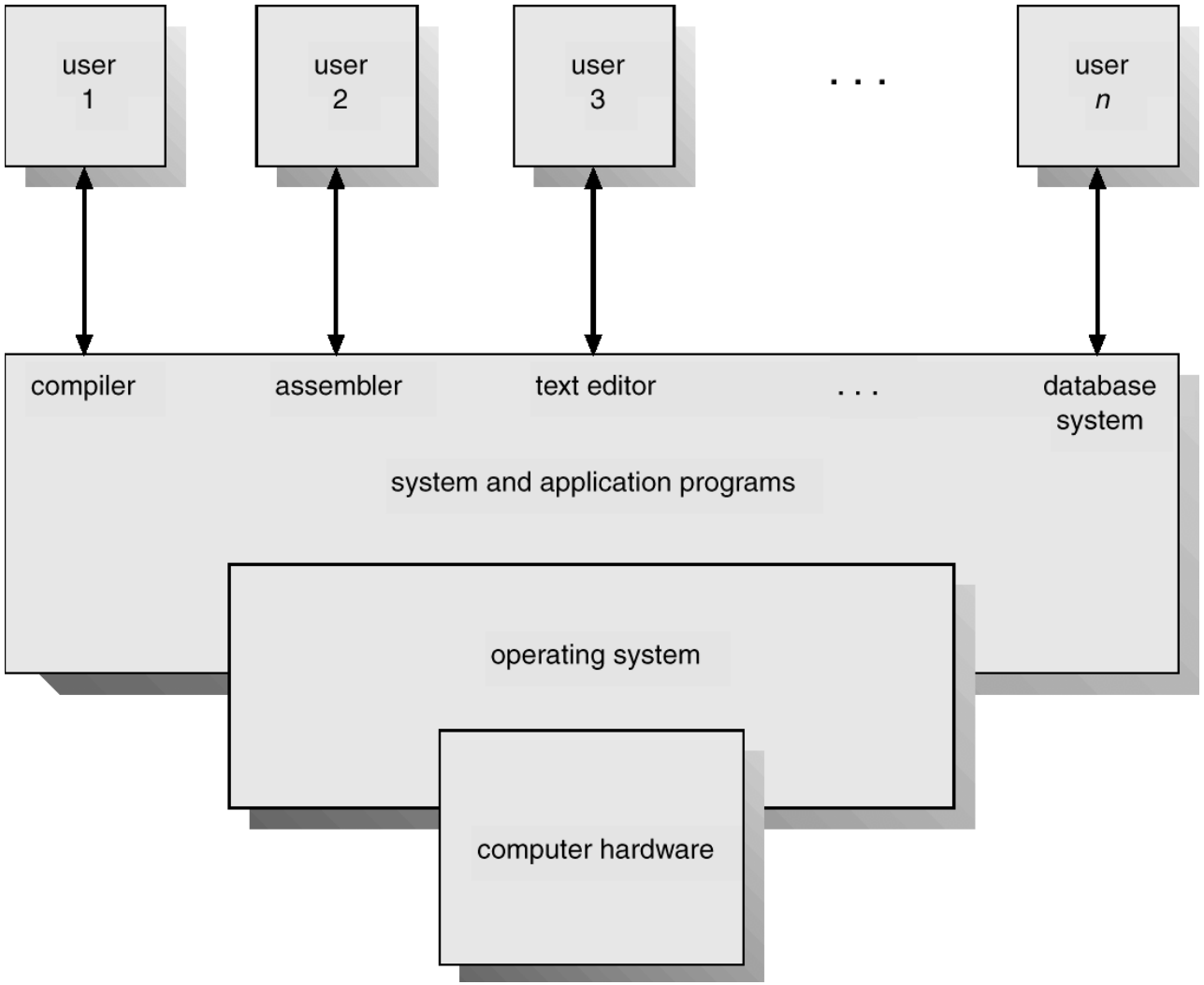
An Operating System?

- What is an Operating System?
  - A program that acts as an intermediary between a user of a computer and the computer hardware.
- What is the purpose of an operating system?
  - To provide an environment in which a user can execute programs.
- What are the goals of an Operating System?
  - The primary goal of an Operating System is to make the computer system convenient to use.
  - The secondary goal is to make the computer system efficient to use.

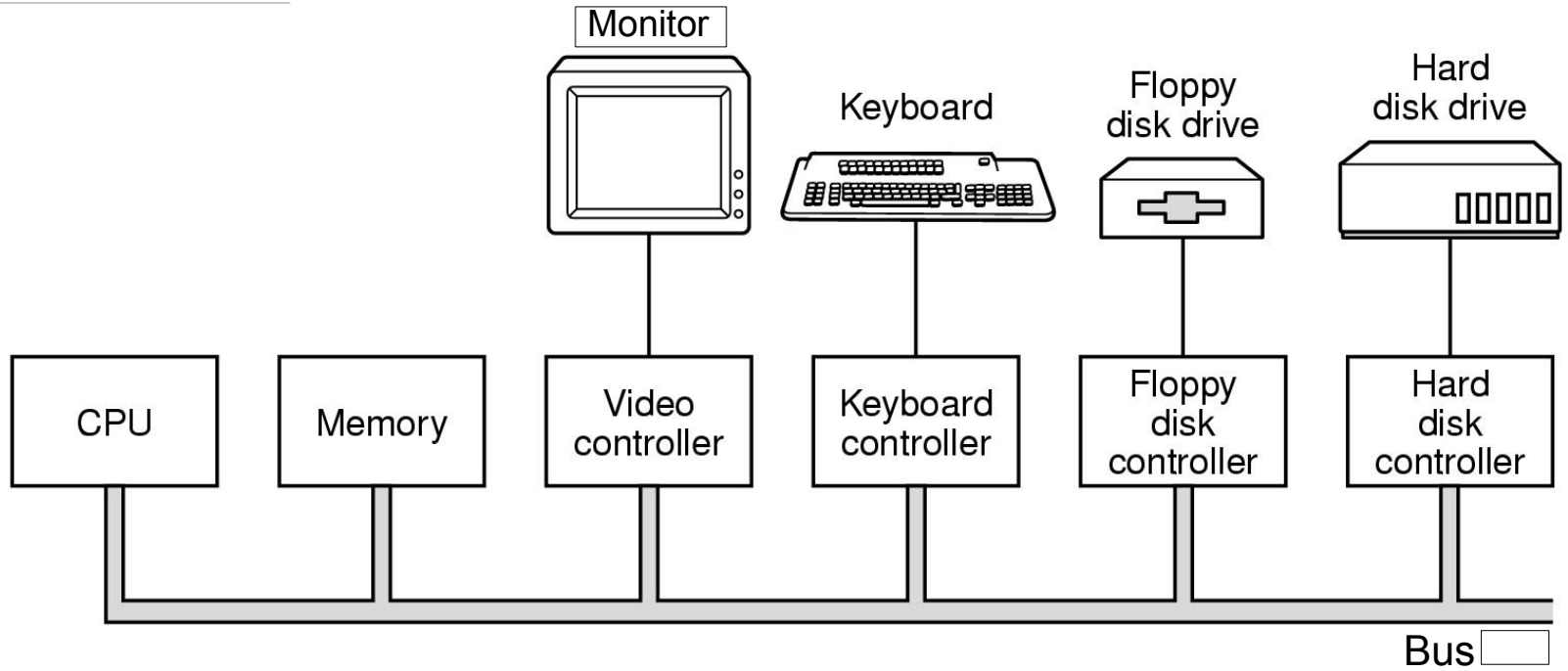
Computer System Components

- Hardware – provides basic computing resources (CPU, memory, I/O devices).
- Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
- Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
- Users (people, machines, other computers).

Abstract View of System Components

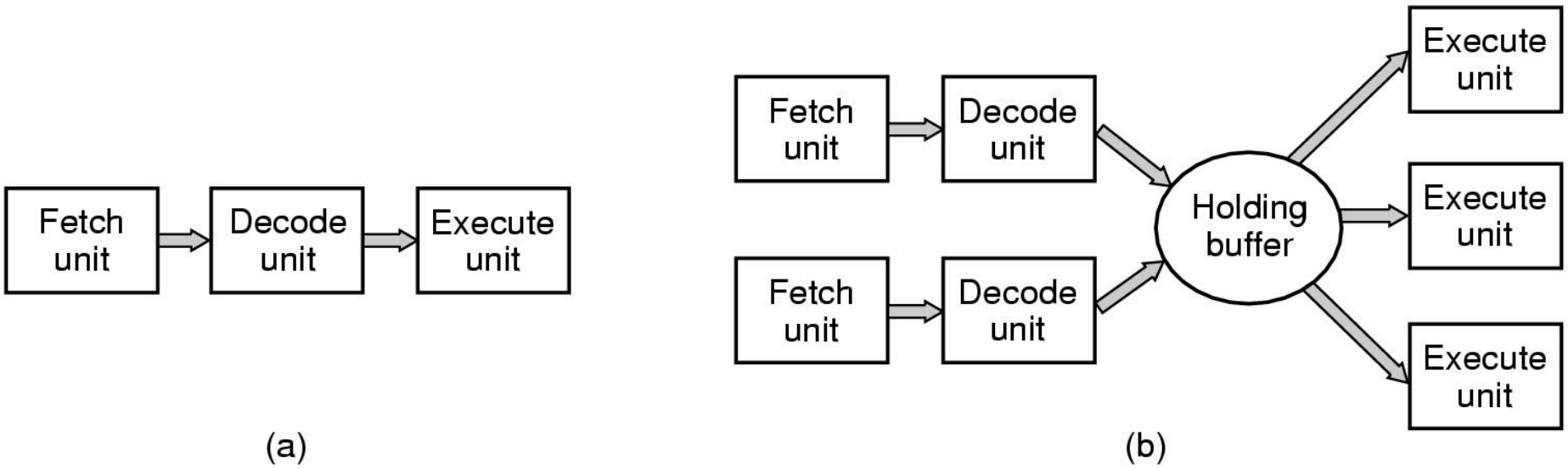


Computer Hardware Review (1)



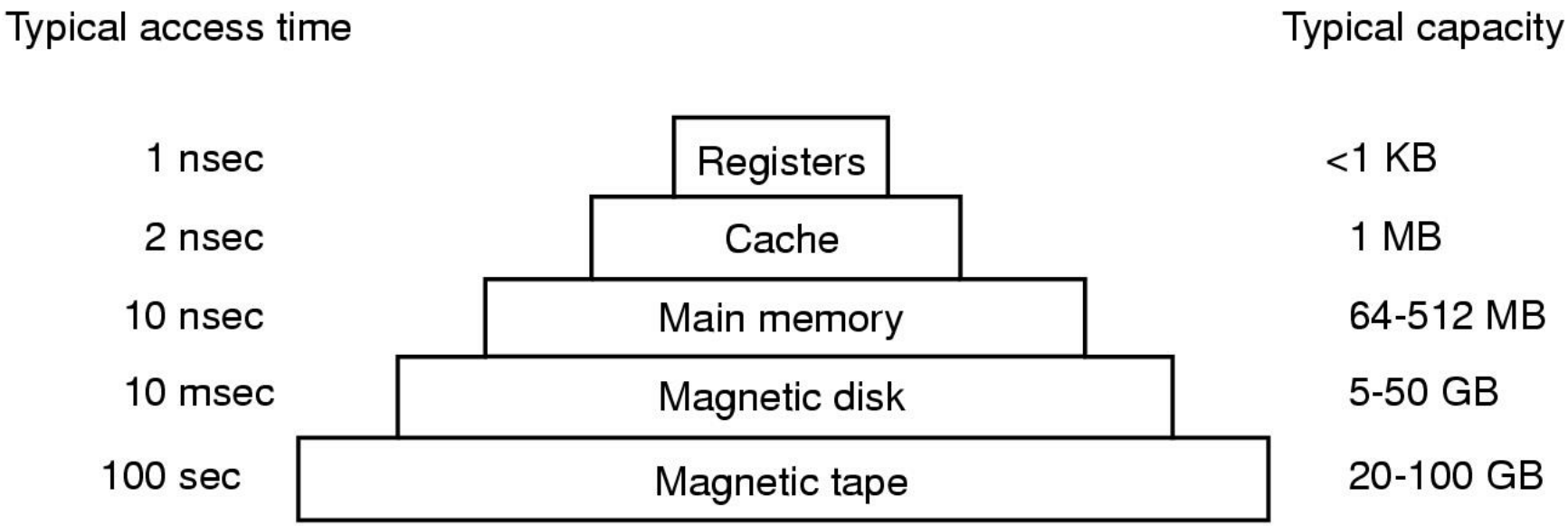
- Components of a simple personal computer

Computer Hardware Review (2)



(a) A three-stage pipeline
(b) A superscalar CPU

Computer Hardware Review (3)



- Typical memory hierarchy
  - numbers shown are rough approximations

Function of Operating System

- OS as Extended machine
  - Computer Architecture shows that computer is made up of chips and wires
  - We do not want to program on the bare metal
  - Virtual machine creates a hardware abstraction
  - Abstract machine can provide hardware independent interfaces
  - Increase portability
  - Allow greater protection
  - Implication is that it is much faster and easier to program with less errors

Function of Operating System

- OS as resource manager
  - Coordination and control of limited resources such as memory, disk, network, etc
  - Deal with resource conflicts
  - Deal with resource fairness
  - Make access efficient as possible

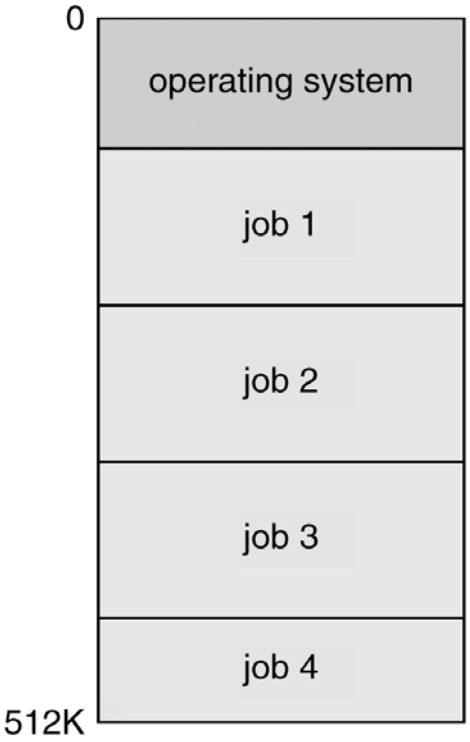
Parts of an Operating System

- No universal agreement on the topic, but most likely
  - Memory Management
  - IO Management
  - CPU Scheduling
  - IPC
  - MultiTasking/Multiprogramming

  (On some Operating System, this functionality is provided by a single program known as the *kernel)*
- What about?
  - File System
  - Multimedia Support
  - UI (X Windows, MSWin)
  - Internet Browser?
  - Why would extras be important

Multiprogramming

- Memory partitioned into several pieces
- CPU Starts a job
- If the job is waiting for IO, the CPU can switch to another task



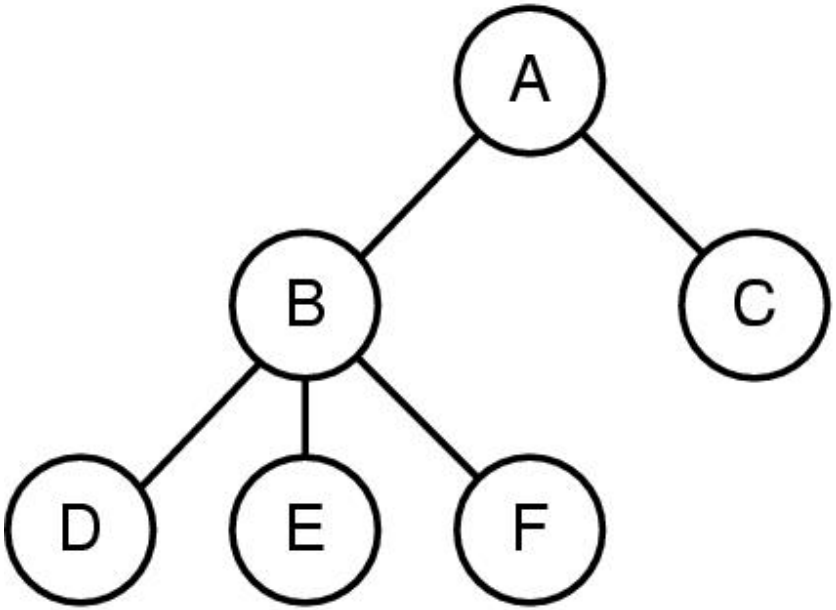
Multitasking (Time-sharing)

- Extension of Multiprogramming
  - Need for user interactivity
  - Instead of switching jobs when waiting for IO, a timer causes jobs to switch
- User interacts with computer via CRT and keyboard
  - Systems have to balance CPU utilization against response time
  - Better device management
- Need for file system to allow user to access data and code
- Need to provide user with an “interaction environment”

Virtual Memory

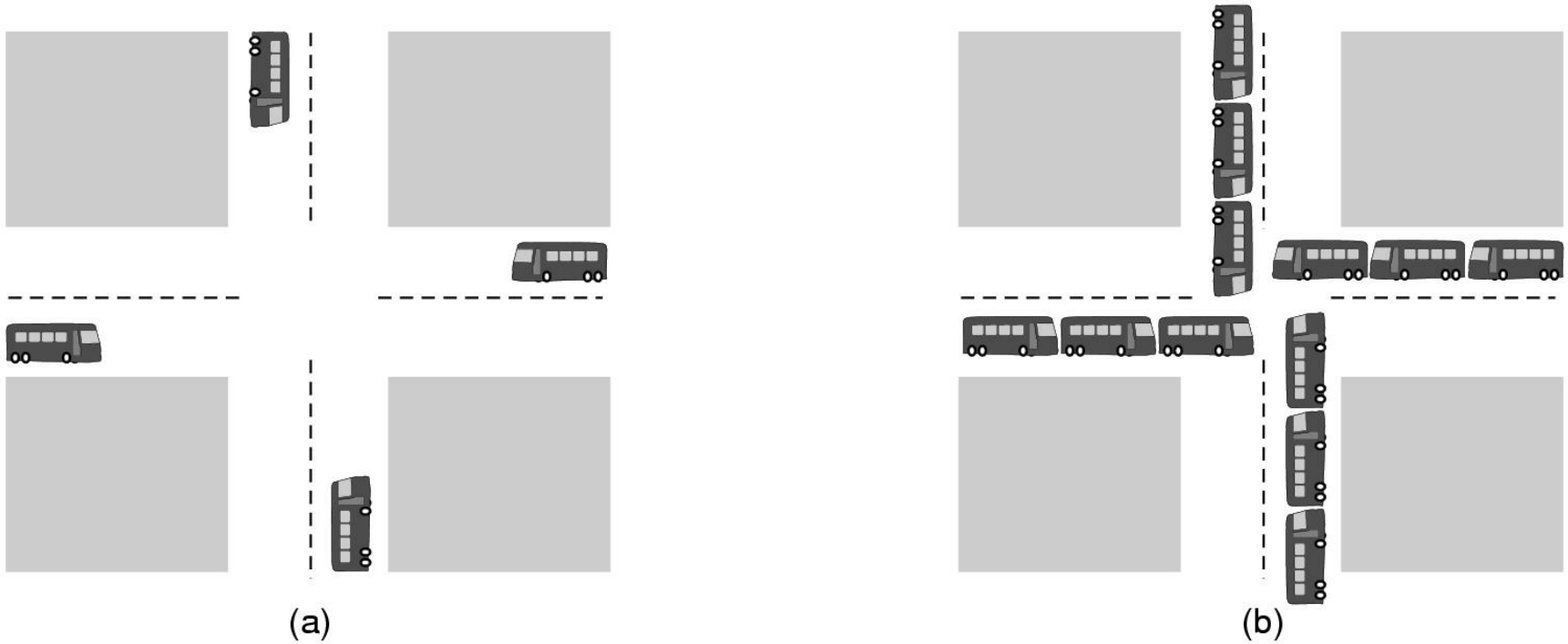
- Programs can be larger than memory
  - Program loaded into memory as needed
  - Active program and data “swapped” to a disk until needed
- Memory space treated uniformly

Operating System Concepts (1): Process Management



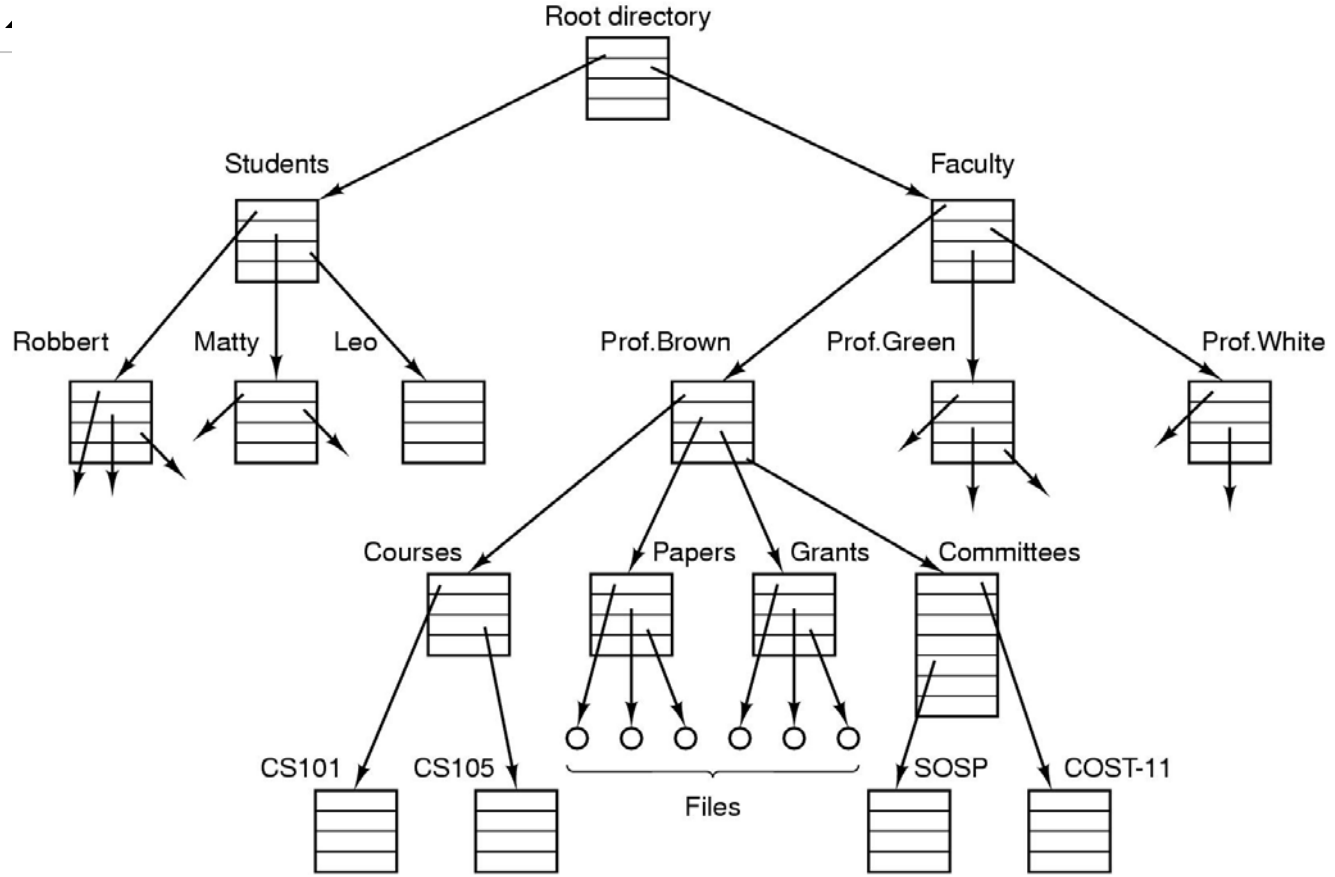
- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F

Operating System Concepts (2): Deadlock Handling



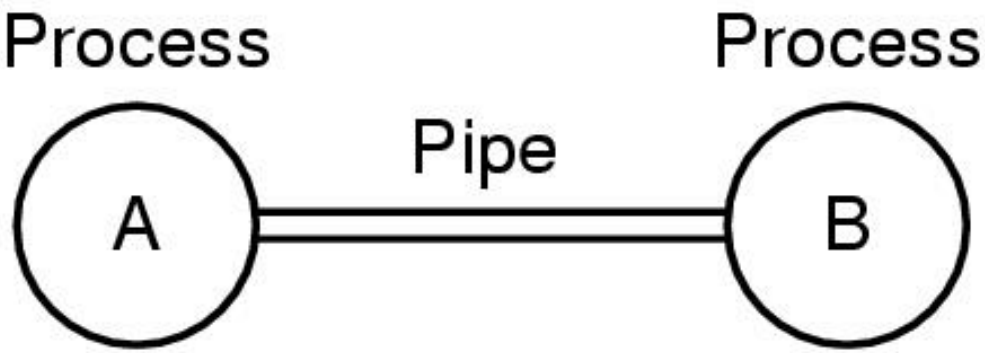
(a) A potential deadlock. (b) an actual deadlock.

Operating System Concepts (3): File System



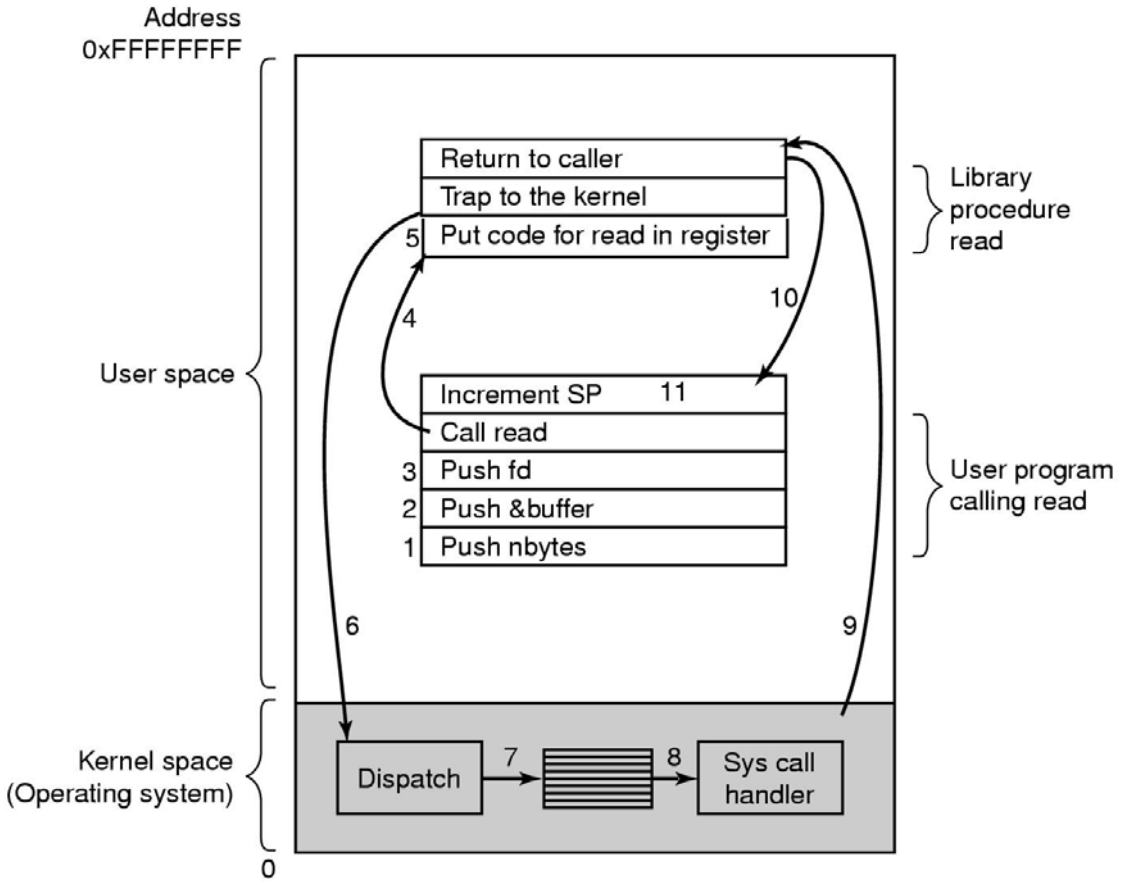
File system for a university department

Operating System Concepts (5): Inter-process Communication



Two processes connected by a pipe

Steps in Making a System Call



There are 11 steps in making the system call

read (fd, buffer, nbytes)

Some System Calls For Process Management and File 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

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

Metric Units

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

The metric prefixes

Parallel Systems

- *Symmetric multiprocessing (SMP)*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints (known as *deterministic*).
- *Hard real-time system*.
  - Secondary storage limited or absent, data stored in short-term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- *Soft real-time system*
  - Limited utility in industrial control or robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications