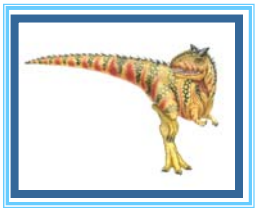
Chapter 1: Introduction



(slides selected/modified/added by R. Doemer, 03/30/10)

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- I/O Structure and Operation (inserted from Chapter 13)
- Storage Management
- Process Management
- Memory Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

(slide modified by R. Doemer, 03/29/10)

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization

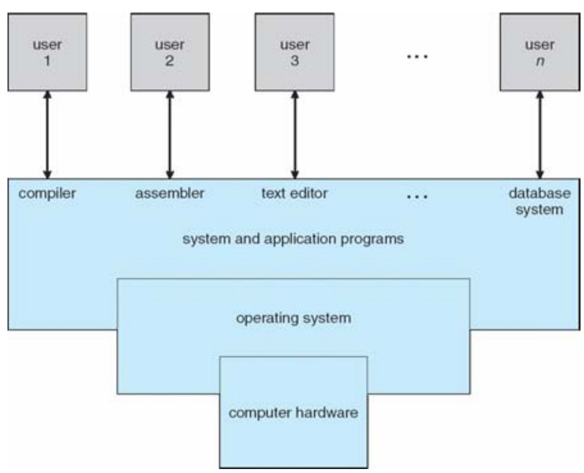
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont)

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation
  - But varies widely
- “The one program running at all times on the computer” is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system)
  - or an application program.

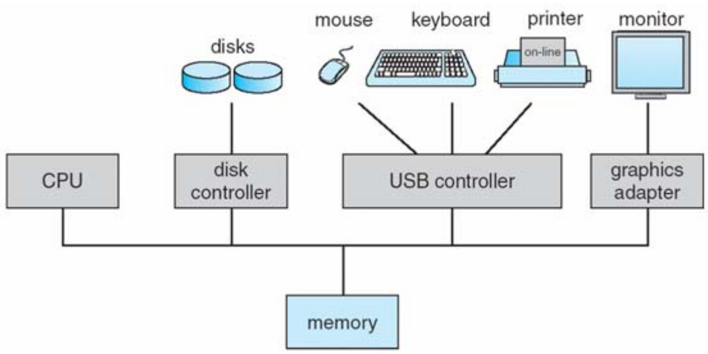
(slide modified by R. Doemer, 03/30/10)

Computer Startup

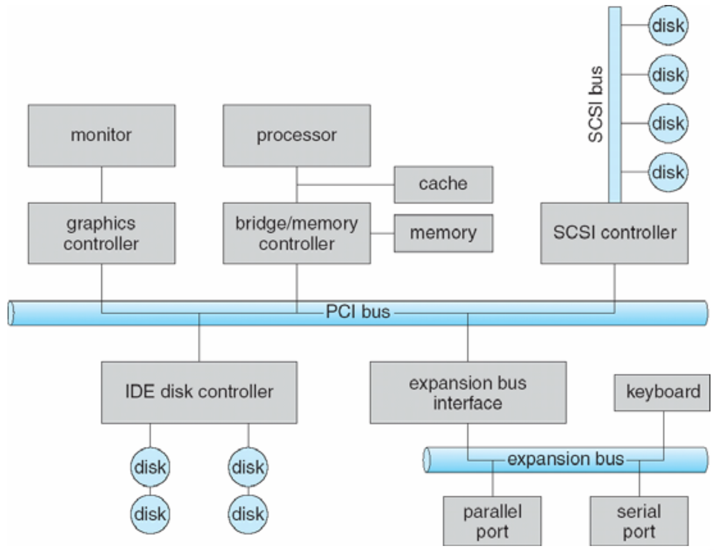
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



A Typical PC Bus Structure



(from Chapter 13, "I/O Systems")

Computer-System Operation

- I/O devices and the CPU execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from controller buffers
- I/O is from the device to the local buffer of controller
- Device controller informs CPU that it has finished its operation by raising an *interrupt*

(slide modified by R. Doemer, 03/30/10)

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine, generally through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- An operating system is **interrupt driven**
- Note:
  A *trap* is a software-generated interrupt caused either by an error or a user request (system-call)

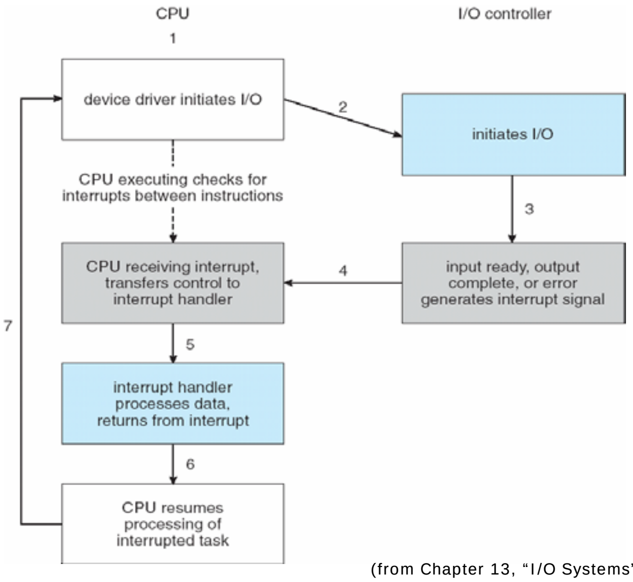
(slide modified by R. Doemer, 03/29/10)

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter on the stack
- The OS then determines which type of interrupt has occurred by one of two schemes:
  - **polling**
  - **vectored interrupt system**
- Separate segments of code determine what action should be taken for each type of interrupt

(slide modified by R. Doemer, 03/29/10)

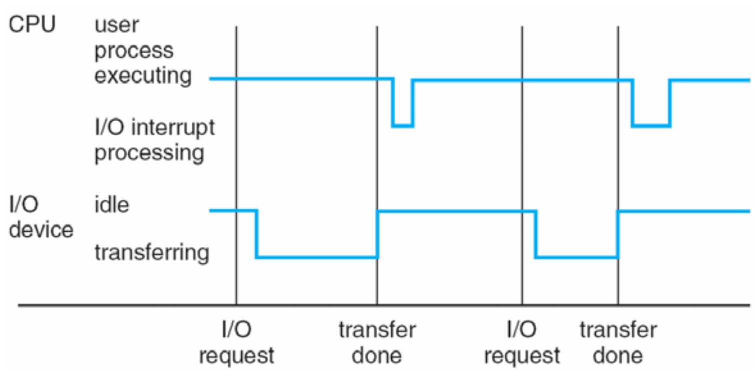
Interrupt-Driven I/O Cycle



(from Chapter 13, "I/O Systems")

Interrupt Timeline

- Interrupt time line for a single process doing output
  - Process issues I/O request
  - I/O device signals "transfer done" via interrupt



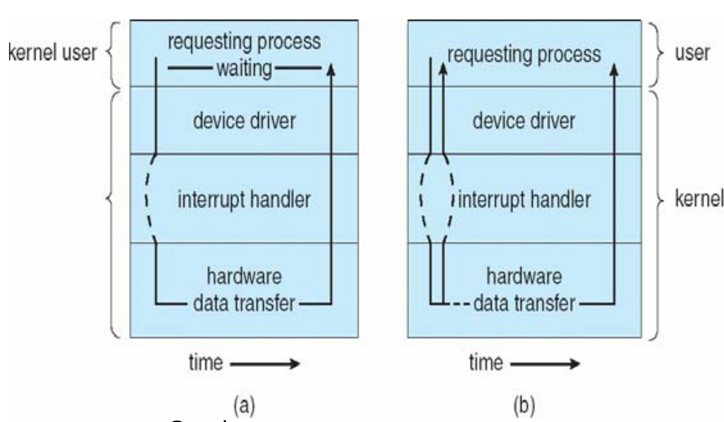
(slide modified by R. Doemer, 03/29/10)

I/O Structure

- *Synchronous* I/O:
  after I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time,
    no simultaneous I/O processing
- *Asynchronous* I/O:
  after I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the operating system
    to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device
    indicating its type, address, and state

(slide modified by R. Doemer, 03/29/10)

Two I/O Methods



Synchronous Asynchronous

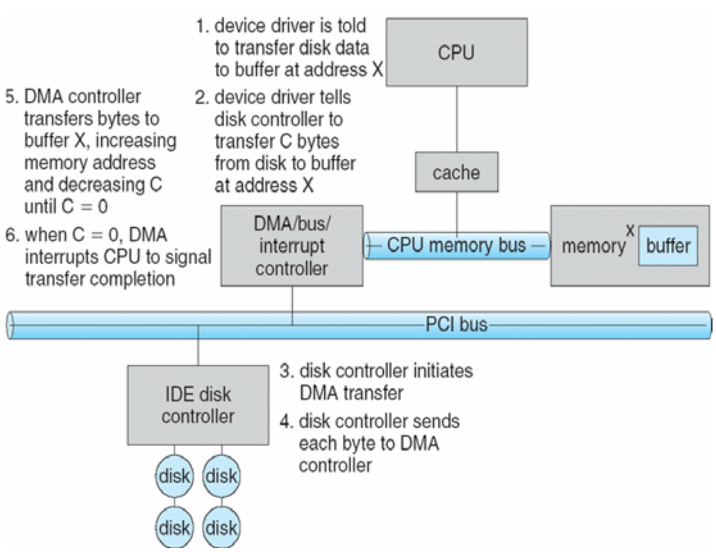
(from Chapter 13, "I/O Systems")

Direct Memory Access Structure

- DMA: Direct Memory Access
- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

(slide modified by R. Doemer, 03/29/10)

Six Step Process to Perform DMA Transfer



(from Chapter 13, "I/O Systems")

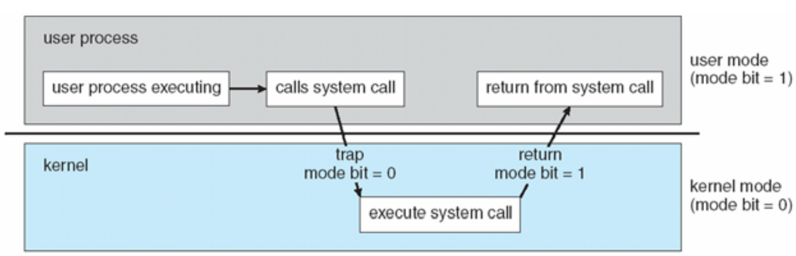
Operating-System Operations

- **Interrupt driven** (by software and hardware)
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Timer** to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Timer decrements counter
  - When counter zero generate an interrupt
  - Set up before for scheduling process to regain control or terminate program that exceeds allotted time

(slide modified by R. Doemer, 03/30/10)

Operating-System Operations

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user



(slide modified by R. Doemer, 01/05/09)

Storage Structure

- CPU registers
  - the only storage capability within the CPU core
- Main memory
  - the only large storage media that the CPU can access directly
- Secondary storage
  - provides large nonvolatile storage capacity
- Magnetic disks
  - Rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer

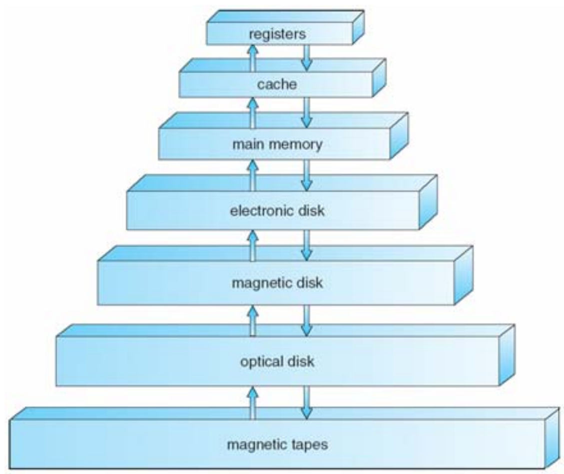
(slide modified by R. Doemer, 03/30/10)

Storage Hierarchy

- Storage systems are organized in a hierarchy
- Storage systems vary in
  - Speed
  - Cost
  - Volatility
- **Caching** is often used between storage systems
  - transparently copying information into faster storage system (e.g. CPU cache holds most-recently used data from main memory)
  - main memory can be viewed as a *cache* for secondary storage

(slide modified by R. Doemer, 03/30/10)

Storage-Device Hierarchy



Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) is checked first to determine if information is there
  - If it is, information is used directly from the cache (fast)
  - If not, data is used from actual storage *and* copied to cache for future use
- Cache is usually smaller (but more costly per byte!) than storage being cached
  - Cache management is an important design problem
  - Factors include *cache size* and *replacement policy*

(slide modified by R. Doemer, 03/30/10)

Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Process Management

- A **process** is a *program in execution.* It is a unit of work within the system.
  - A program is a *passive entity*.
  - A process is an *active entity*.
- A process needs **resources** to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- A single-threaded process has one **program counter** specifying the location of the next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- A multi-threaded process has one program counter per thread
- A typical system has many processes (user, system processes) running concurrently on one or more CPUs
  - Concurrency is implemented by multiplexing the available CPUs among the active processes (and/or threads)

(slide modified by R. Doemer, 03/30/10)

Process Management Activities

The operating system is responsible for

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

(slide modified by R. Doemer, 03/30/10)

Operating System, Outlook…

- **Multiprogramming** needed for efficiency
  - CPU and I/O devices underutilized by single user
  - Multiprogramming organizes jobs (code and data) so that CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)**
  - CPU switches jobs so frequently that users can interact with each job
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨ **process**
  - Several jobs ready to run at the same time ⇨ **CPU scheduling**
  - Not all processes fit in memory ⇨ **swapping**
  - Processes only partially in memory ⇨ **Virtual memory**

(slide modified by R. Doemer, 03/30/10)

End of Chapter 1