

# Operating Systems(CI45/CY45)

# Course Outcomes

- Describe the basic structure and functionality of Operating System.
- Apply different scheduling algorithms to schedule multiple tasks for execution by the processor and compare their performance trade-offs.
- Describe the need for controlled access to computing resources by co-operative processes.
- Apply deadlock detection and prevention algorithms to solve the given problem
- Illustrate Primary and secondary memory management strategies
- Illustrate operating system principles for achieving protection and security.

# Teaching Methodology:

- Black board Teaching/PowerPoint presentation
- Industry driven case study

# Assessment Methods

- GATE based aptitude test - 10 Marks
- Programming Assignments 10nmarks
- Two internals, 30Marks each will be conducted and the Average of best of both will be taken.
- Rubrics for evaluating programming assignments.
- Final examination, of 100Marks will be conducted and will be evaluated for 50Mark

# Course Content

## UNIT-I

**Introduction:** What operating systems do; Computer System organization; Computer System architecture; Operating System structure; Operating System operations; Protection and security; Distributed system; Special purpose systems; Virtual machines.

**Operating system structures:** operating system services, user operating system Interface, System calls, Types of system calls, Operating system structure, System boot

## **UNIT-II**

**Process Management:** Basic concept; Process scheduling; Operations on processes; Inter process Communication

**Threads:** Overview; Multithreading models;

**Process scheduling:** Basic concepts, Scheduling criteria, scheduling algorithms, multiple processor scheduling, Algorithm evaluation.

## **UNIT-III**

**Process Synchronization:** Synchronization, The Critical section problem; Peterson's solution; Synchronization hardware; Semaphores; Classical problems of synchronization; Monitors.

**Deadlocks:** System model; Deadlock characterization; Methods for handling deadlocks; Deadlock prevention; Deadlock avoidance; Deadlock detection and recovery from deadlock.

## **UNIT-IV**

**Memory Management Strategies:** Background; Swapping; Contiguous memory allocation; Paging; Structure of page table; Segmentation.

**Virtual Memory Management:** Background; Demand paging; Copy-on write; Page replacement; Allocation of frames; Thrashing



## **UNIT-V**

**File System:** File concept; Access methods; Directory structure; File system mounting; file sharing; Protection.

**Secondary Storage Structures:** Disk scheduling; FCFS Scheduling, SSTF scheduling, SCAN, C-SCAN scheduling, Look Scheduling.

**System Protection:** Goals of protection, Principles of protection, Domain of protection, Access matrix, Implementation of access matrix, Access control, Revocation of access rights, Capability-Based systems.

# REFERENCES

## **Text Book:**

- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne Operating System Principles, 8th edition Wiley- India, 2011

## **Reference Books:**

- D.M Dhamdhere Operating systems - A concept based Approach, 2nd Edition, Tata McGraw-Hill, 2002
- Harvey M Deital Operating systems, 3rd Edition, Addison Wesley, 1990.
- Operating Systems: Principles and Practice (2nd Edition), by Thomas Anderson and Michael Dahlin.

# UNIT-1

## Chapter 1: Introduction



**What is an Operating System?**

**Any Answers?????**

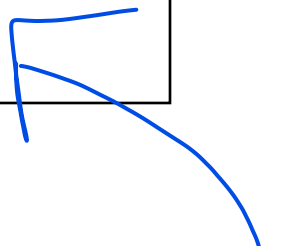
# Definition

- An Operating system is a program/Software that manages the computer hardware, provides a basis for application programs and acts as an intermediary between the computer user 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

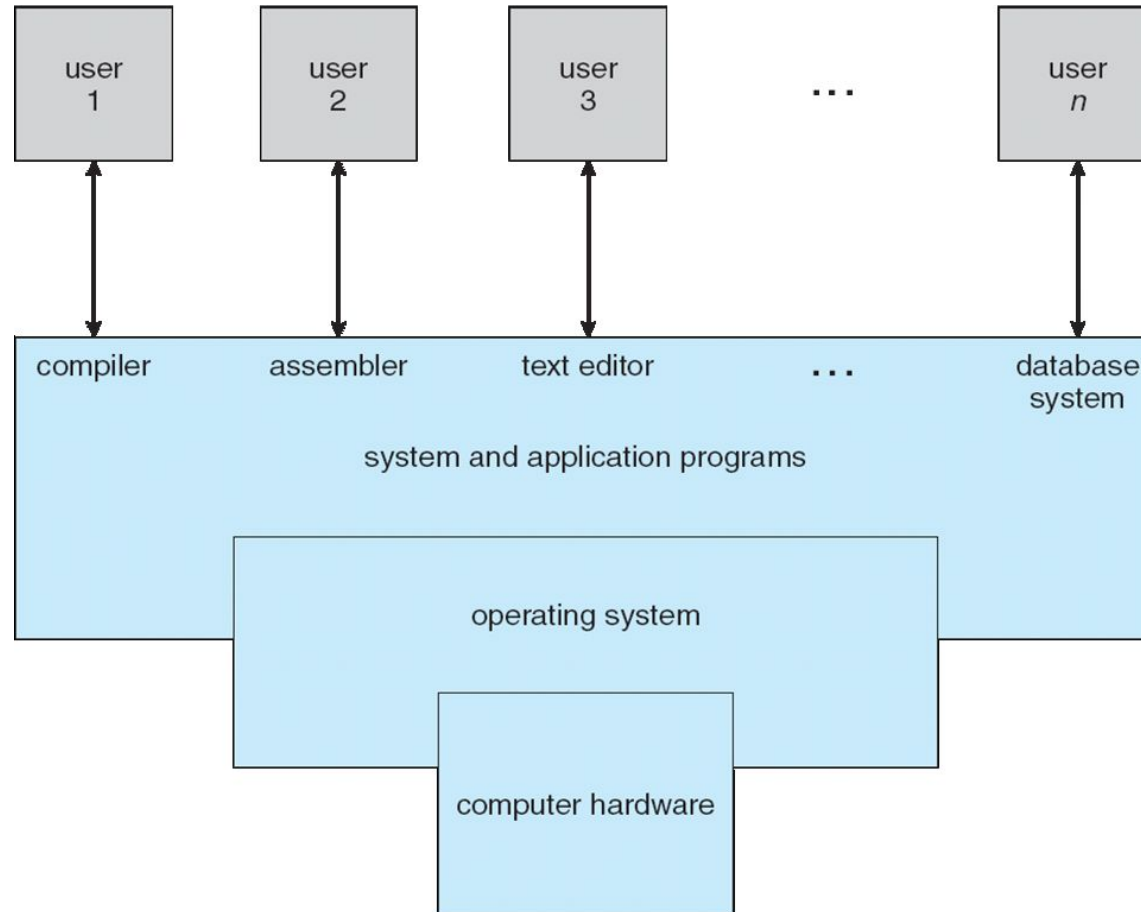
# Computer System Structure

(In order to Understand the functionality of operating system its necessary to have insight on the Structure of computer system)

- Computer system can be divided into four components:
  - **Hardware** – provides basic computing resources
    - 4 CPU, memory, I/O devices
  - **Operating system**
    - 4 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
    - 4 Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - 4 People, machines, other computers



# Four Components of a Computer System



Handwritten notes in blue ink:

- A large "V" shape.
- The text "AS" (likely referring to Assembler).
- The text "Reqs" (likely referring to Requirements).
- The text "OS" (Operating System).
- The text "N/W" (Network).

# What Operating Systems Do

To understand more fully the operating systems role, Let us explore operating systems from two viewpoints: that of the user and that of the system.

## User view:

- In case of single user system, the operating system is designed mostly for ease of use with some attention paid to performance and none paid to resource utilization.
- In case of mainframes or minicomputers, users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization to assure that all available CPU time, memory, and I/O are used efficiently.
- In still other cases, users sit at workstations connected to networks of other workstations and These users have dedicated resources at their disposal, but they also share resources such as networking and servers-file, compute, and print servers. Therefore, their operating system is designed to compromise between individual usability and resource utilization.
- In case of handheld computers, operating systems are designed mostly for individual usability, but performance per unit of battery life is important as well.
- Some computers have little or no user view. For example, embedded computers in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.



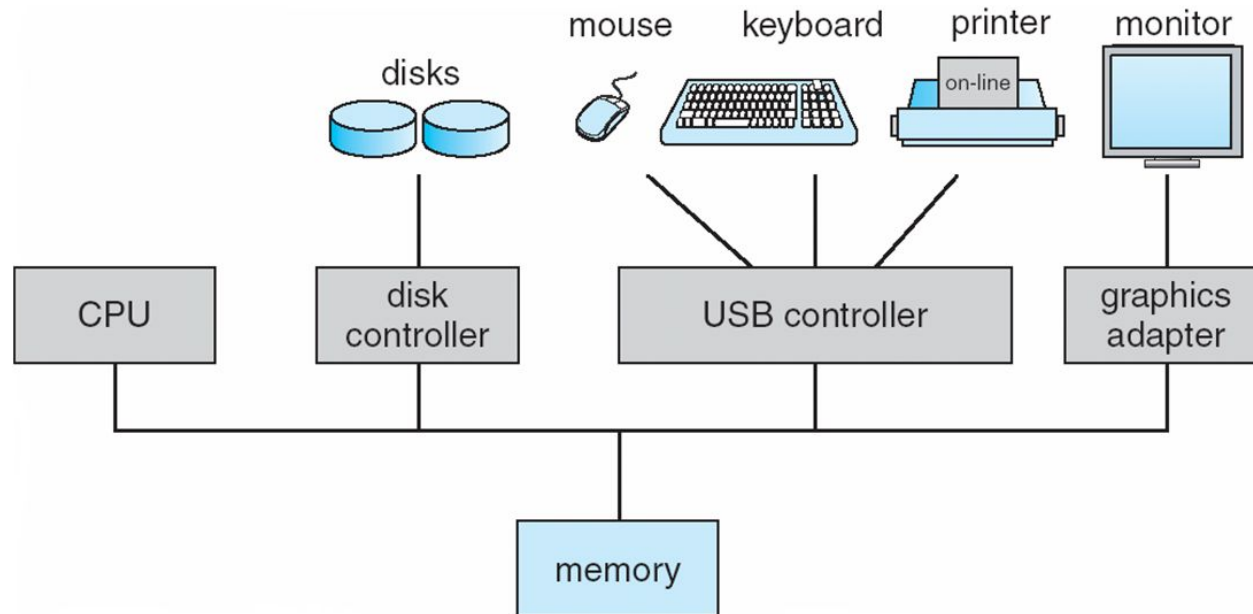
# System view

From computer's point of view, the operating system is the program most intimately involved with the hardware. A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the resource allocator/ manager of these resources.

- 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.

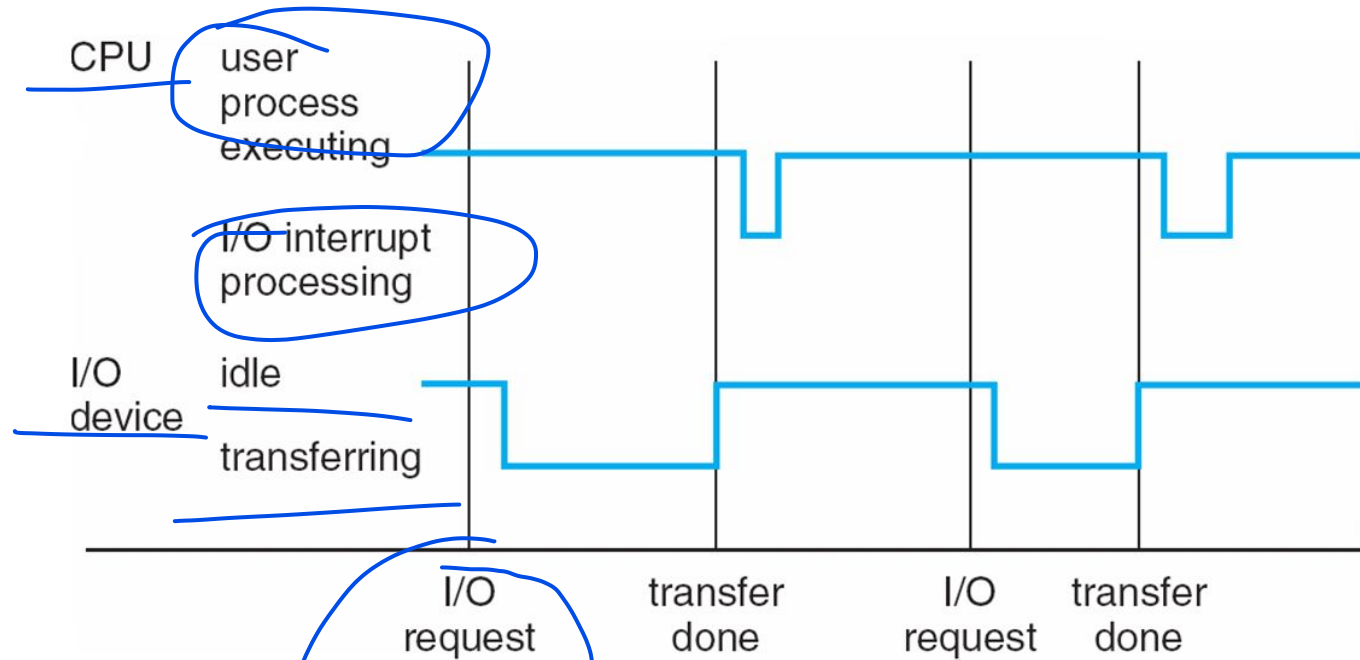
# Computer System Organization

- Computer-system operation
- A modern general-purpose computer system consists of :
  - One or more CPUs, device controllers connected through common bus providing access to shared memory.
  - The CPU and the device controllers can execute concurrently, competing for memory cycles.



- For a computer to start running, when it is powered up or rebooted-it needs to have an initial program to run. This initial program is called bootstrap program.
- Typically, it is stored in read-only memory(ROM) or electrically erasable programmable read-only memory(EEPROM) known by the general term firmware within the computer hardware.
- It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program must know how to load the operating system and how to start executing that system. To accomplish this goal, the bootstrap program must locate and load into memory the operating system kernel.
- The operating system then starts executing the first process, such as "init," and waits for some event to occur. The occurrence of an event is usually signaled by an interrupt from either the hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus. Software may trigger an interrupt executing a special operation called a system call.
- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to Interrupt service routine. The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.

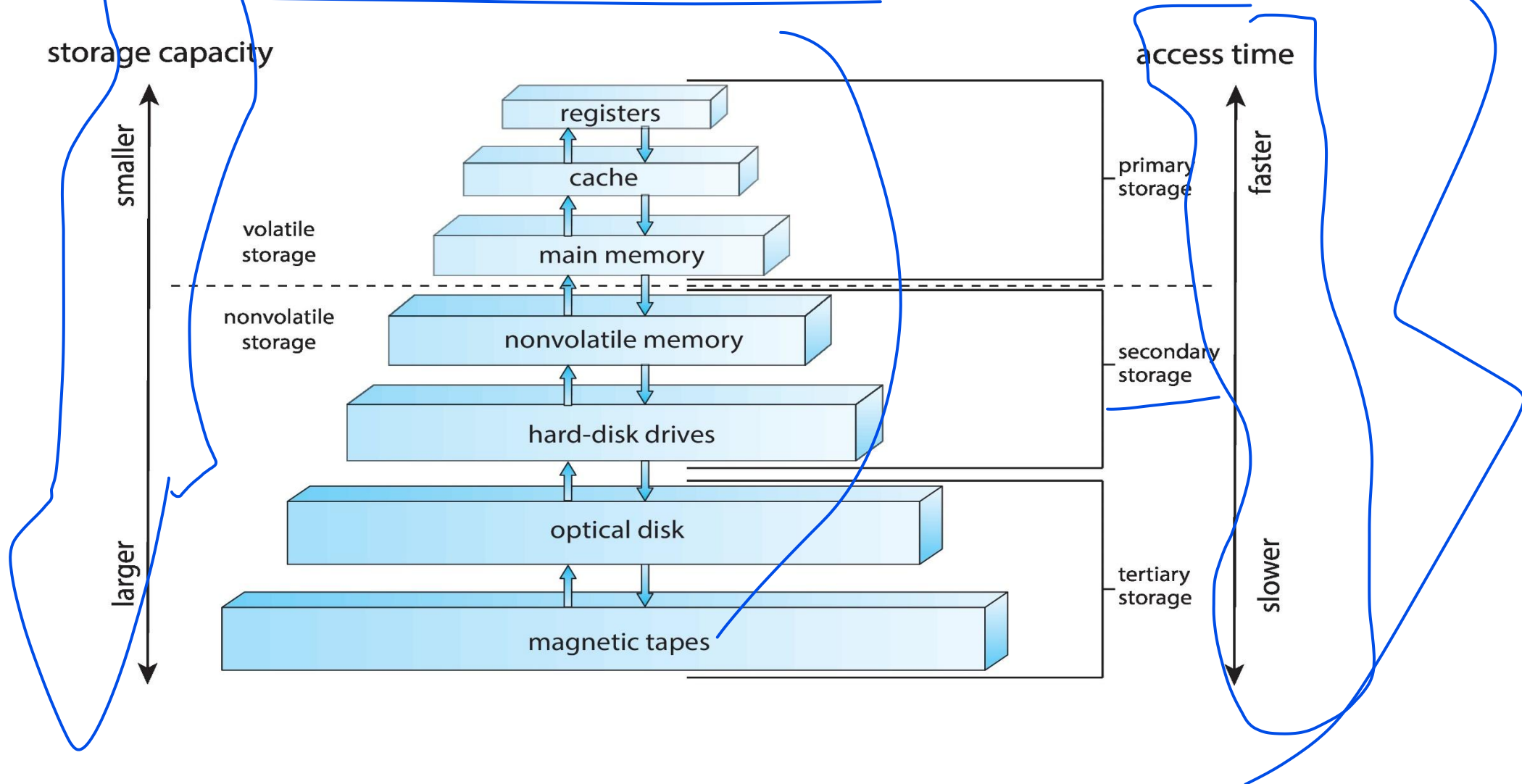
# Interrupt Timeline



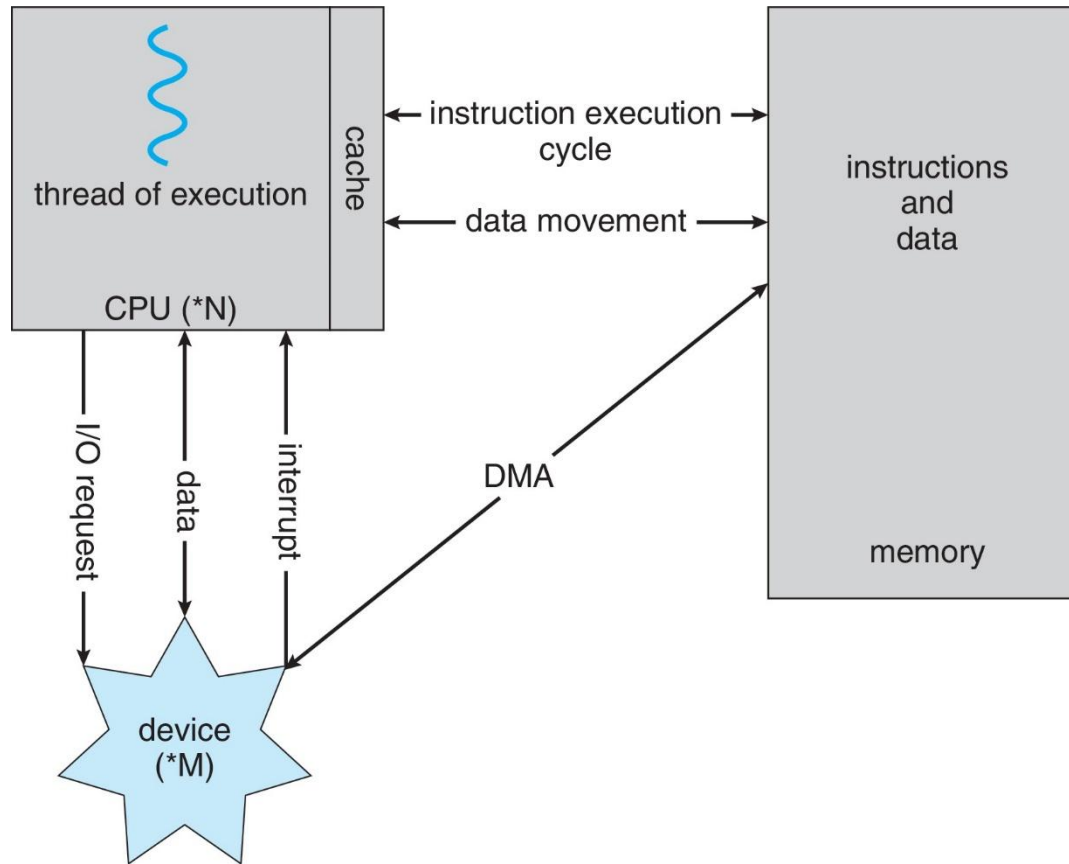
# Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - Random access(any byte of memory can be accessed without touching the preceding bytes)
  - Typically volatile in nature.
  - Main memory is implemented in a semiconductor technology called Dynamic Random-access Memory (DRAM).
  - Read only memory (ROM) - cannot be changed and hence ,only static programs are stored there.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity
- Hard Disk Drives (HDD) – 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
- Becoming more popular as capacity and performance increases, price drops.

# Storage-Device Hierarchy



# How a Modern Computer Works



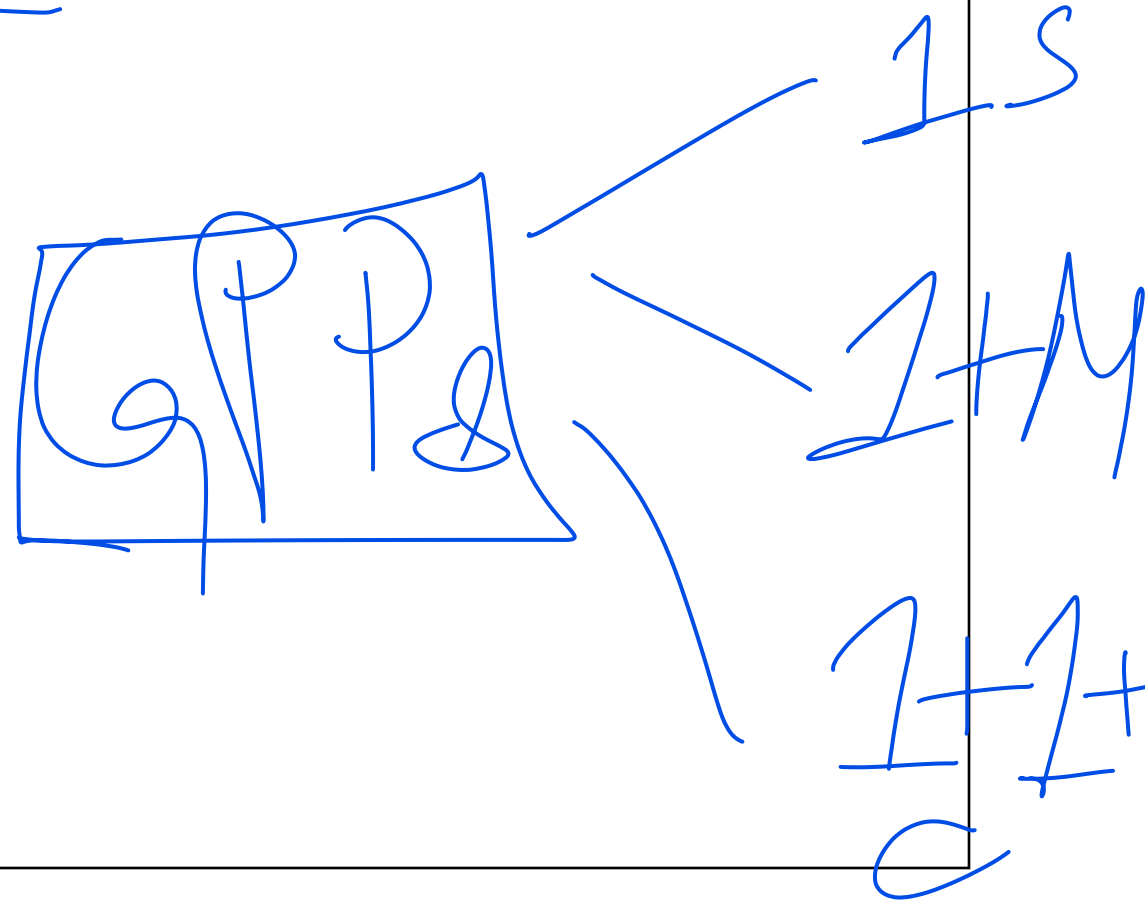
- A general purpose computer system consists of CPU's and multiple device controllers that are connected through a common bus .
- Each device controller is incharge of a specific type of device.
- A device controller maintains some local buffer storage and a set of special purpose registers.
- The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.
- Operating systems have a device drivers for each device controllers. This device driver understands the device controller and presents the uniform interface to the device to the rest of the operating system.
- To start an I/O operation, the device driver loads the appropriate registers within the device controller. The device controller examines the contents of these registers to determine what action to take (such as "read a character from the keyboard").
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.
- For bulk data transfer DMA(Direct memory access) is used instead of interrupt driven I/O.



# Computer-System Architecture

Computer systems are broadly classified into following types according to the number of general purpose processors used:

- Single-Processor Systems
- Multiprocessor Systems
- Clustered systems



# Single-Processor Systems

- On a single-processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.
- They may include other special-purpose processors as well. They may come in the form of device-specific processors, such as disk, keyboard and graphics controllers.
- All of these special-purpose processors run a limited instruction set and do not run user processes.
- Sometimes they are managed by the operating system, in that the operating system sends them information about their next task and monitors their status.

# Multiprocessor systems

- Multiprocessor systems(also known as **parallel systems** or **tightly coupled systems**) have two or more processors in close communication, sharing the computer bus, the clock, memory, and peripheral devices.
- Multiprocessor systems have three main advantages:
  1. **Increased throughput** : By increasing the number of processors , we expect to get more work done in less time.
  2. **Economy of scale**: Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies.
  3. **Increased reliability** : If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

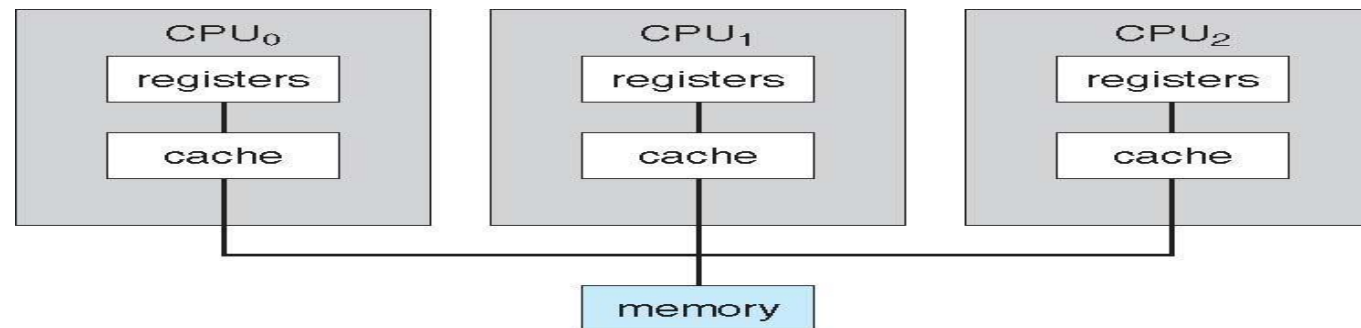
# Types of Multiprocessor systems

## Asymmetric multiprocessing:

- This scheme defines a master-slave relationship in which each processor is assigned a specific task.
- A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks. The master processor schedules and allocates work to the slave processors.

## Symmetric multi processing:

- SMP means that all processors are peers; no master-slave relationship exists between processors.
- In SMP each processor performs all tasks within the operating system. Fig below shows symmetric multi processing architecture.



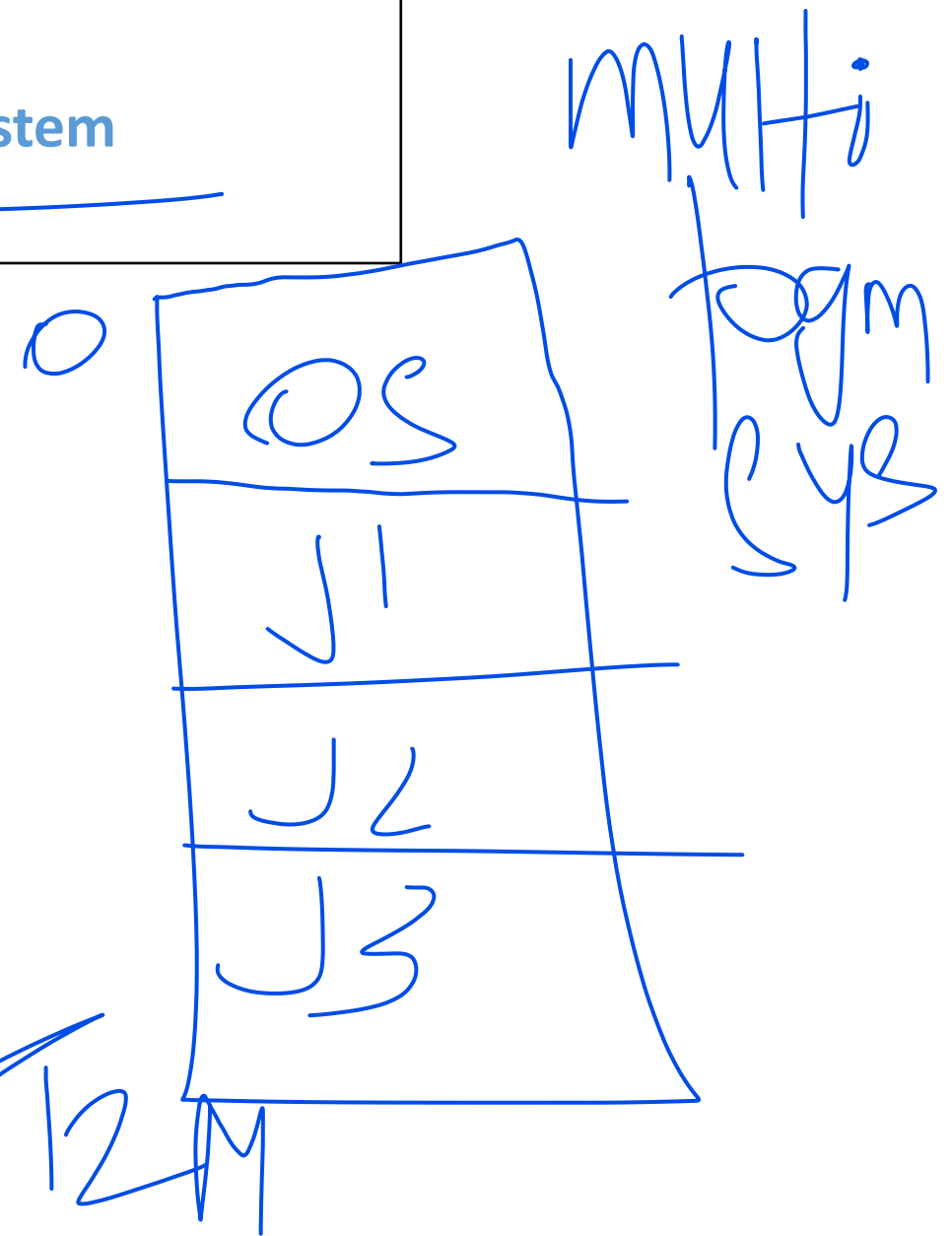
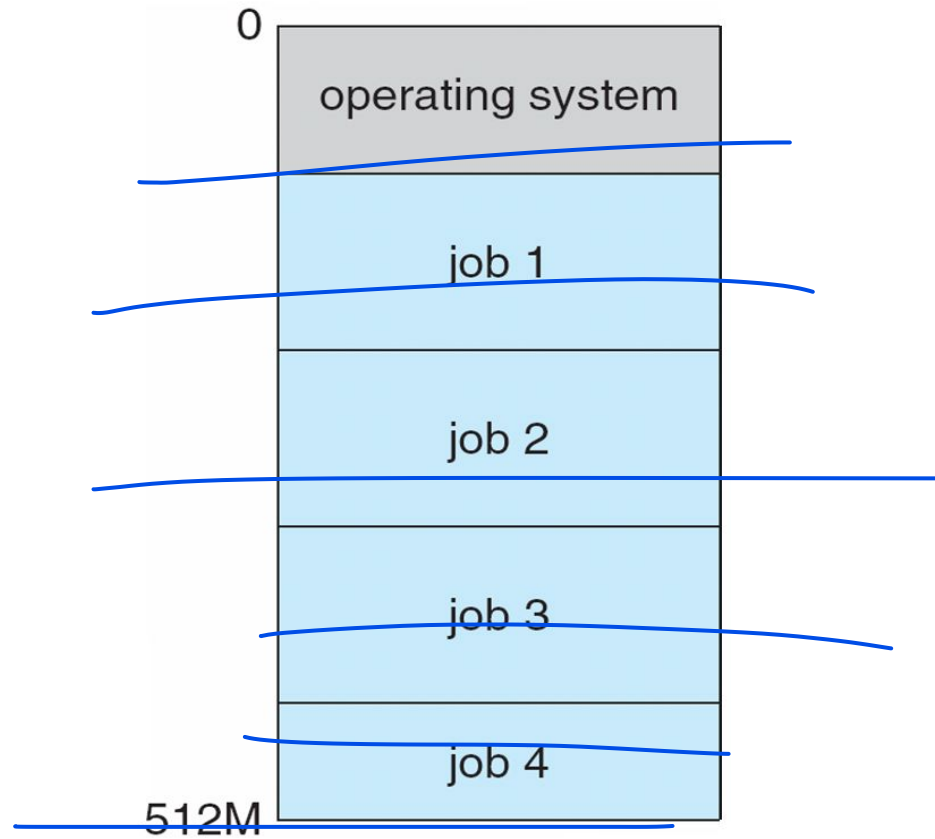
# Clustered Systems

- Clustered systems gather together multiple CPUs to accomplish computational work.
- Clustered systems differ from multiprocessor systems, in that they are composed of two or more individual systems coupled together.
- Clustered computers share storage and are closely linked via a local-area network (LAN).
- Clustering is usually used to provide high-availability service; that is, service will continue even if one or more systems in the cluster fail.
- Clustering can be structured **asymmetrically** or **symmetrically**.
- In asymmetric clustering, one machine is in hot-standby mode while the other is running the applications. The hot-standby host machine monitor the active server. If that server fails, the hot-standby host becomes the active server.
- In symmetric mode, two or more hosts are running applications, and are monitoring each other. This mode is obviously more efficient, as it uses all of the available hardware.

# Operating-System Structure

- An operating system provides the environment within which programs are executed.
- One of the most important aspects of operating systems is the ability to multiprogram. A single user cannot, in general, keep either the CPU or the IO devices busy at all times.
- Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the 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



• **Timesharing (multitasking system)** is logical extension of multiprogramming 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.

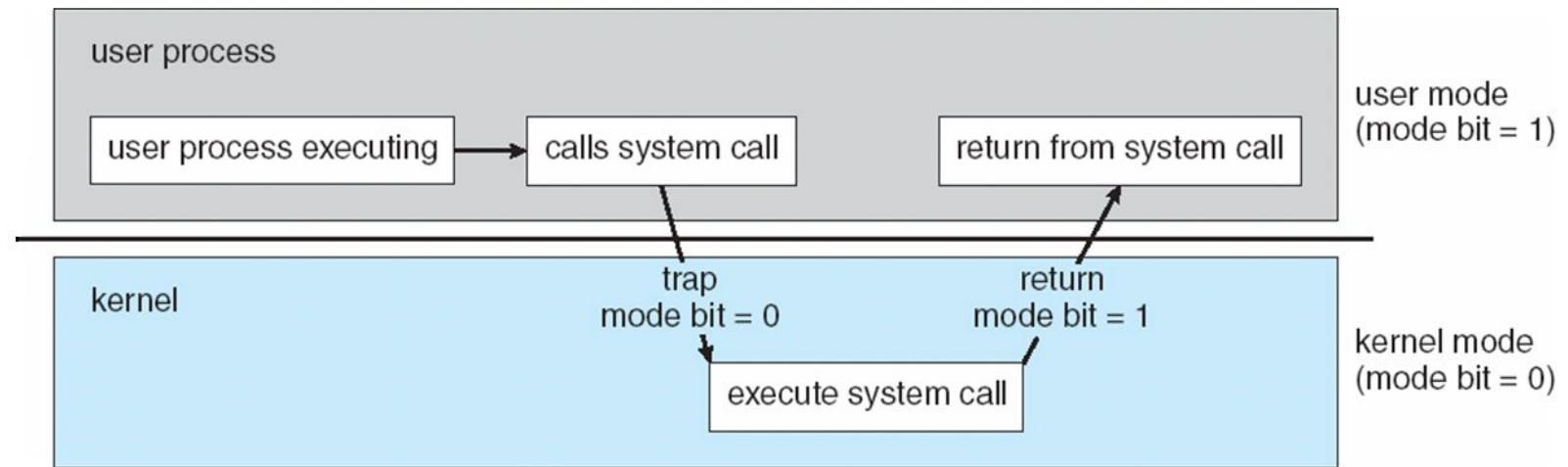


# Operating-System Operations

## Dual-mode operation

- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code.
- Most computer systems provide hardware support that allows to differentiate among various modes of execution.
- **Dual-mode** operation allows OS to protect itself and other system components.
- Computer system operates in two separate modes of operation: user mode and kernel mode (also called supervisor mode, system mode, or privileged mode).
- A bit, called the mode bit is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- **Mode bit** provided by hardware Provides ability to distinguish when system is running user code or kernel code

- When the computer system is executing on behalf of a user application, the system is in user mode.
- when a user application requests a service from the operating system (via a system call), it makes transition from user to kernel mode to fulfill the request. This is illustrated as shown in the figure below:



**Fig: Transition from user to kernel mode**

- At system boot time, the hardware starts in kernel mode.
- The operating system is then loaded and starts user applications in user mode.
- Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the state of the mode bit to 0).
- The dual mode of operation provides us with the means for protecting the operating system from errant users and errant users from one another.
- We accomplish this protection by designating some of the machine instructions that may cause harm as privileged instructions.
- The hardware allows privileged instructions to be executed only in kernel mode.
- If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

# System calls

System calls provides an interface between a process and operating system to allow user-level processes to request services of the operating system.

- When a system call is executed, it is treated by the hardware as a software interrupt. Control passes through the interrupt vector to a service routine in the operating system, and the mode bit is set to kernel mode.
- The kernel examines the interrupting instruction to determine which system call has occurred.
- The kernel executes the request, and returns control to the instruction following the system call.

# Timer

- Timer is used to prevent infinite loop / process hogging resources.
  - Timer is set to interrupt the computer after some time period.
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter is 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

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

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and 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)
    - 4 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
    - 4 Creating and deleting files and directories
    - 4 Primitives to manipulate files and directories
    - 4 Mapping files onto secondary storage
    - 4 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 – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

# 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 (**group 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

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - 4 **Network** is a communications path, **TCP/IP** most common
      - **Local Area Network (LAN)**
      - **Wide Area Network (WAN)**
      - **Metropolitan Area Network (MAN)**
      - **Personal Area Network (PAN)**
    - **Network Operating System** provides features between systems across network
      - 4 Communication scheme allows systems to exchange messages
      - 4 Illusion of a single system

# Special purpose systems

- Real-Time Embedded Systems

- These devices are found everywhere, from car engines and manufacturing robots to VCRs and microwave ovens. They tend to have very specific tasks.
- The systems they run on are usually primitive, and so the operating systems provide limited features. Usually, they have little or no user interface, preferring to spend their time monitoring and managing hardware devices, such as automobile engines and robotic arms.
- Embedded systems almost always run real-time operating systems. A real-time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data; thus, it is often used as a control device in a dedicated application.

- Multimedia Systems

- Multimedia describes a wide range of applications that are in popular use today. These include audio files such as MP3 DVD movies, video conferencing, and short video clips of movie previews or news stories downloaded over the Internet.

- Handheld Systems

- Handheld systems include personal digital assistants (PDAs), such as Palm and Pocket-PCs, and cellular telephones, many of which use special-purpose embedded operating systems. Developers of handheld systems and applications face many challenges, most of which are due to the limited size of such devices.

# Computing Environments - Traditional

- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

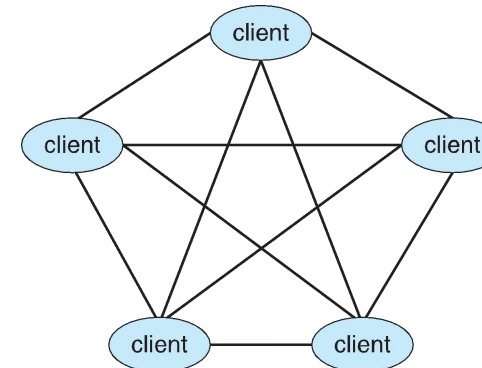
# Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**



# Computing Environments - Peer-to-Peer

- 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
    - 4 Registers its service with central lookup service on network, or
    - 4 Broadcast request for service and respond to requests for service via **discovery protocol**
- Examples include Napster and Gnutella, **Voice over IP (VoIP)** such as Skype



**End of Chapter 1**

# **UNIT-1**

## **Chapter-2 : Operating system structures**

# Operating System Services

- Operating systems provide an environment for execution of programs.
- It also provides certain services to programs and to the users of those programs.
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (UI).
    - 4 Example: **Command-Line (CLI) Interface**, **Graphical User Interface (GUI)**, **Batch Interface** etc..
  - **Program execution** - The system must be able to load a program into memory and to run that program, and execution, either normally or abnormally (indicating error).
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device. For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file information, permission management.

programs  
users

# Operating System Services (Cont.)

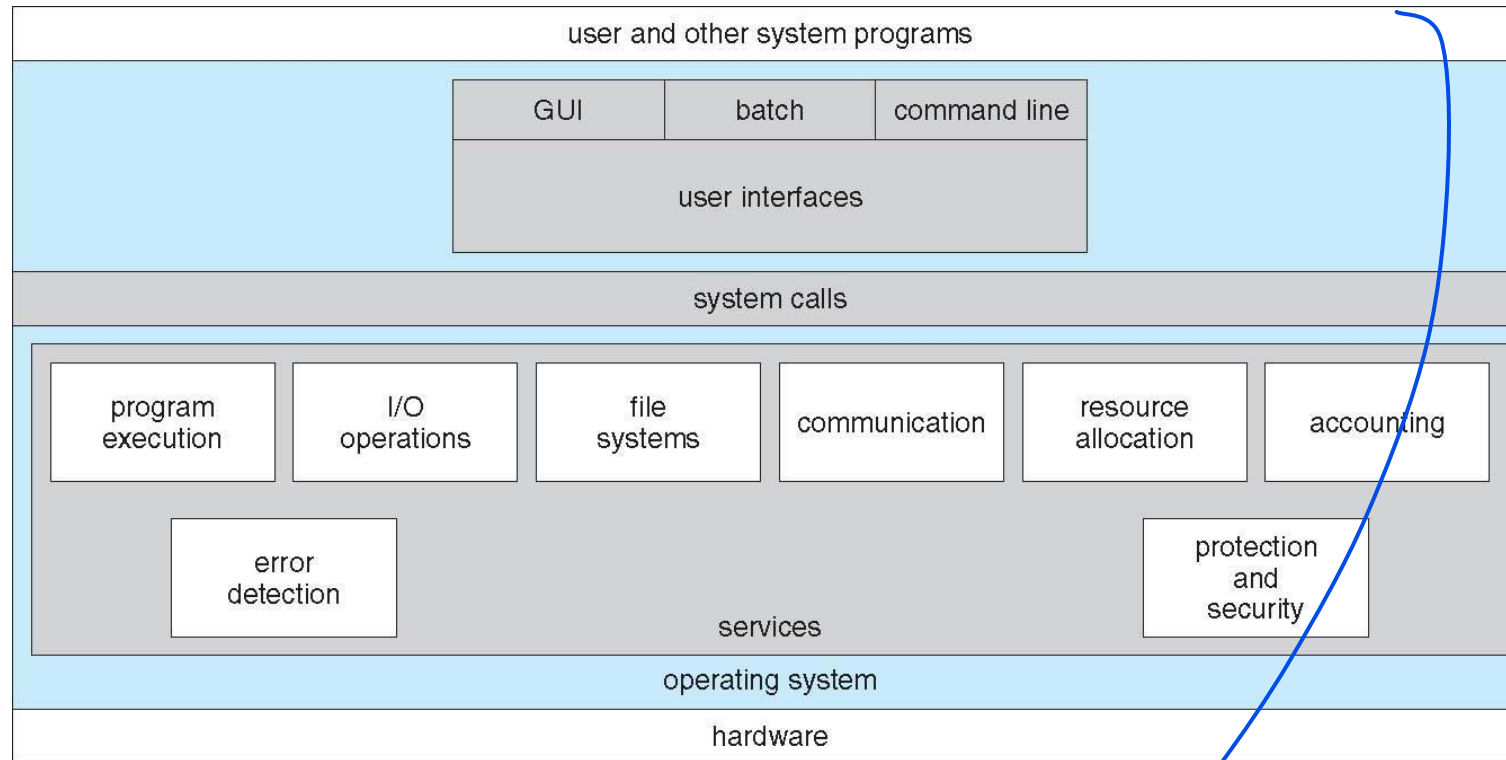
- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - 4 Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection** – OS needs to be constantly aware of possible errors.
  - 4 Errors May occur in the CPU and memory hardware, in I/O devices, in user program.
  - 4 For each type of error, OS should take the appropriate action to ensure correct and consistent computing.
  - 4 Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.

# Operating System Services (Cont.)

Another set of OS functions exists for ensuring the efficient operation of the system itself

- ❑ **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them.
  - ❑ Many types of resources are managed by OS - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code. OS should ensure fair allocation of resources among multiple users or multiple jobs.
- ❑ **Accounting** - OS maintains accounting information to keep track of which users use how much and what kinds of computer resources.
- ❑ **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other.
  - ❑ **Protection** involves ensuring that all access to system resources is controlled
  - ❑ **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

# A View of Operating System Services



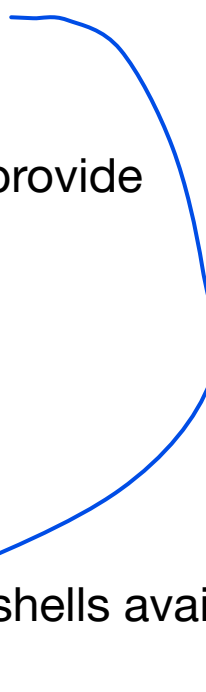
# User Operating System Interface - CLI

- Command Line Interface (CLI) or **command interpreter** allows direct command entry
  - 4 Sometimes implemented in kernel, sometimes by systems program
  - 4 Sometimes multiple flavors implemented – **shells**
  - 4 Primarily fetches a command from user and executes it
    - Sometimes commands built-in, sometimes just names of programs
      - » If the latter, adding new features doesn't require shell modification



# User Operating System Interface - GUI

---

- User-friendly **desktop** metaphor interface
    - Usually mouse, keyboard, and monitor
    - **Icons** represent files, programs, actions, etc
    - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a **folder**)
    - Invented at Xerox PARC
  - Many systems now include both CLI and GUI interfaces
    - Microsoft Windows is GUI with CLI “command” shell
    - Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available
    - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)
- 

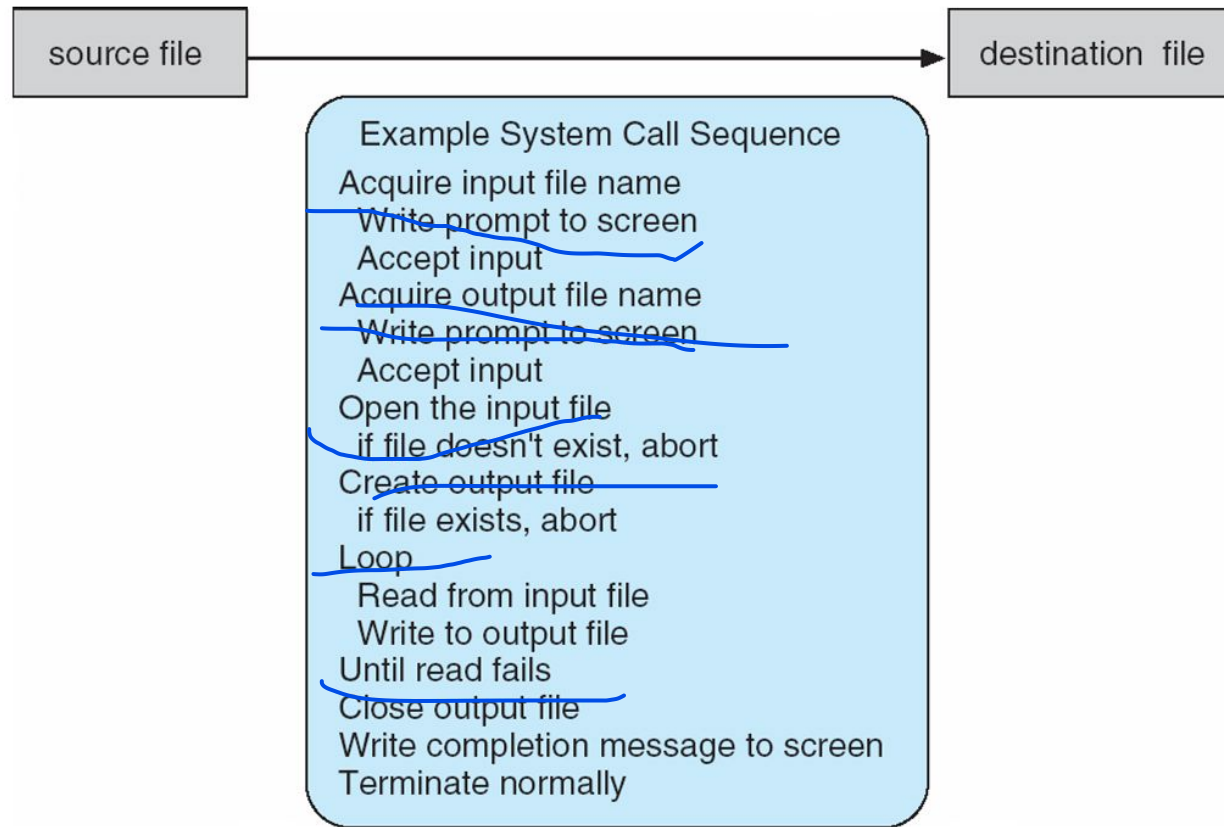
# System Calls

- System calls provides Programming interface to the services provided by the OS.
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use.
- Three most common APIs are ~~Win32 API for Windows, POSIX API for POSIX based systems~~ (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?  
( System calls are operating system specific but API's are independent of OS.)

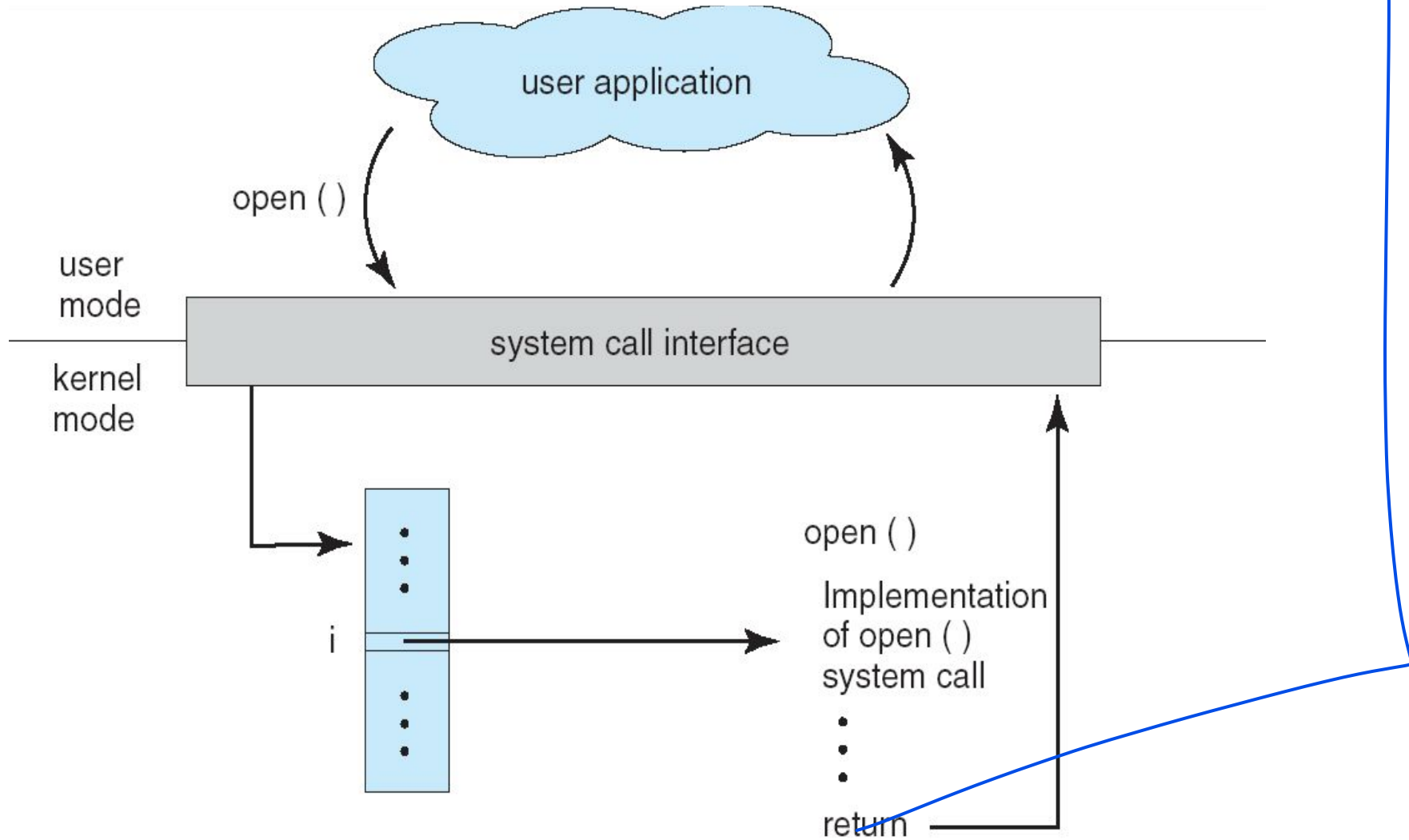
Ex: Code

# Example of System Calls

- System call sequence to copy the contents of one file to another file



# API – System Call – OS Relationship



# Types of System Calls

System calls can be grouped roughly into six major categories:

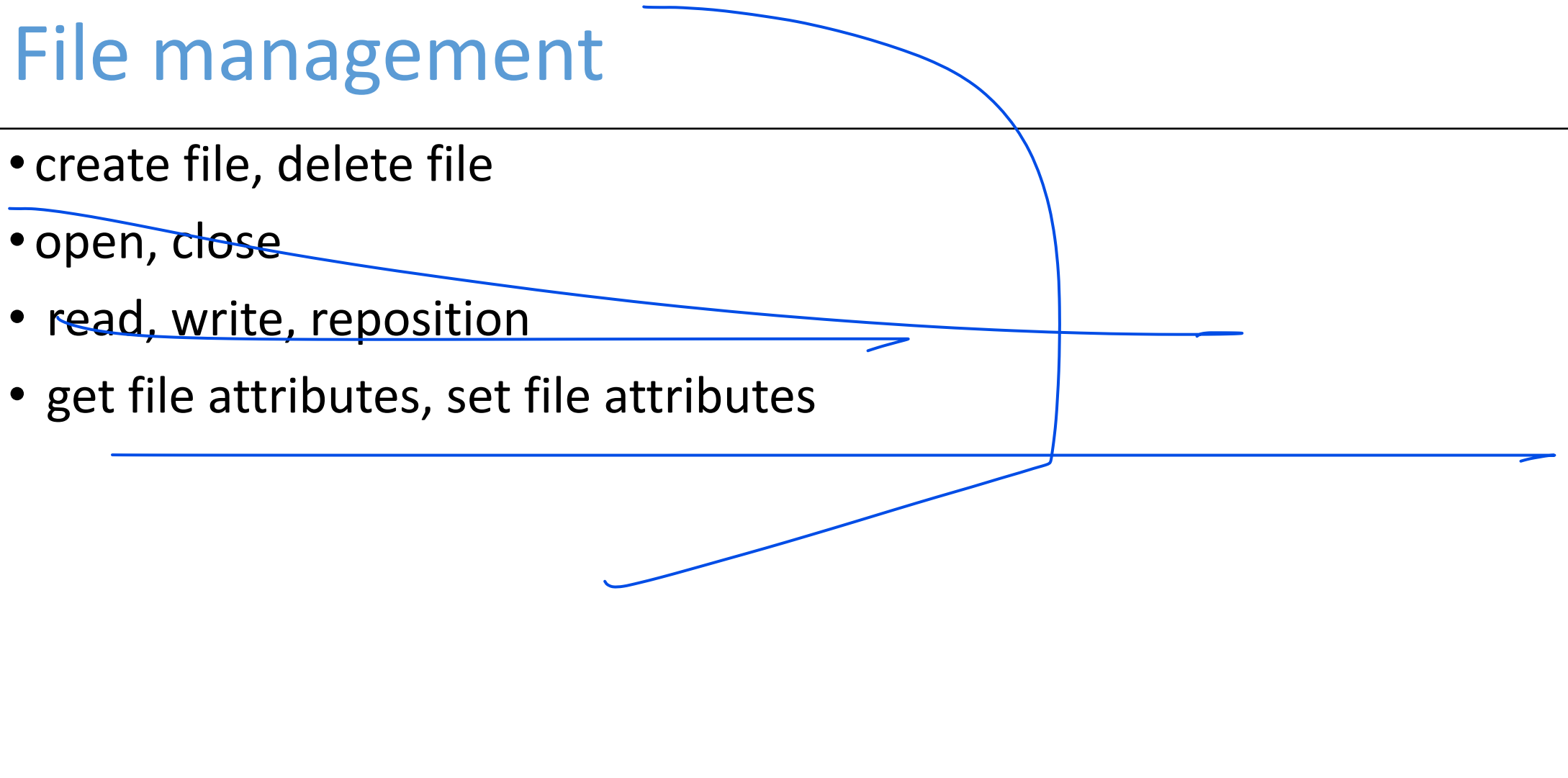
- Process control
- File manipulation
- Device manipulation
- Information maintenance
- Communications
- Protection

# Process control

## Process control System Calls

- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory

# File management

- create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- 
- Hand-drawn blue lines and arrows extending from the list items to the right side of the slide:
- A horizontal line with an arrow pointing right from the first item.
  - A horizontal line with an arrow pointing right from the second item.
  - A horizontal line with an arrow pointing right from the third item.
  - A horizontal line with an arrow pointing right from the fourth item.
  - A curved line starting from the top of the list, arching over the top right, and then pointing down towards the right side.

# Device management

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices



# Information maintenance



- get time or date, set time or date
- get system data, set system data
- get process, file, or device attributes
- set process, file, or device attributes

# Communications

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

# System programs

- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex
- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a registry - used to store and retrieve configuration information

# System Programs (cont'd)

- File modification
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

# Operating System Design and Implementation

In this section, we discuss problems we face in designing and implementing an operating system.

## Design goals

- Define goals and specifications.
- The design of the system will be affected by the choice of hardware and the type of system: batch, time shared, single user, multiuser, distributed, real time, or general purpose.
- Requirements may be much harder to specify. The requirements can, however, be divided into two basic groups: *user goals* and *system goals*.
  - *User goals* – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - *System goals* – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

## Operating System Design and Implementation (Cont.)

- Important principle to separate
  - Policy:** What will be done?
  - Mechanism:** How to do it?
- Mechanisms determine how to do something, policies decide what will be done
  - The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.

# Operating system structure



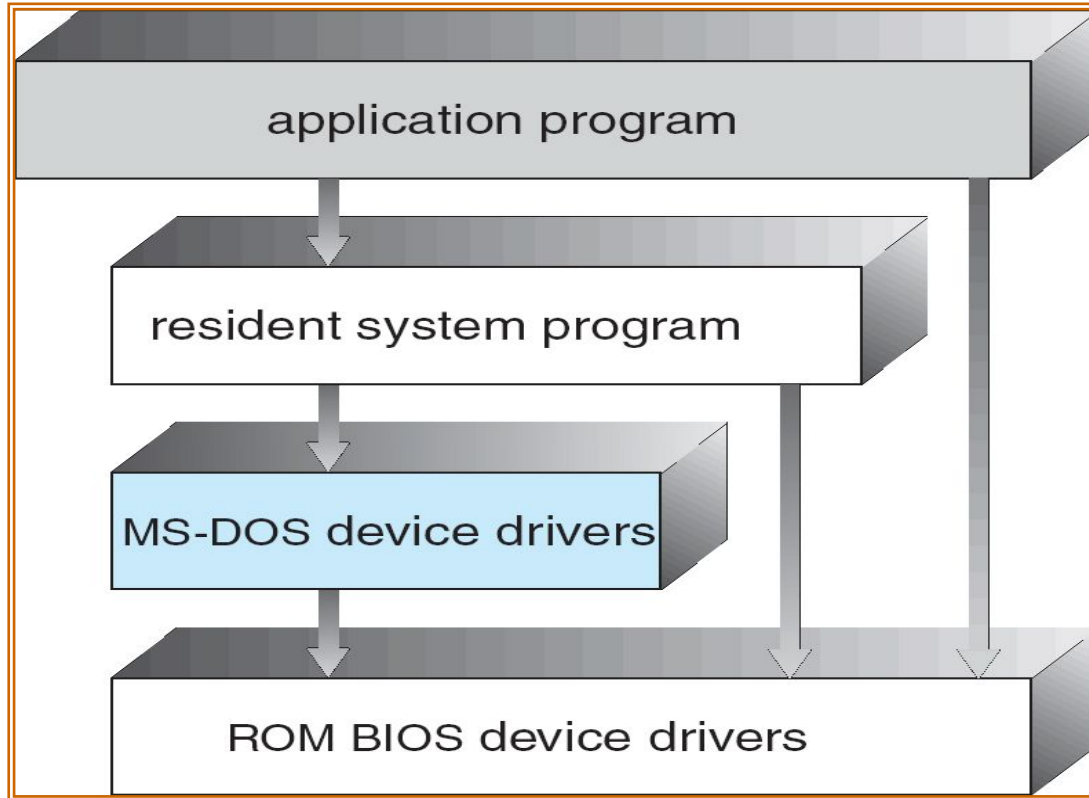
- Simple Structure
- Layered Approach
- Microkernels
- Modules

# Simple Structure

- MS-DOS was written to provide the most functionality in the least space.
    - Not divided into modules
    - In MS-DOS , the interfaces and levels of functionality are not well separated.
- Example: In MS-DOS, application programs are able to access the basic I/O routines to write directly to the display and disk drives. Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail.

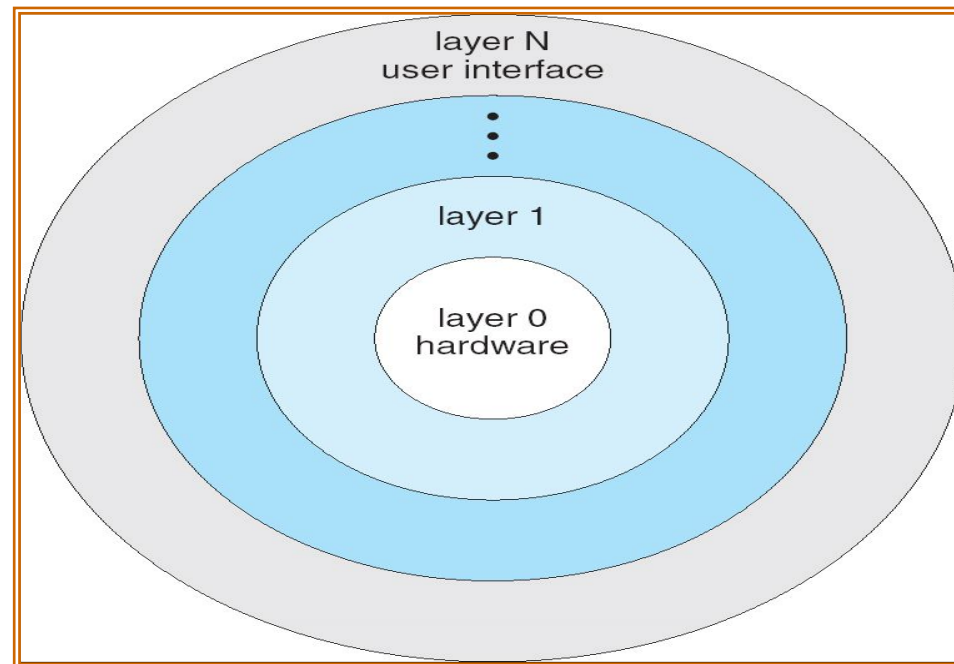


# MS-DOS Layer Structure



# Layered Approach

- A Layered approach in which the operating system is broken up into a number of layers (levels).
- The bottom layer (layer 0) is the hardware; the highest (layer  $N$ ) is the user interface.
- This layering structure is depicted in Figure



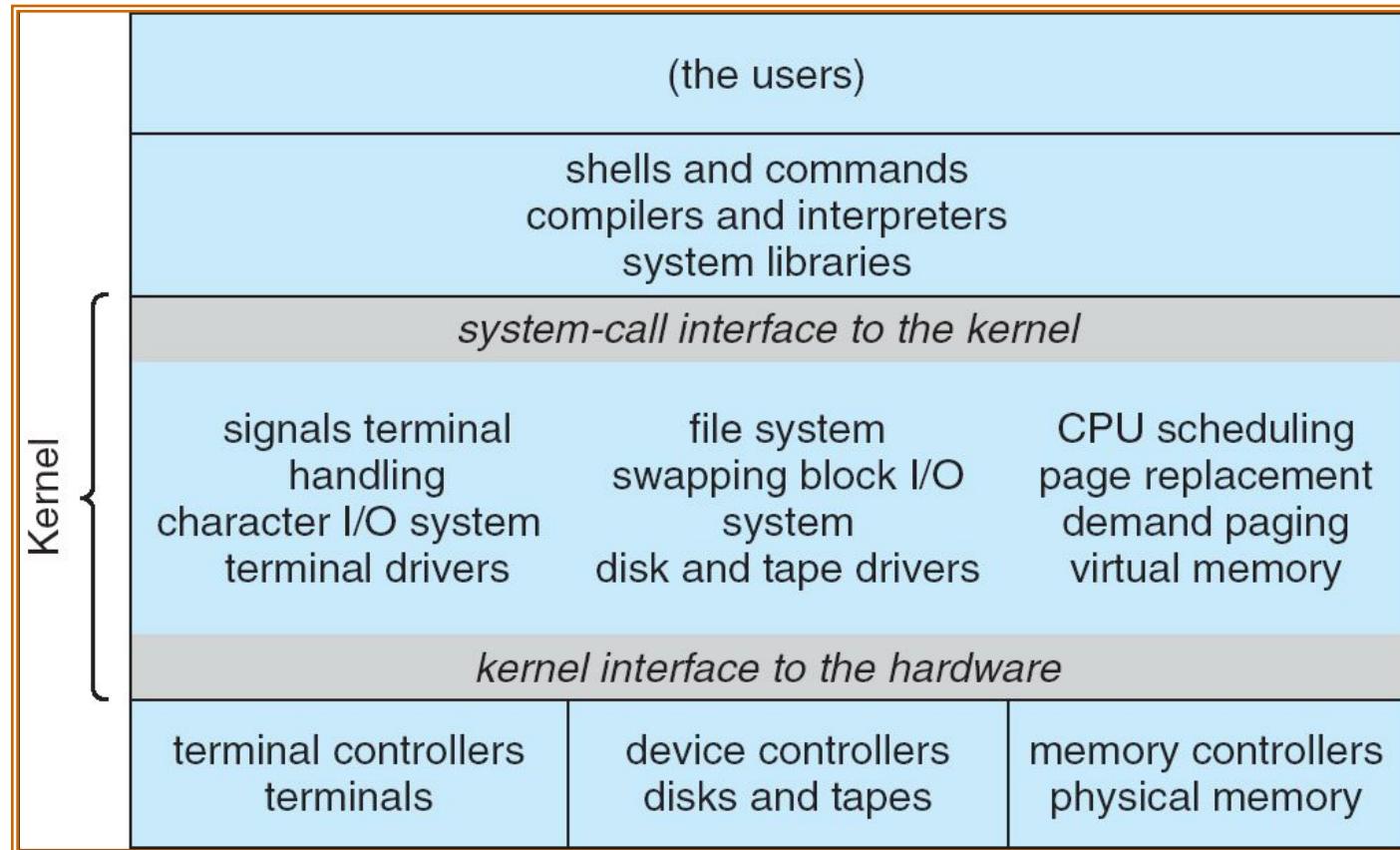
- An operating-system layer is an implementation of an abstract object made up of data and the operations that can manipulate those data.
- With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers.
- The main advantage of the layered approach is simplicity of construction and debugging.
- A layer does not need to know how operations are implemented; it needs to know only what these operations do. Hence, each layer hides the existence of certain data structures, operations, and hardware from higher-level layers.
- The major difficulty with the layered approach involves appropriately defining the various layers. Because a layer can use only lower-level layers, careful planning is necessary.
- A final problem with layered implementations is that they tend to be less efficient than other types. Executing a system call in layered system takes longer duration than on a non layered system.

# Example of Layered Approach

## Traditional UNIX system

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
  - Systems programs
  - The kernel
    - 4 Consists of everything below the system-call interface and above the physical hardware
    - 4 Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

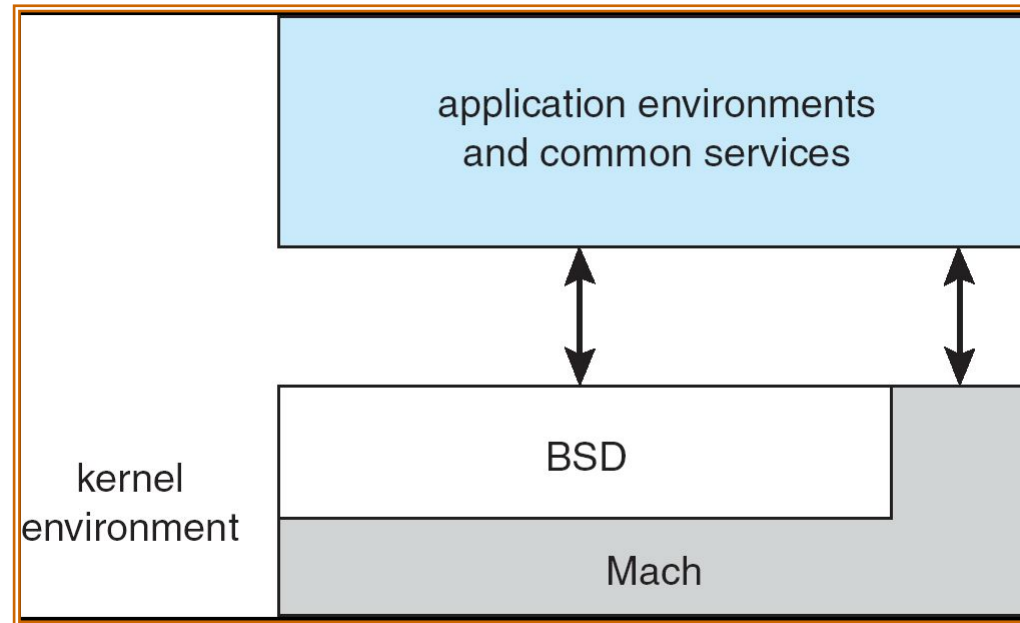
# UNIX System Structure



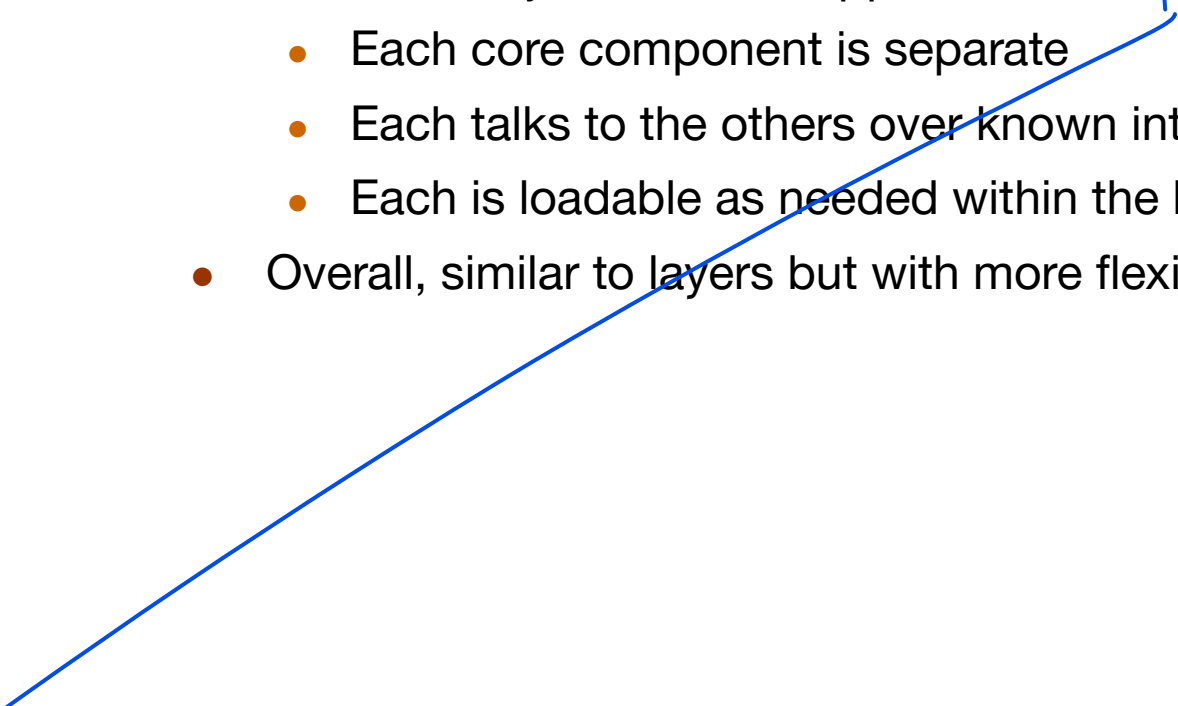
# Microkernels

- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs.
- however, microkernels provide minimal process and memory management, in addition to a communication facility.
- The main function of the microkernel is to provide a communication facility between the client program and the various services that are also running in user space. Communication is provided by *message passing*.
- One benefit of the microkernel approach is ease of extending the operating system. All new services are added to user space and consequently do not require modification of the kernel.
- The resulting operating system is easier to port from one hardware design to another.
- The microkernel also provides more security and reliability, since most services are running as user-rather than kernel-processes. If a service fails, the rest of the operating system remains untouched.

# Mac OS X Structure

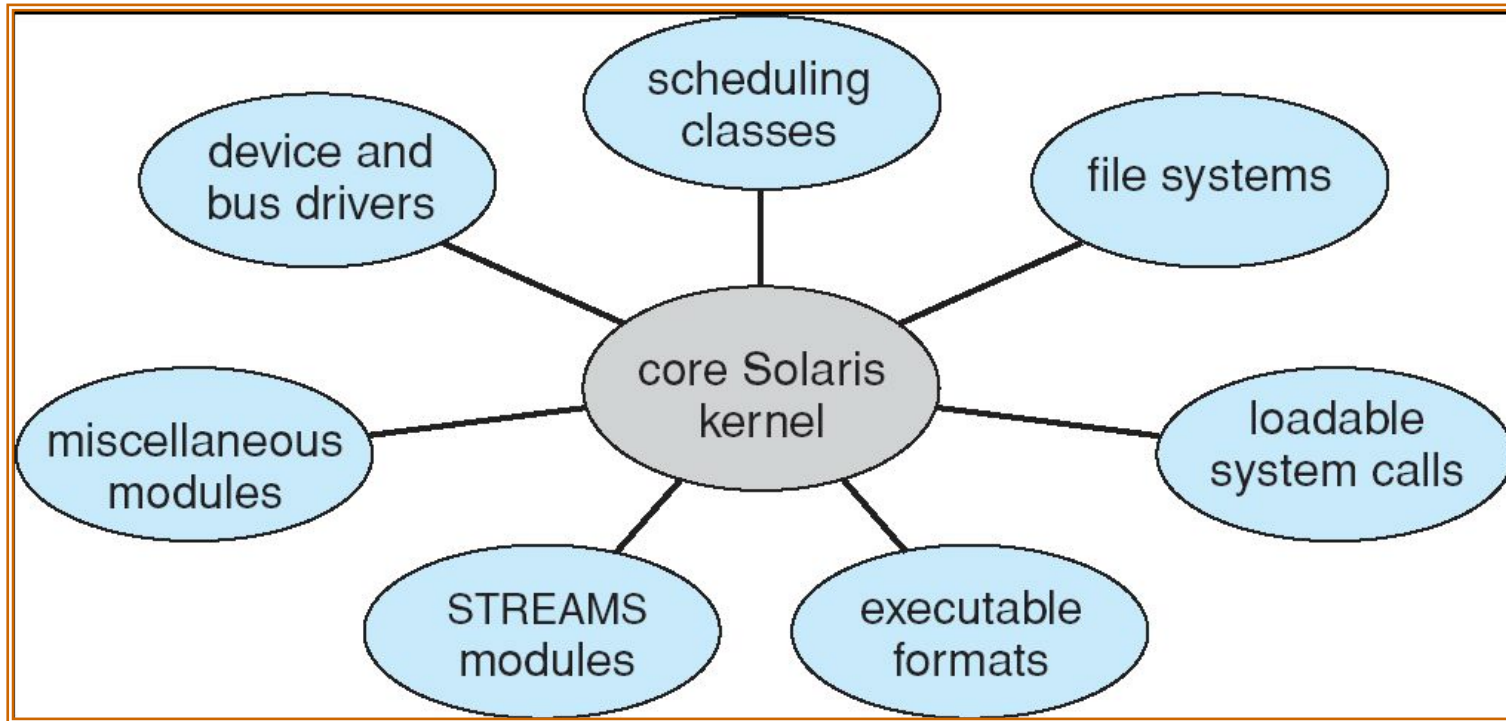


# Modules

- Most modern operating systems implement kernel modules
    - Uses object-oriented approach
    - Each core component is separate
    - Each talks to the others over known interfaces
    - Each is loadable as needed within the kernel
  - Overall, similar to layers but with more flexible
- 



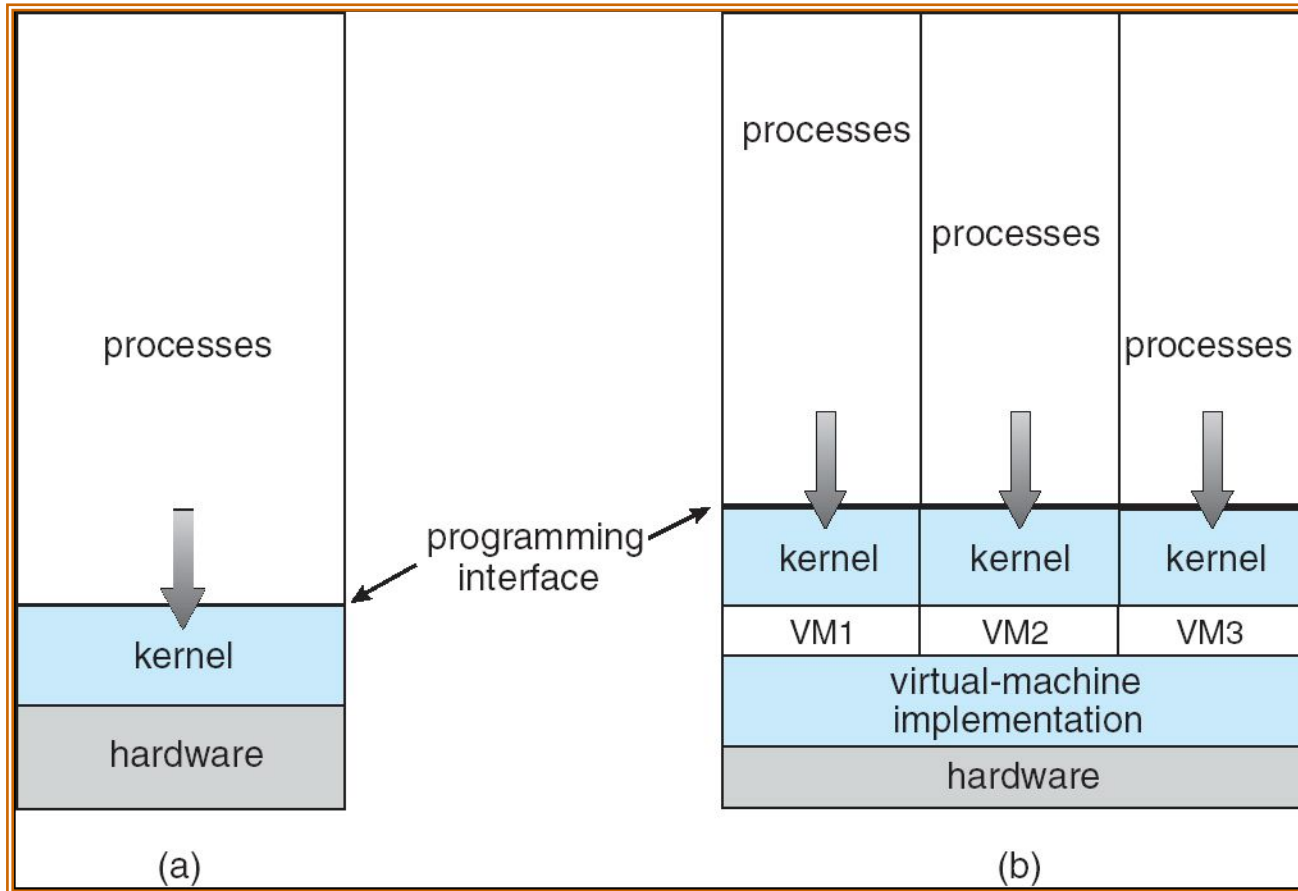
# Solaris Modular Approach



# Virtual Machines

- A *virtual machine* takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
- A virtual machine provides an interface *identical* to the underlying bare hardware
- The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory
- The resources of the physical computer are shared to create the virtual machines
  - CPU scheduling can create the appearance that users have their own processor
  - Spooling and a file system can provide virtual card readers and virtual line printers
  - A normal user time-sharing terminal serves as the virtual machine operator's console

## Virtual Machines (Cont.)

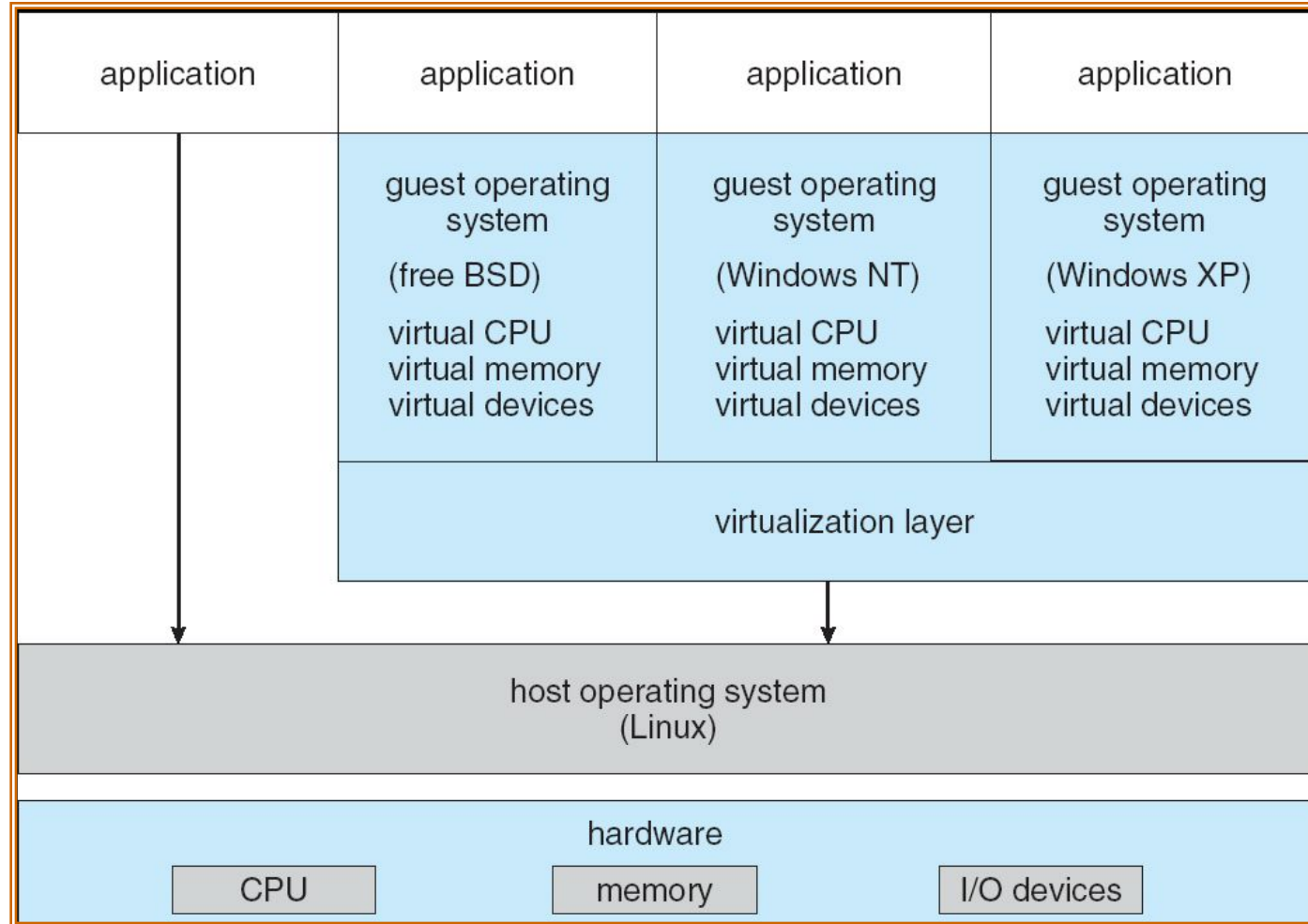


(a) Nonvirtual machine (b) virtual machine

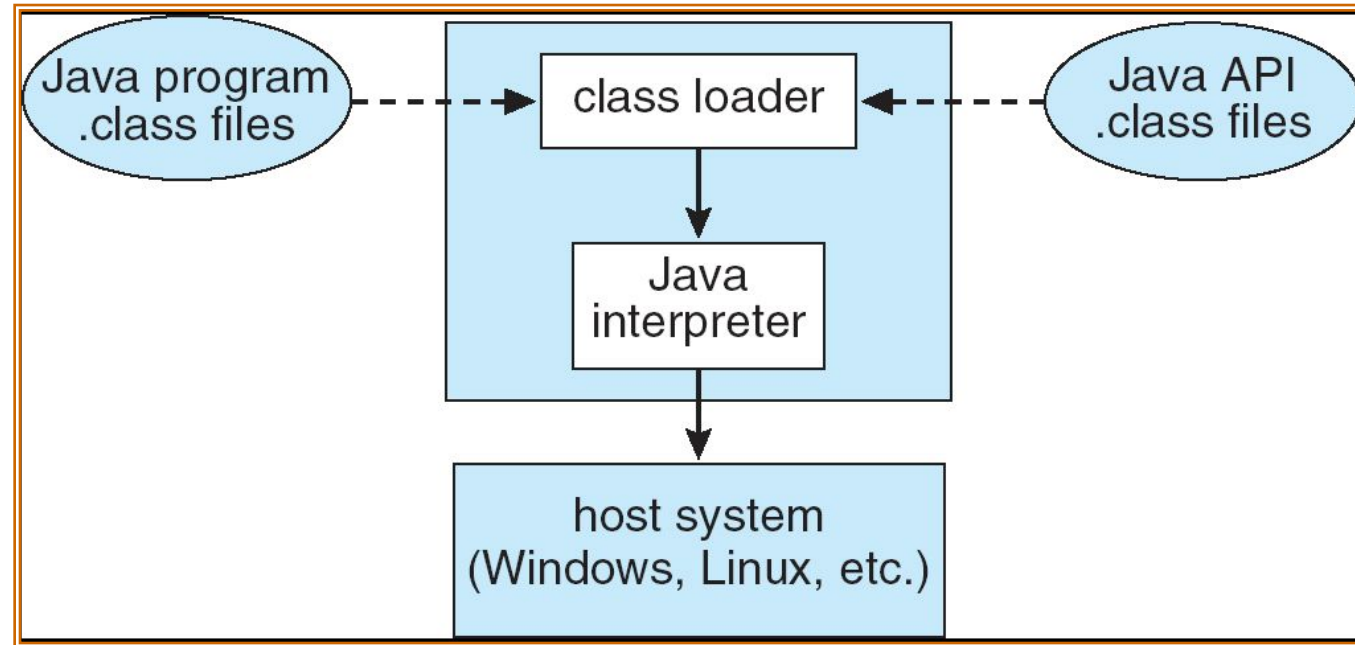
## Virtual Machines (Cont.)

- The virtual-machine concept provides complete protection of system resources since ,each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine

# VMware Architecture



# The Java Virtual Machine



**Thank you**