Operating System Concepts

Xin-Shun Xu(许 信顺）Ph.D. Professor

Building: 3-B, Room: 210
xuxinshun@sdu.edu.cn

School of Computer Science and Technology, Shandong University

Self-introduction

- Position
  - Professor, School of Computer Science and Technology, Shandong University
  - A leader of **MIMA** Group (Machine Intelligence & Media Analysis URL: http://mima.sdu.edu.cn)
- Research Interest
  - machine learning, media analysis & retrieval, data mining, pattern recognition, artificial intelligence

Slides

http://mima.sdu.edu.cn/Members/~xinshunxu

許信順
Xin-Shun Xu

PhD, Professor

MIMA Group
School of Computer Science and Technology,
Shandong University, Jinan, China

山东大学
SHANDONG UNIVERSITY

MIMA
Machine Intelligence & Media Analysis

Sections:

- Biography
- Research Interest
- Publications
- Projects
- Courses
- Correspondence

Links:

- CFP
- AI Datasets
- IEEE
- ACM

Biography:

Currently I am a professor of School of Computer Science and Technology at Shandong University and a leader of MIMA (Machine Intelligence and Media Analysis) Group.

I received my M.S. degree in Computer Science from Shandong University, China in 2002, and Ph.D. degree in computer science from University of Toyama, Japan in 2005. In the same year, I joined School of Computer Science and Technology of Shandong University as an associate professor. From 2009 to 2012, I am also a postdoctoral fellow in LAMDA Group, led by professor Zhi-Hua Zhou.

Research Interest:

My research interests include: Machine Learning, Information Retrieval, Computational Intelligence, Bioinformatics, Pattern Recognition, Data mining and Combinatorial Optimization. Now I am working on Multi-Instance Learning, Multi-Label Learning, Semi-supervised Learning, Ensemble Learning and Media Analysis & Retrieval.

📖 Recommended Reading

- **Operating System Concepts(7th), Abraham Silberschatz etc., Wiley, 2007**
- **Applied Operating System Concepts**
- **Modern Operating Systems(3rd) Andrew S. Tanenbaum, Prentice Hall, 2008**
- **Operating Systems：Internals and Design Principles(6th) William Stallings**，2008
- 计算机操作系统(第3版)，汤子瀛 *etc.*，西安电子科技大学出版社
- 计算机操作系统教程 张尧学 史美林 清华大学出版社
- 计算机操作系统教程 周长林 左万历 高等教育出版社

Remarks

- Lectures and books are important, but not enough.
- You can get more materials from library/www.
- You should review what have been taught with more hours than the class hours/week.
- You should write code as much as possible.

No Pain, No Gain!!!

Lessons & Contents

- Lessons 56 hours
- Laboratory 16 hours
- OS Design (lecture) 16 hours
- OS Design (project) 32hours
- Chapters Part 1-5(Chapter 1-15)

How to get a mark

- Lessons, assignments, projects, readings
- Final Exam
- Scores = Exam + Assignments + Attendance + Projects

The objectives to Learn OS

- How to use Windows or Linux?
- Basic concepts, structure,
- Principles, design methods and implementation

What can you do after learning

- Design new operating system
- Modify operating system
- Beneficial to programming
- Some algorithms
- How to buy/choose an operating system for yourself, your friend or your corp.

Chapter 1 Introduction

Contents

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments

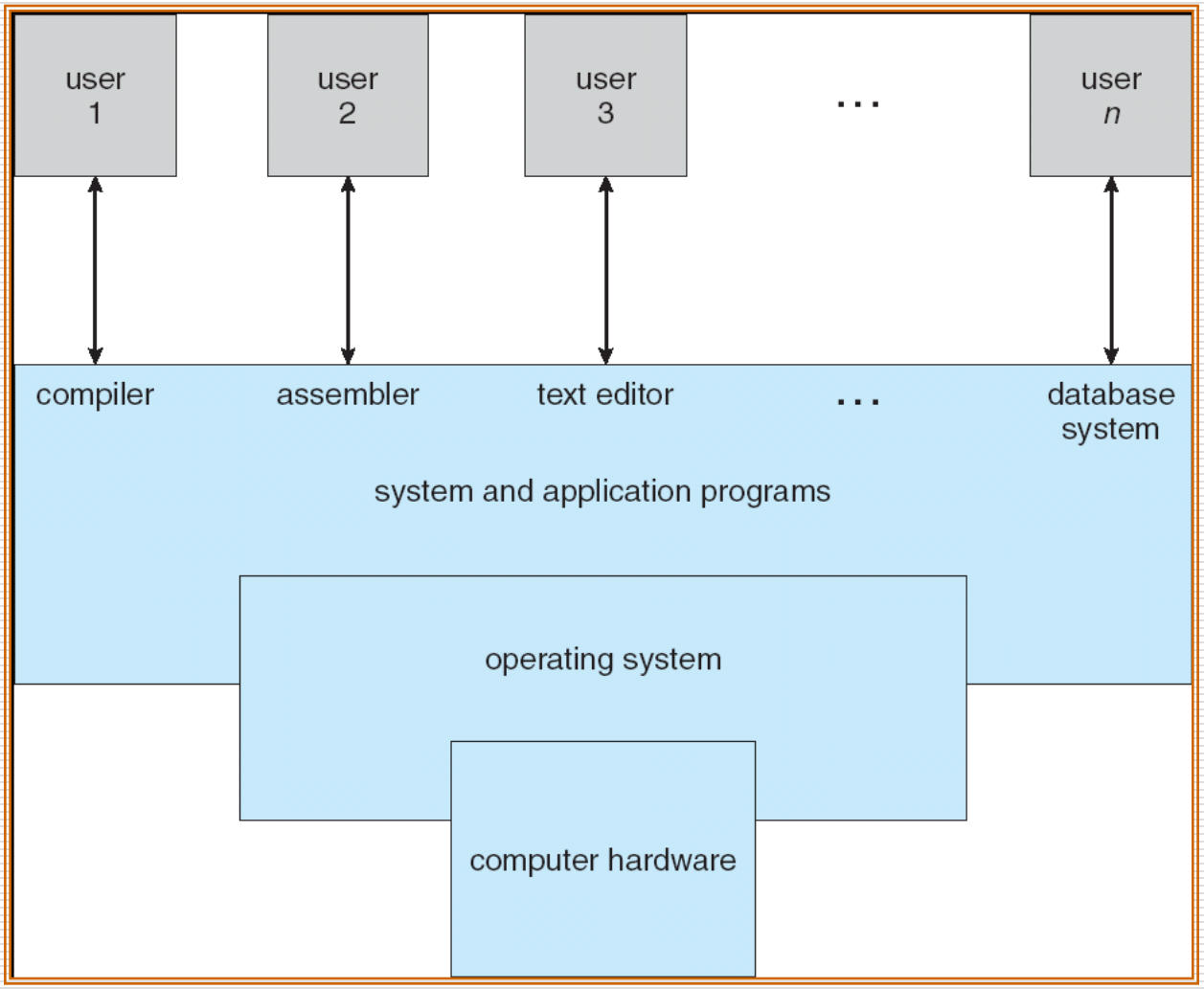
Objectives

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

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

Computer System Structure



What is an Operating System?

- A program that manages the computer hardwares and softwares, provides a basis for application programs and acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Make the system to be efficient
  - Make the system to be convenient

What Operating System Do-An Example

```
#include <stdio.h>
 int main(int argc, char *argv[])
 {
        printf("Hello World");
        return 0;
 }
```

What Operating System Do-An Example

- User tells OS to execute Hello program
- OS finds this program, and check it.
- File system searches the section on the disk
- Create a new child process to execute the program
- Allocate memory for this process
- OS sets the CPU context, and jump to the beginning of this program
- Run the first instruction
- Read more pages
- OS finds the device that the string to be sent
- The device is controlled by a process

What Operating System Do-An Example

- OS sends the string to the process.
- The process informs the window system
- Window system confirms that this is a legal operation, and translates the string into pixels
- Write the pixels into buffer
- Hardware translates the pixels into signals, and sends them
- Finally, you get “Hello World”.

Research on OS from different views

- User view
  - From this view, the OS is designed mostly for ease of use
- System view
  - OS is a resource allocator
  - A control program
    - Manage the execution of user programs to prevent errors and improper use of the computer

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

History of OS

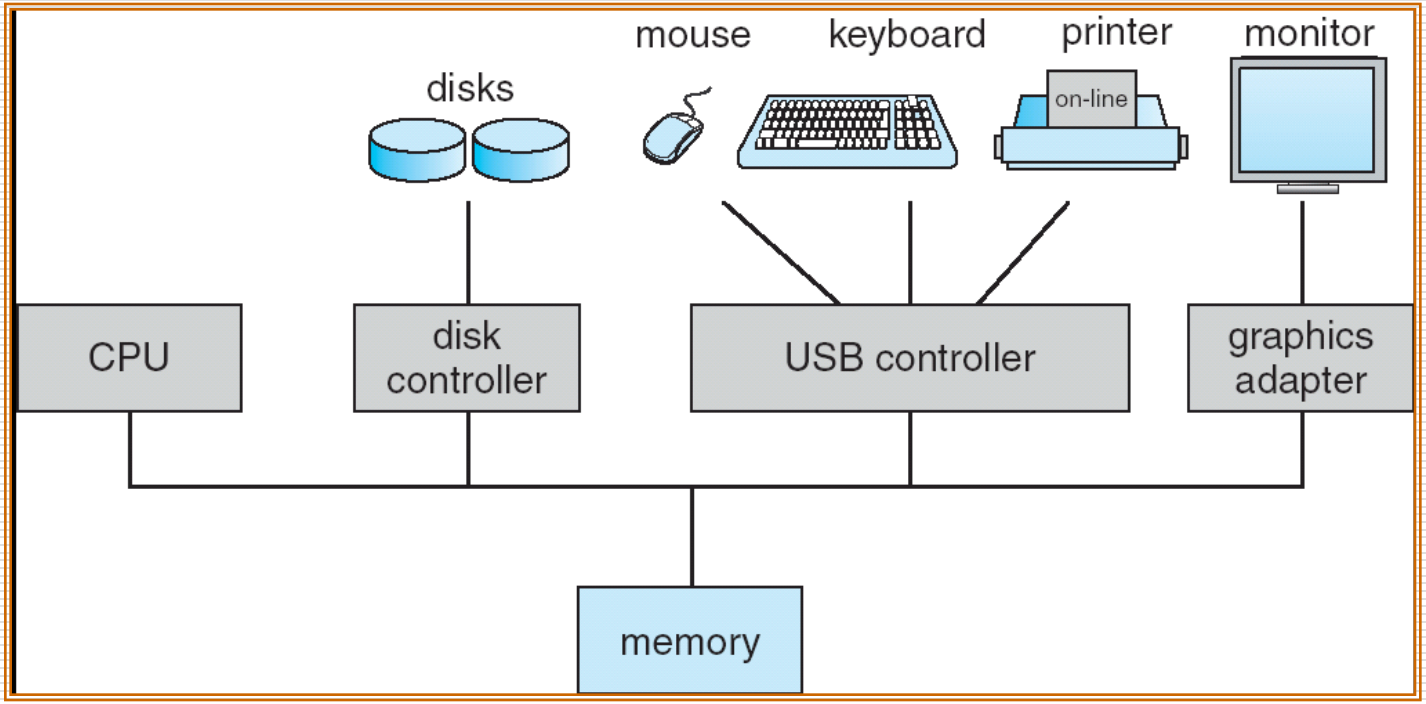
- With the development of computer technology, OS becomes more powerful.
  - To improve the efficiency of resources
  - To become more comfortable for users
  - Hardwares are developing
  - Architecture changes

History of OS

- Nearly no OS
- Batch processing OS
- Multi-program System
- Time-Sharing System
- Desktop OS
- Parallel System
- Clustered System

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



Computer System Organization

- Computer-system operation
  - 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 Operation

- I/O devices and the CPU can 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 local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

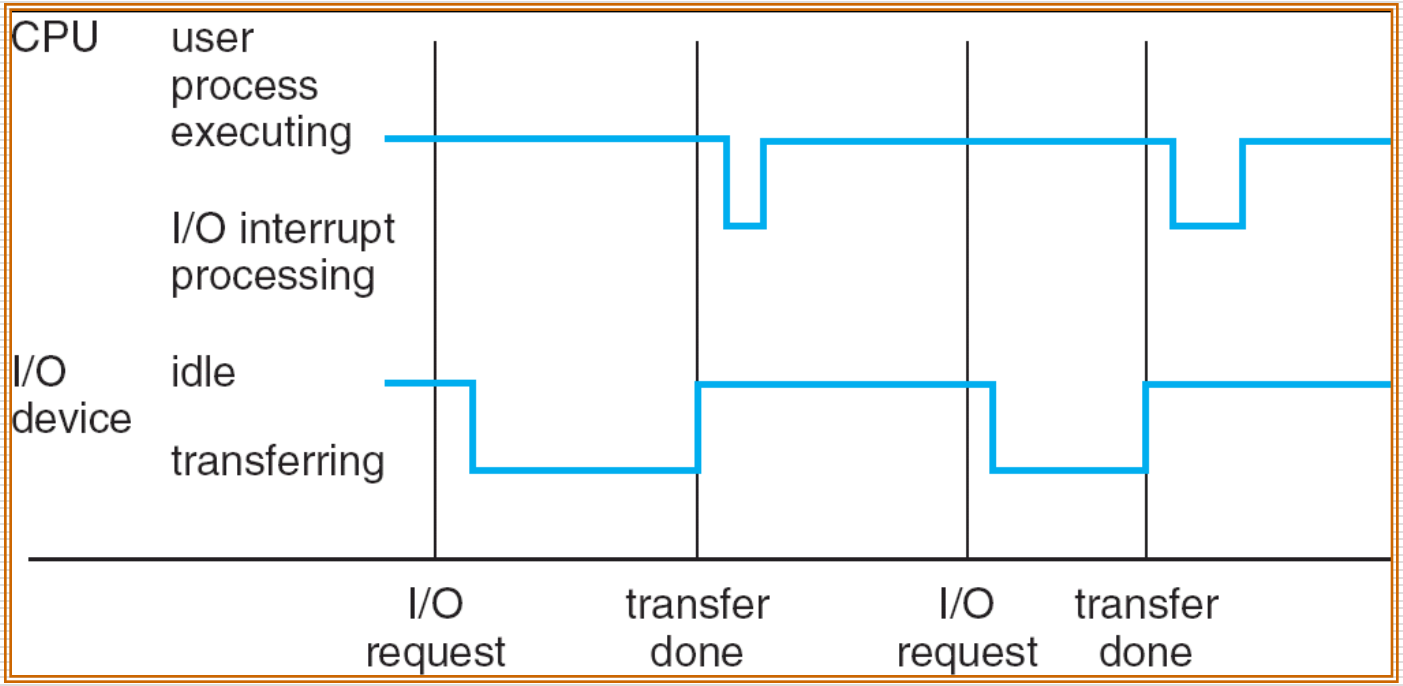
Common Functions of Interrupts

- 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*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- Software may trigger an interrupt by executing a special operation called a system call (also called a monitor call)
- An operating system is *interrupt* driven.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - *polling*
  - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline



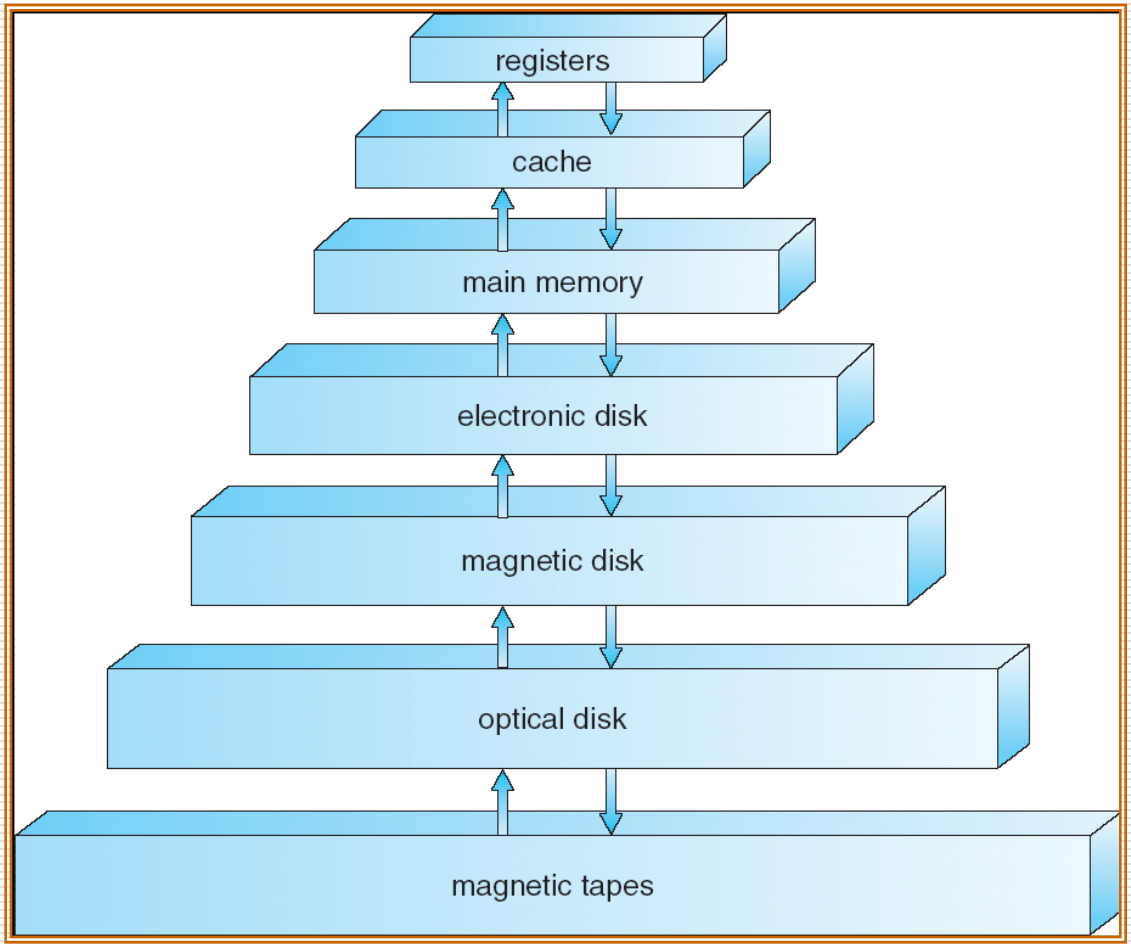
Storage Structure

- Main memory – only large storage media that the CPU can access directly.
  - Main Memory is usually too small to store all programs and data permanently
  - Main memory is volatile storage device that loses its contents when power is turned off or otherwise lost
- Secondary storage – extension of main memory that 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.
- Others
  - Register, Cache, CDROM, DVD, Floppy disk, Flash memory

Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility

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

- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.
- 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) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

The place to use caching

- Between processor and main memory
- Between processor and peripherals
- Between device and device

Can it really work?

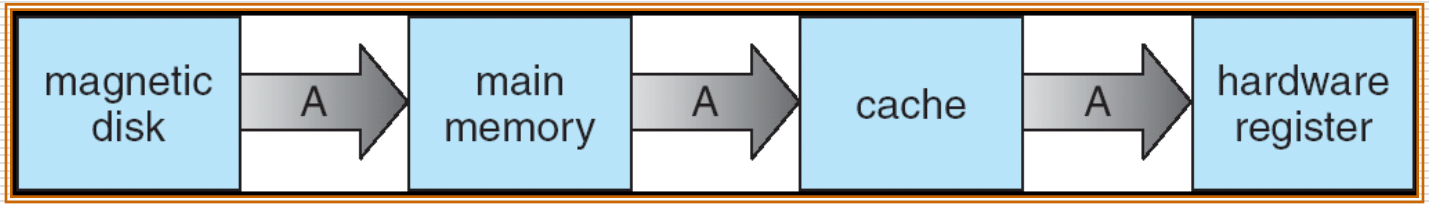
- Suppose there is a 2-level storage system
- In level 1, access time is 0.1μs
- In level 2, access time is 1μs
- If data are in level 1, they can be read directly
- If data are in level 2, then they must firstly be read into level 1, and read them from level 1.
- Suppose hit rate is 95%, then the average access time is:

$$(0.95)(0.1\mu s)+(0.05)(0.1\mu s+0.1\mu s+1\mu s)=0.155\mu s$$

- It really works!!!

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy

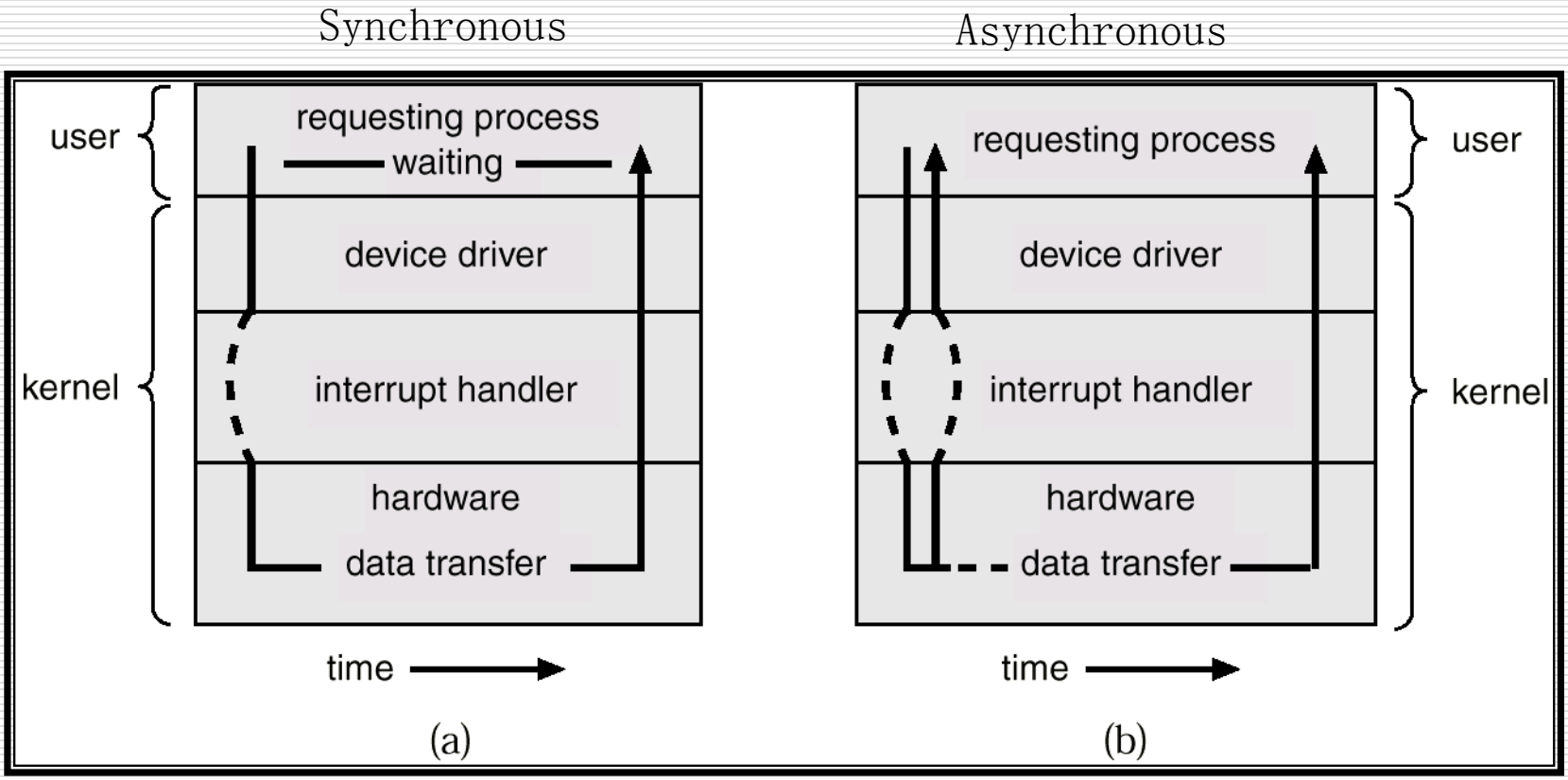


- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

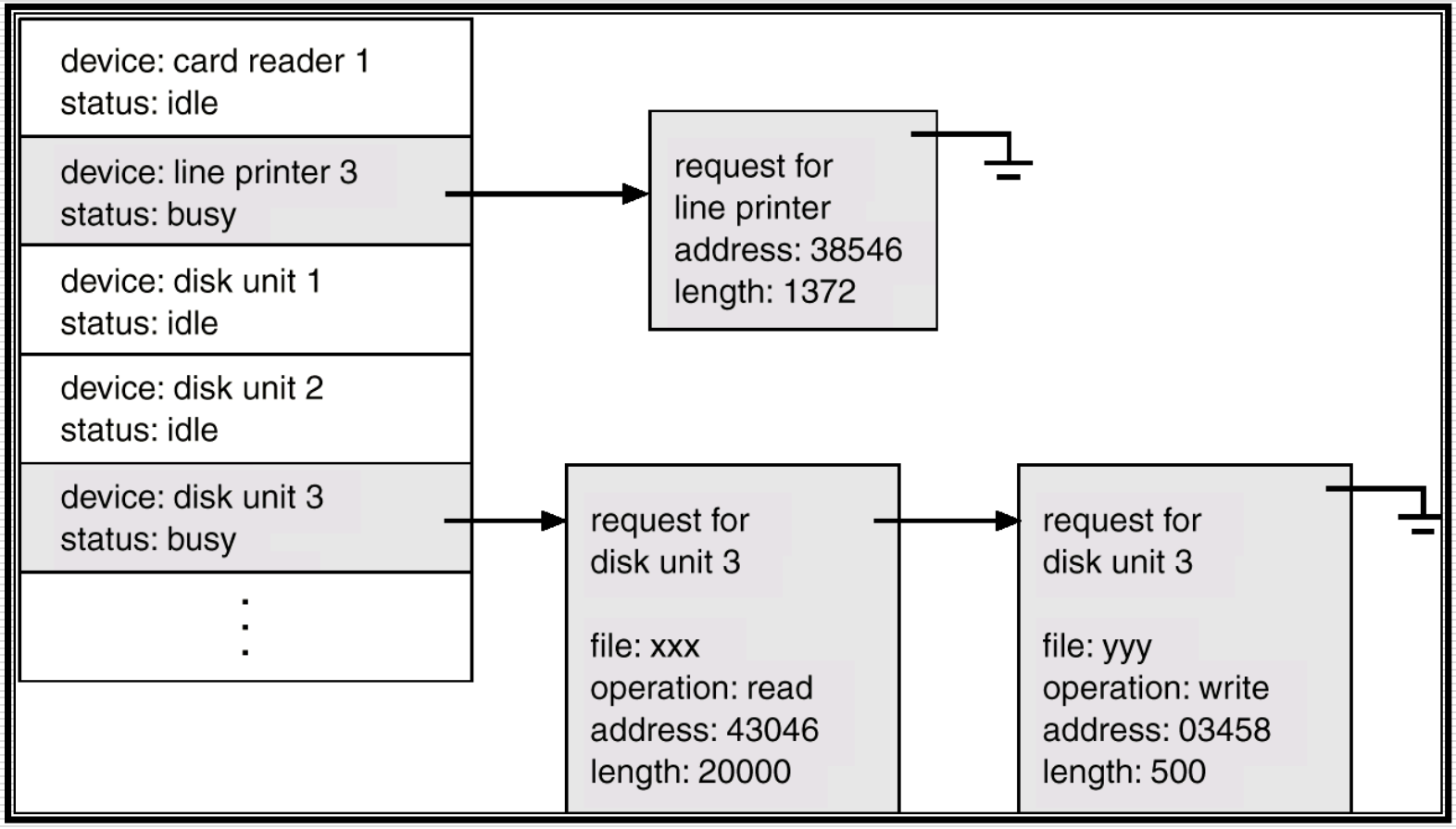
I/O Structure

- (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.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Two I/O Methods



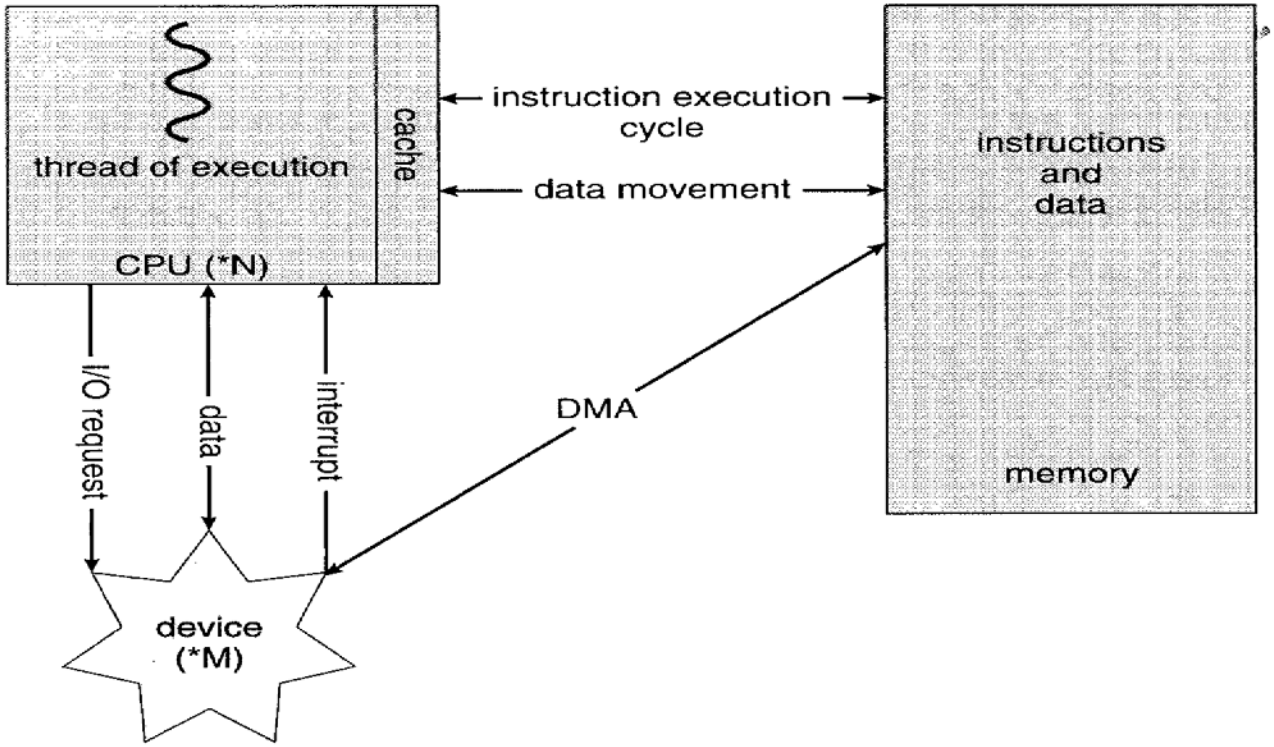
Device-Status Table



Direct Memory Access Structure

- 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.

How a modern computer system works



Computer System Architecture

- Single-processor Systems
- Multiprocessor Systems
- Clustered Systems

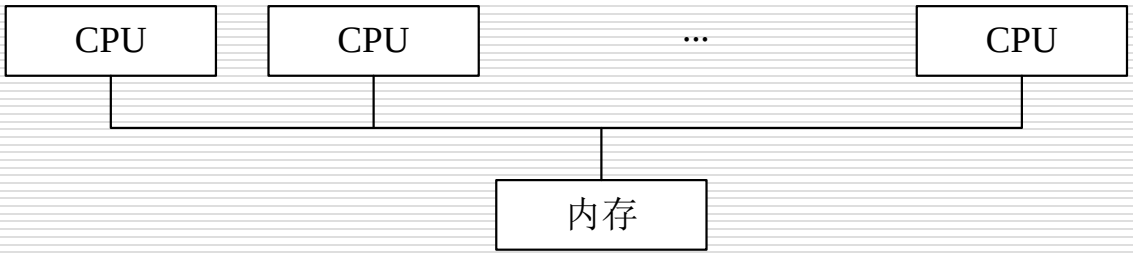
Multiprocessor System

- Increased throughput
- Economy of Scale
- Increased reliability
  - Graceful degradation
  - Fault tolerance

Asymmetric multiprocessing

Symmetric multiprocessing

Blade servers



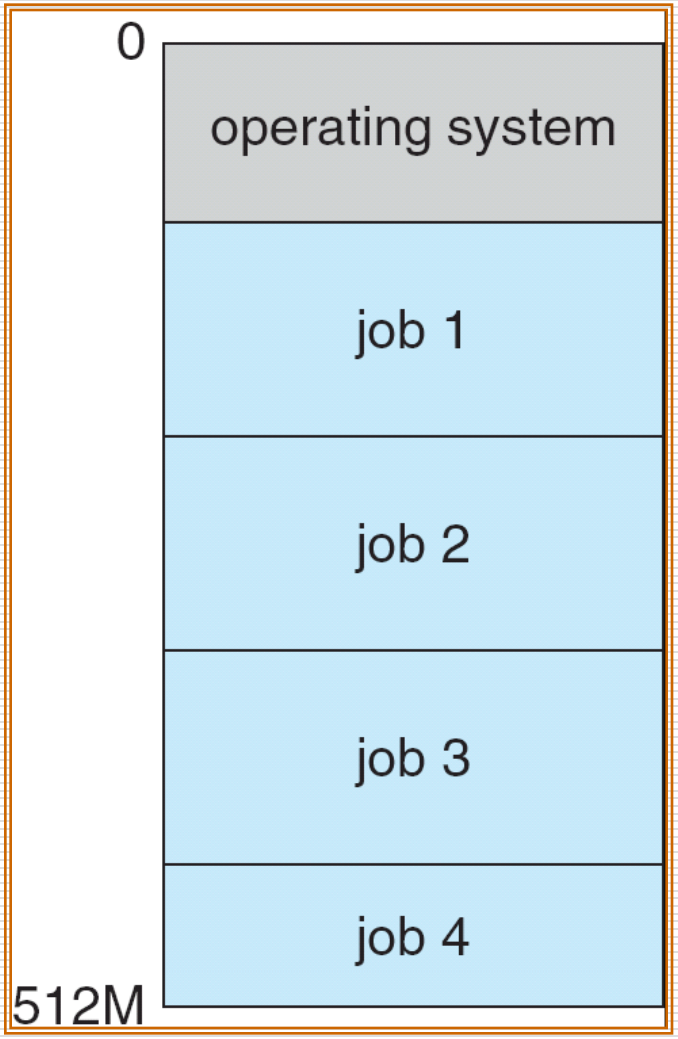
Clustered Systems

- The generally accepted definition is that clustered computers share strorage and are closely linked via a local-area-network(LAN) or a faster interconnect such as infiniBand
- Clustering is usually used to provide high-availability service
- Asymmetric clustering
- Symmetric clustering

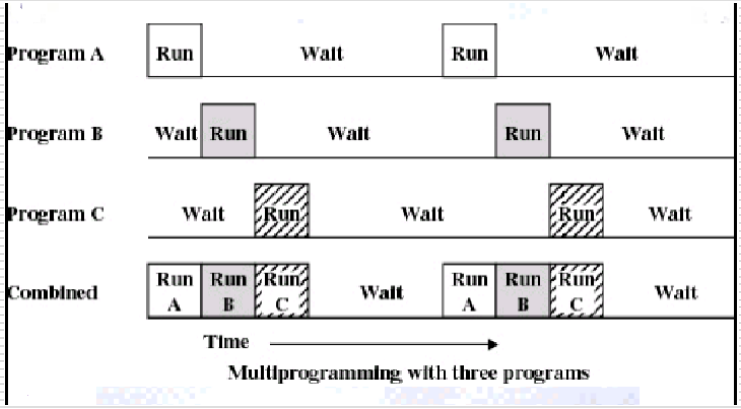
Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so 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

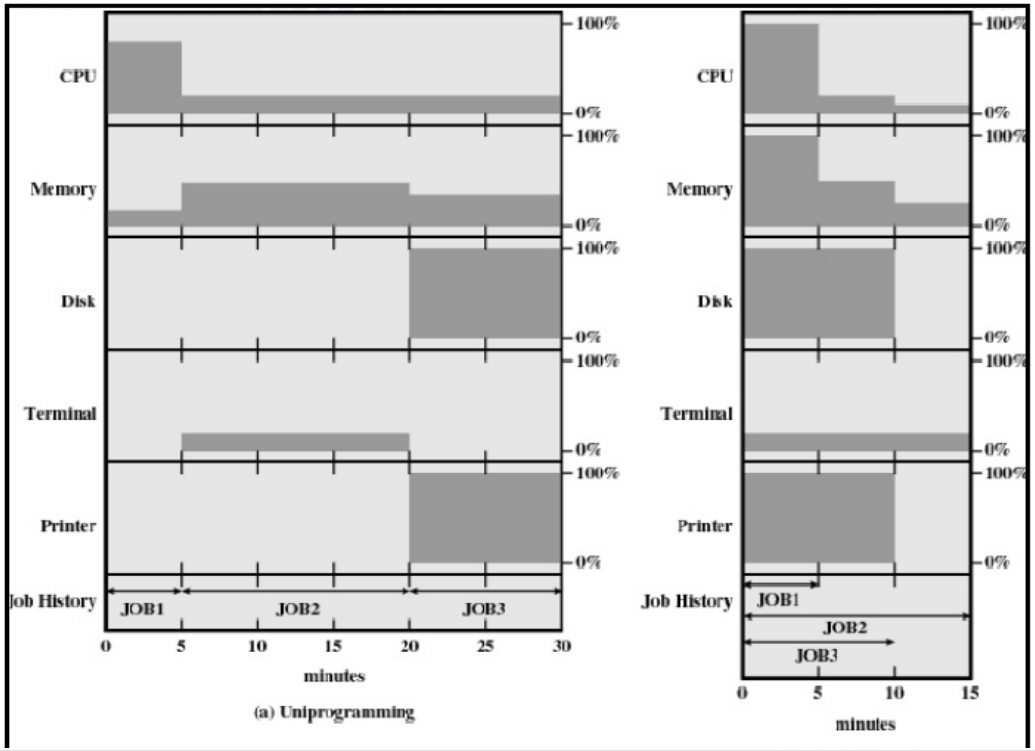
Memory Layout for Multiprogrammed System



Multiprogramming can improve efficiency?



	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min.	15 min.	10 min.
Memory required	50K	100 K	80 K
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes



Operating System Structure

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory □**process**
  - If several jobs ready to run at the same time □ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory
  - **Physical memory**

Operating System Structure

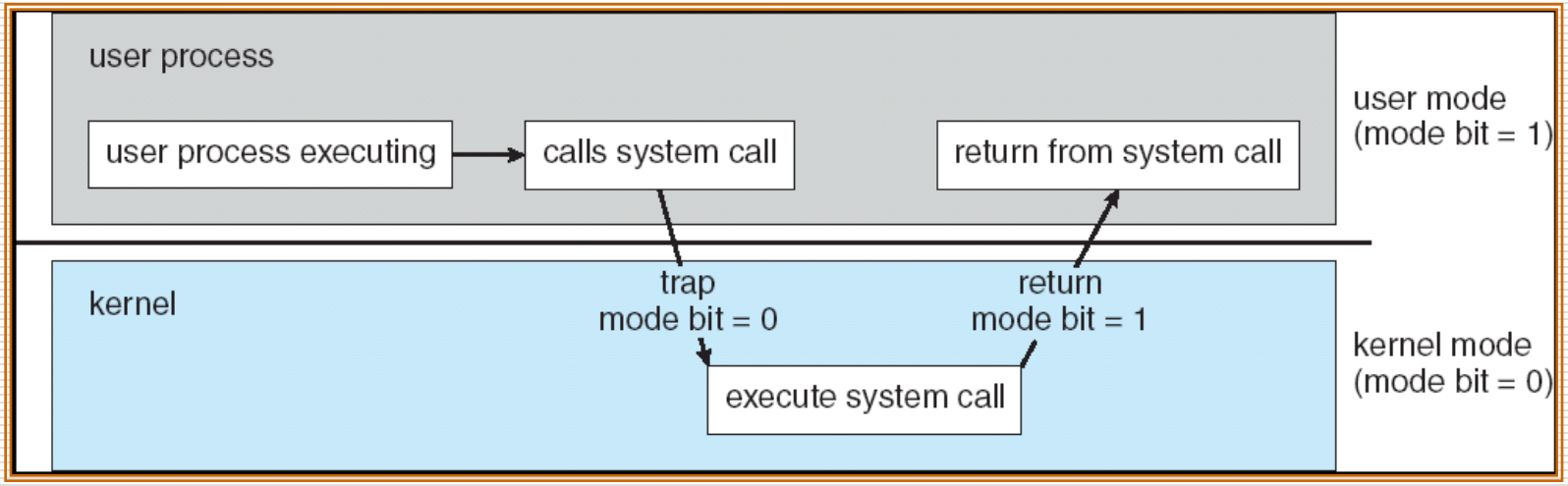
- Job pool
  - It is used to keep the jobs on the disk
  - This pool consists of all processes residing on disk awaiting allocation of main memory.
- Job scheduling
  - It chooses the jobs from job pool to bring them into memory

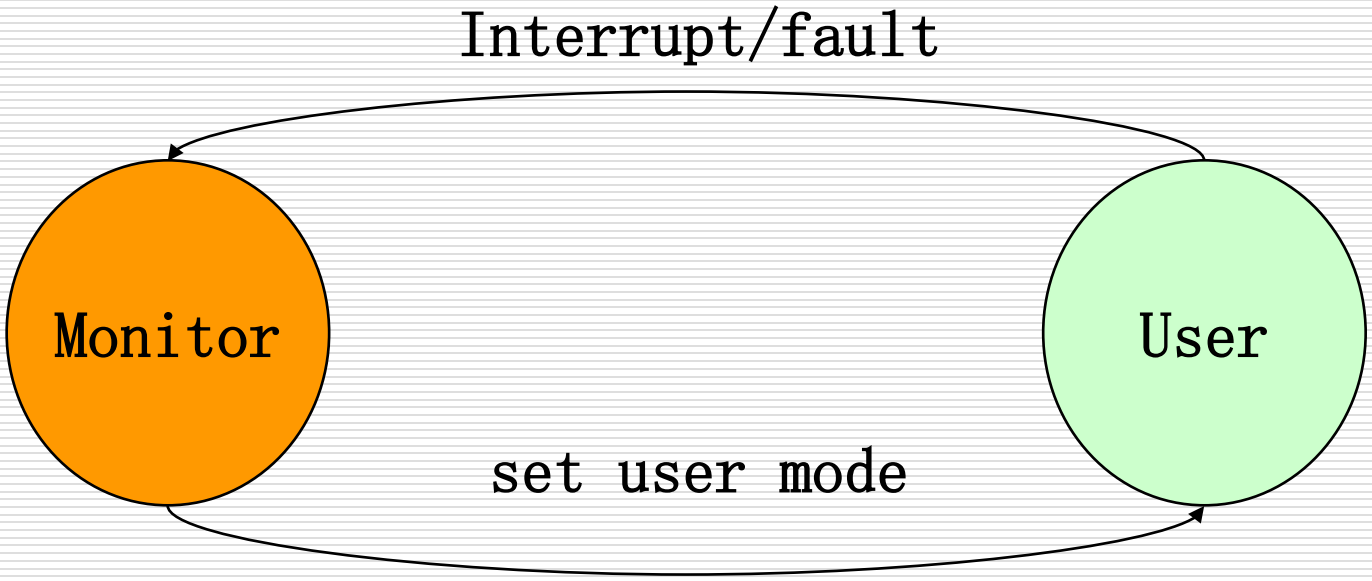
Operating-System Operations

- Interrupt driven by 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

Transition from User to Kernel Mode

- **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 mode





Privileged instructions can be issued only in monitor mode.

Who can modify the mode bit

- Can kernel change the mode bit?
  - Yes. Kernel is completely trusted.
- Can user process change the mode bit?
  - Not **directly**
  - User programs need to invoke the kernel

When to transition from user to kernel?

1. Exceptions (interrupts, traps)
   - Access something out of your valid address space
     - (e.g. segmentation fault)
   - Disk I/O finishes, causes interrupt
   - Timer pre-emption, causes interrupt
   - Page faults
2. **System calls**
   - Similar in purpose to a function call

Example system calls

- Process management
  - Fork/exec (start a new process), exit, getpid
- Signals
  - Kill (send a signal), sigaction (set up handler)
- Memory management
  - Brk (extend the valid address space), mmap
- File I/O
  - Open, close, read, write
- Network I/O
  - Accept, send, receive
- System management
  - Reboot

Timer

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Process Management

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

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- 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

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

Storage Management

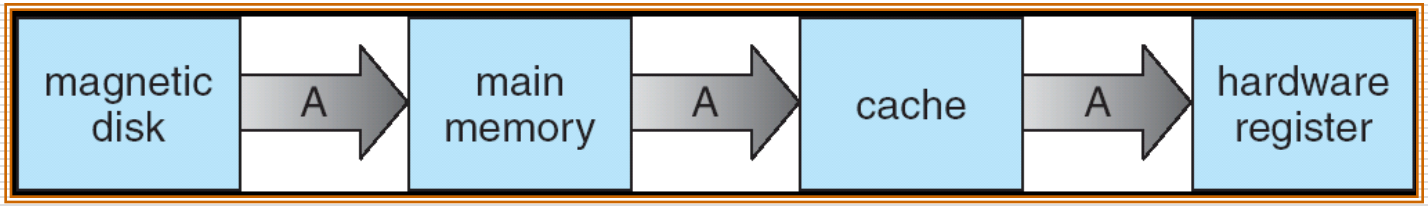
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential 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

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time.
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Cache management

- Careful selection of the cache size and of a replacement policy can greatly increase performance.
- Problems in multitasking environment



I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (g**roup ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Distributed Systems

- A distributed system is a collection of physically separate, that are networked to provide the users with access to the various resources that the system retains.
- Local-area network(LAN)
- Metropolitan-area network(MAN)
- Small-area network
- A network operating system is an operating system that provides features such as file sharing across the network and that includes a communication scheme that allows different processes on different computers to exchange messages.

Special purpose systems

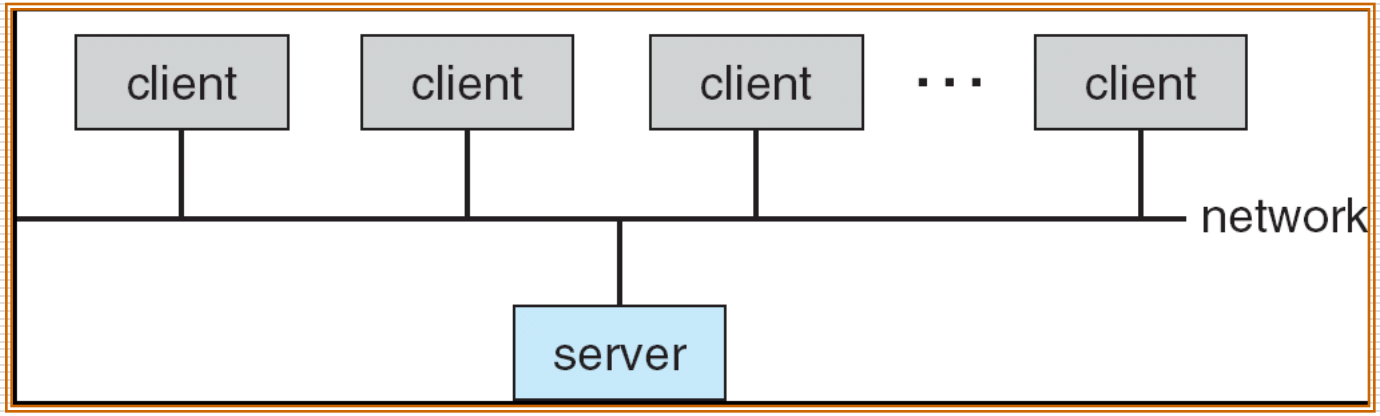
- Real-time embedded systems
- Multimedia systems
- Handheld systems

Computing Environments

- Traditional computing
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Computing Environments (Cont.)

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via *discovery protocol*
  - Examples include *Napster* and *Gnutella*

Web-Based Computing

- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

Assignments

- 1.10, 1.11, 1.13

End of Chapter 1

Any Question?