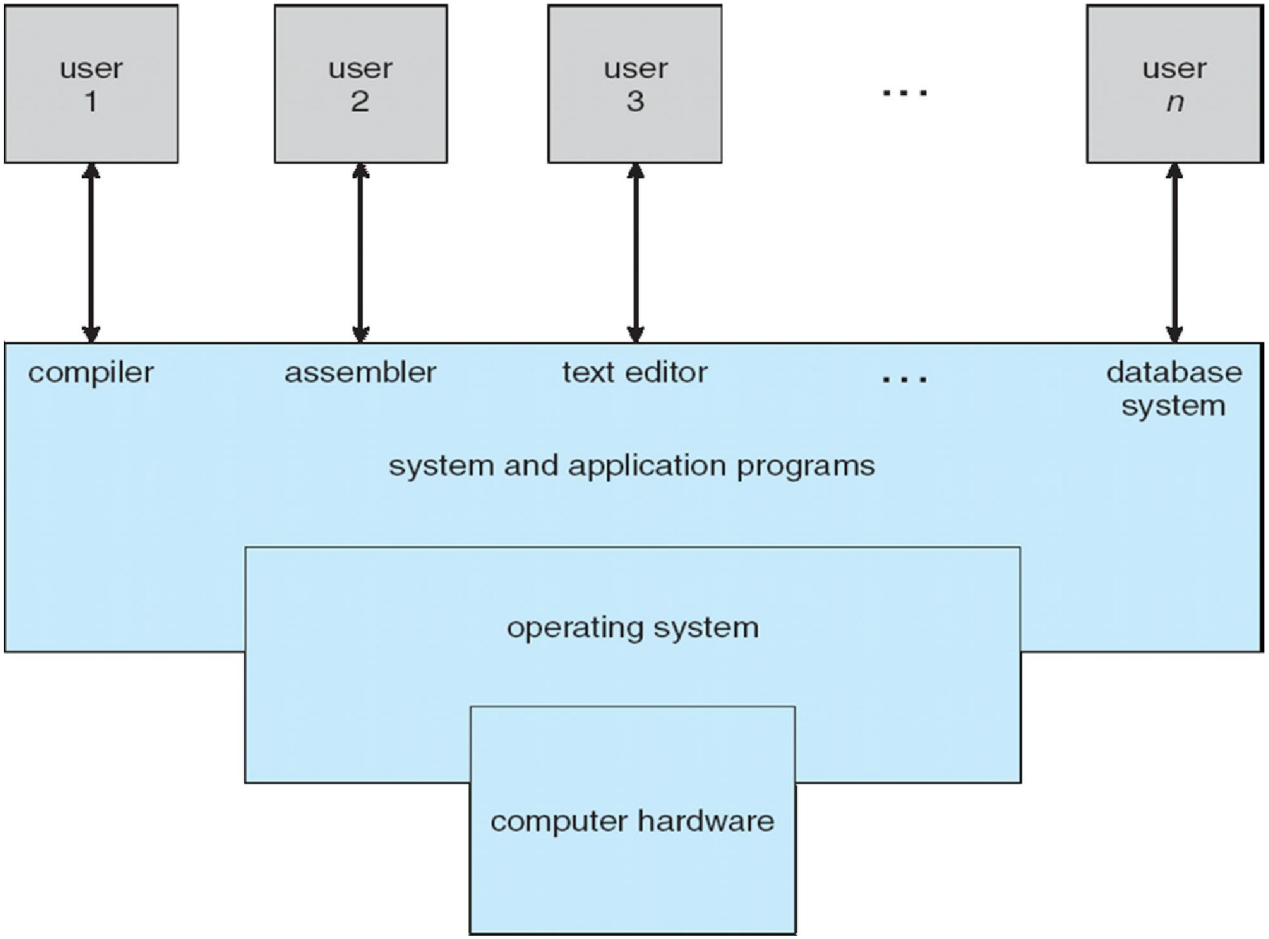
Unit 1
Introduction to Operating Systems and its Structures

Contents

1.1 What does operating system do?

- OS is a program that acts as an **interface** between a user of a computer and the computer hardware
- Operating system goals:
  - To manage all the resources in an efficient manner
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use

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


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

- The Operating System provides services to the user and the computer
- The services provided by Operating systems should be viewed in two different perspectives
  - **User Perspective**
  - **System Perspective**

User View

- The user's view of the computer varies according to the interface being used.
- For Single users – **Ease of use and Convenience.** Don't Care about resource utilization
- For Shared Computers Connected to Mainframe or minicomputer – All users must be happy. Hence **Maximize resource utilization**
- For Workstations and Servers – compromise both **convenience and resource utilization**

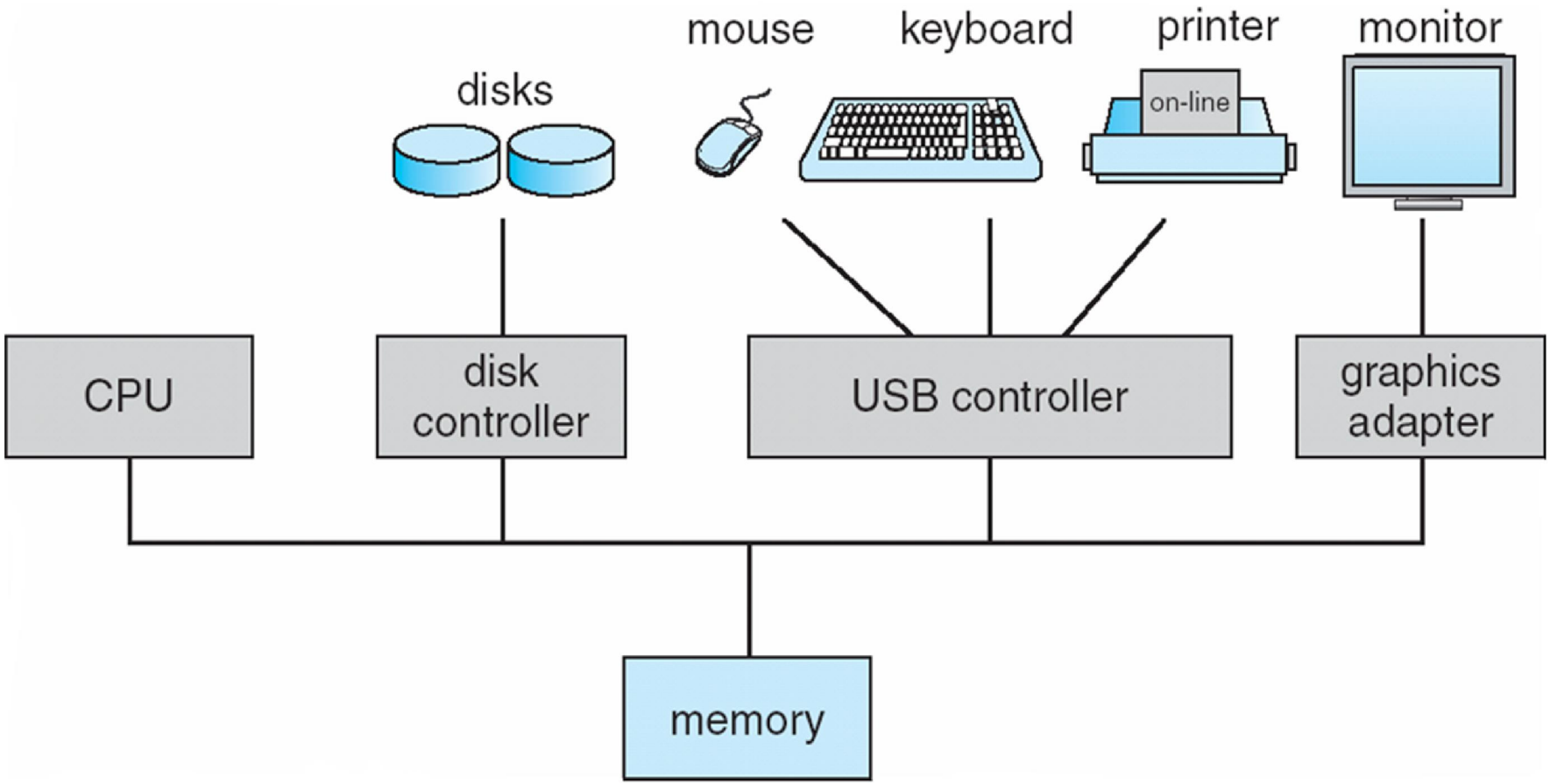
Systems View

- From the computer's point of view, the operating system is the program most intimately involved with the hardware.
- 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

1.2 Computer System Organization

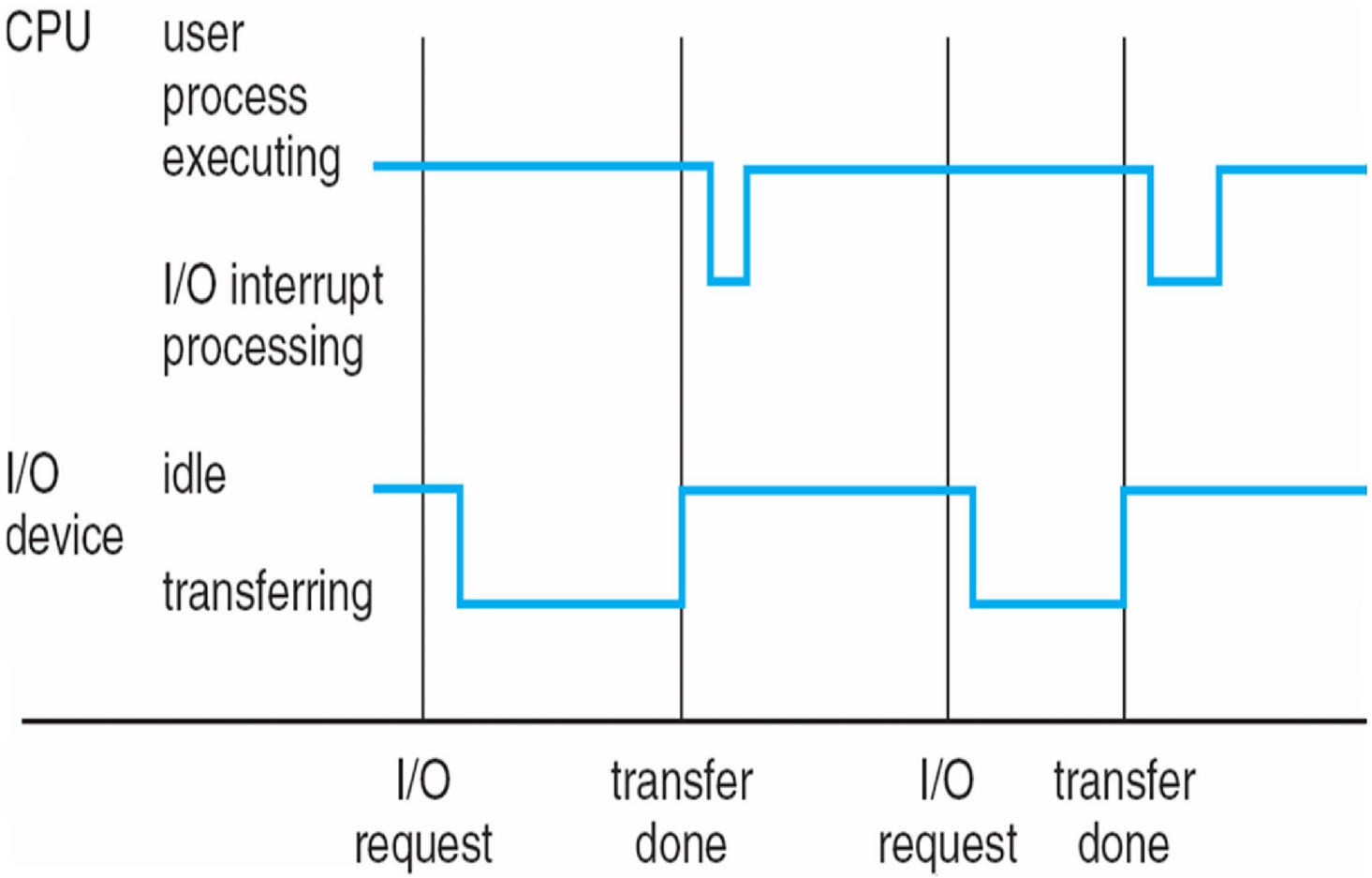
- To understand the Organization of computer system, we have to learn the following concepts.
  - Computer-System Operation
  - Storage Structure
  - I/O Structure

1.2.1 Computer-System Operation



10

- For a computer to start running, needs to have an initial program to run called **bootstrap program.**
- **Program / IO devices calls the OS for service, then**
  - 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
  - Device controller informs CPU that it has finished its operation by causing an interrupt


CPU
user process executing
I/O interrupt processing
I/O device
idle
transferring
I/O request
transfer done
I/O request
transfer done

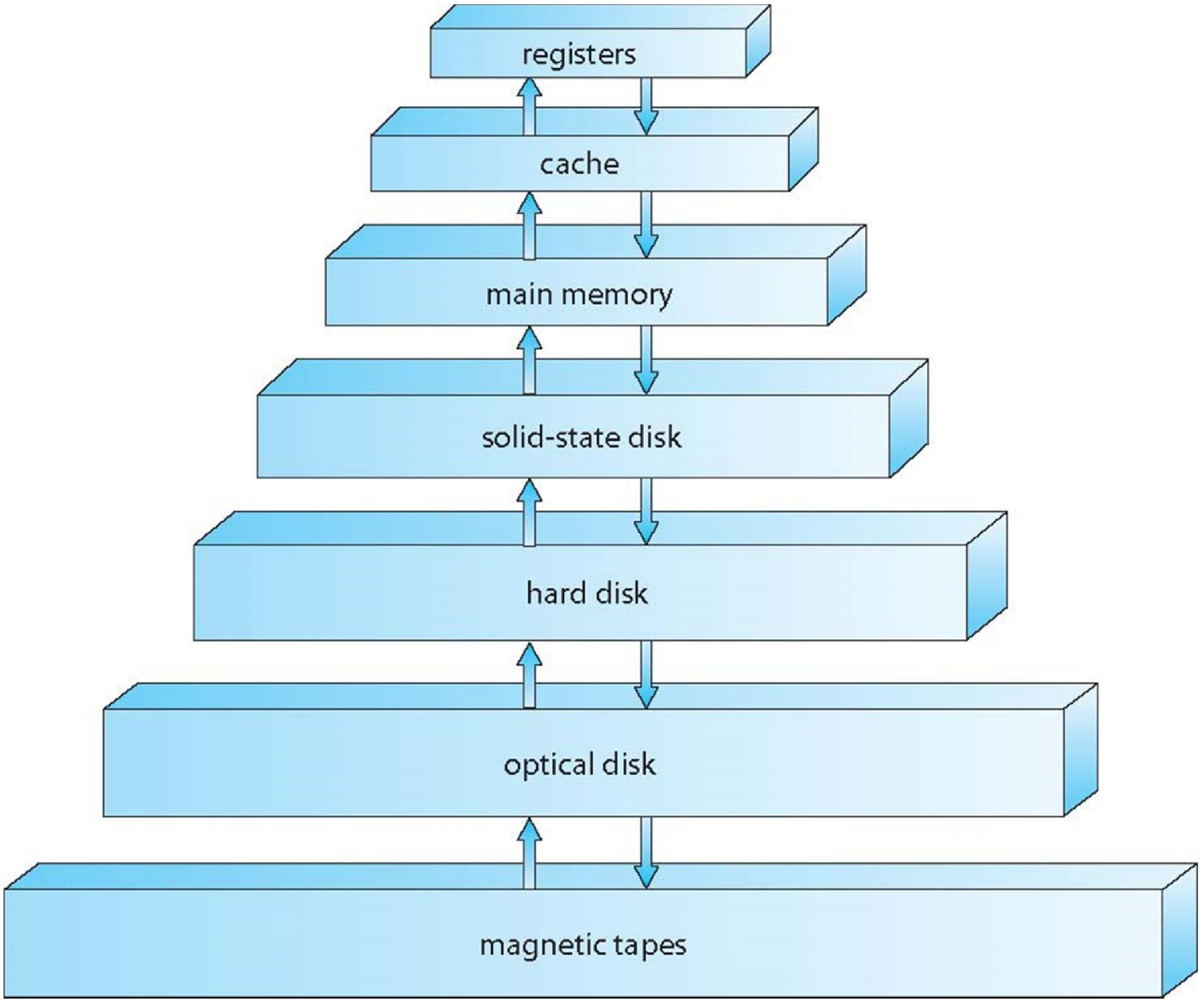
- Interrupt must transfer control to the appropriate interrupt service routine
- Interrupt architecture must save the address of the interrupted instruction
- These service routines of various devices are stored in the low memory.
- These addresses are maintained in a table by the OS
- After the interrupt is serviced, the saved return address is loaded into the program counter and computation resumes as though the interrupt has not occurred

1.2.2 Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- 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

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage

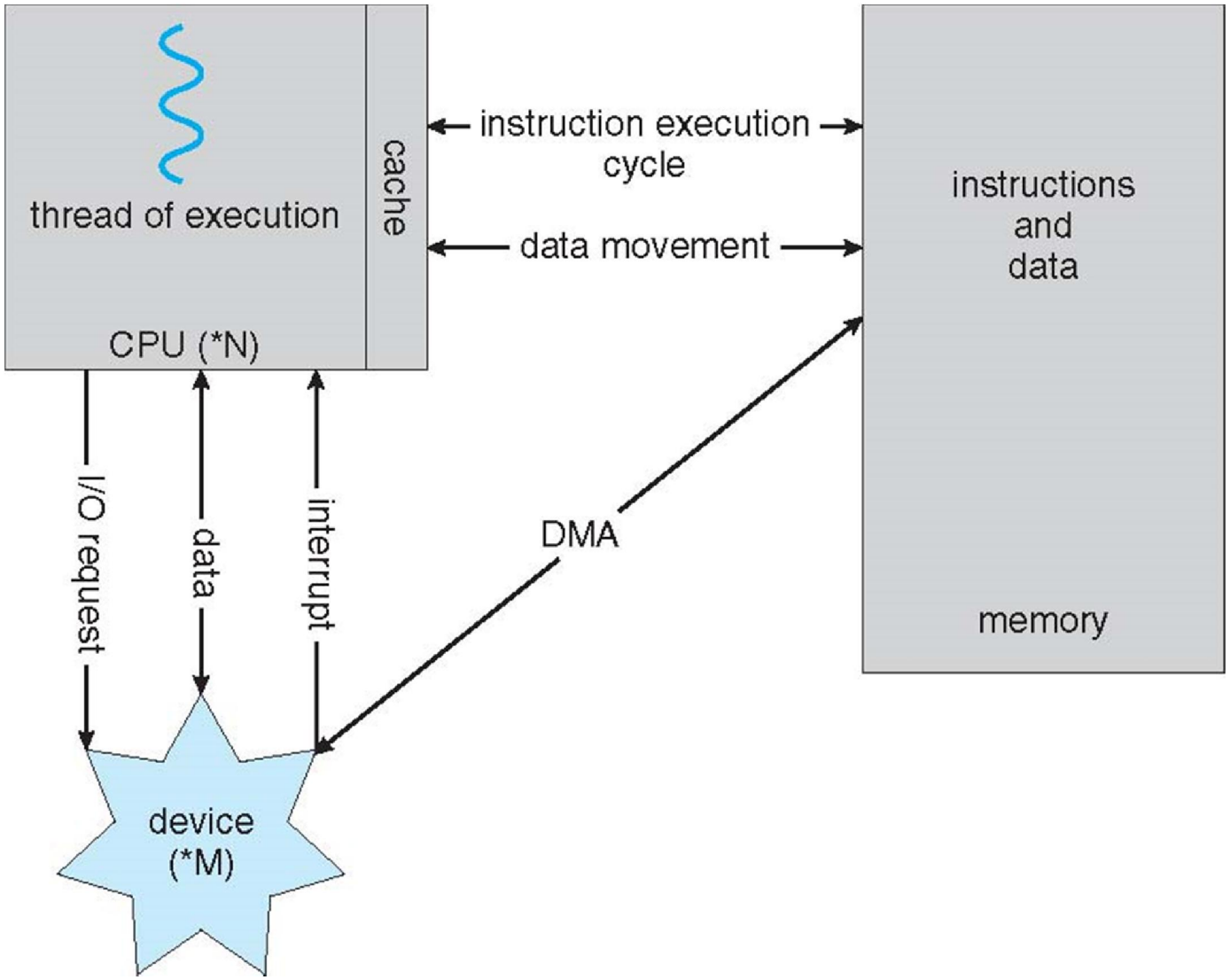
Storage Device Hierarchy



Storage Device Comparison

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

1.2.3 I/O Structure



1.3 Types of Systems

- Based on Computer System architecture
  - Single Processor Systems, Multi-Processor , Clustered System
- Based on OS Architecture
  - Multiprogramming, Multitasking
- Based on Computing Environment
  - Traditional Computing, Client Server Computing, Peep-to-Peer Computing, Web Based Computing
- Based on Special Functionality
  - Real Time embedded Systems, Multimedia Systems, Hand Held Systems
- Based on Networking
  - Local Area Network (LAN), Wide Area Network (WAN), Metropolitan Area Network (MAN)

1.3.1 Computer System Architecture

- A Computer System may be organized in a number of different ways, which we can categorize roughly according to the number of general purpose processors used.
  - Single processor systems
  - Multi processor systems
  - Clustered Systems

Single Processor Systems

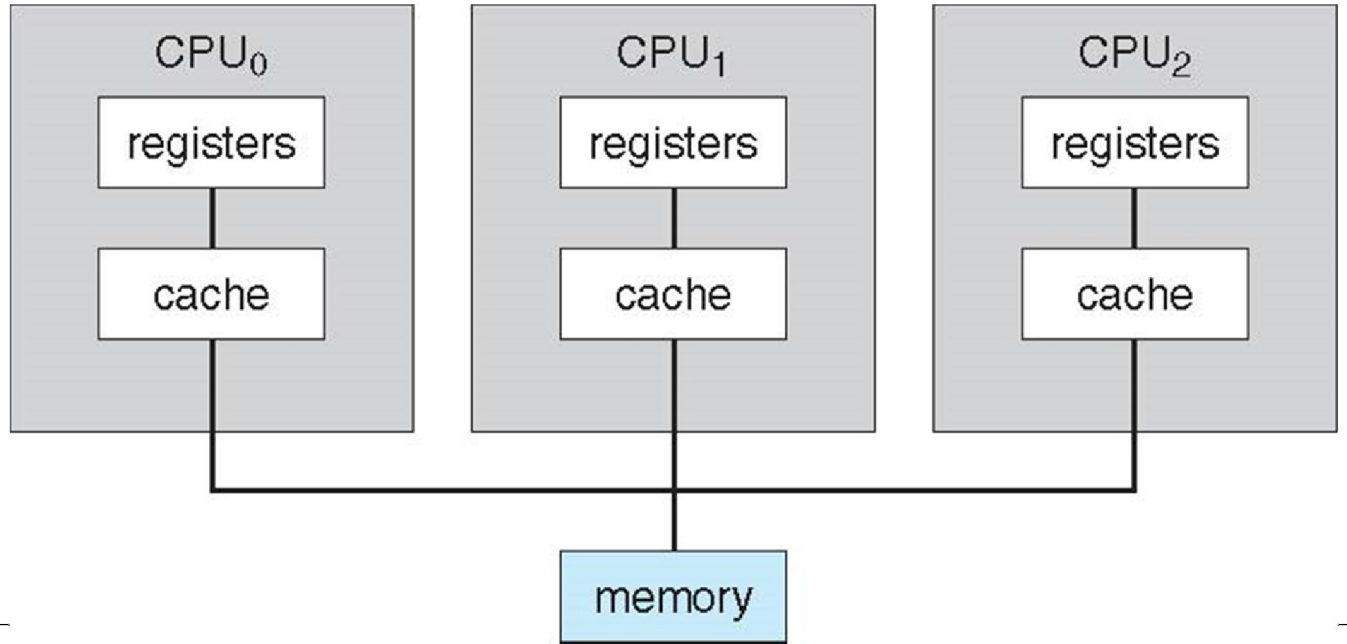
- Most systems use single processor systems.
- They perform only one process at a given time, and it carries out the next process in the queue only after the current process is completed.
- OS monitors the status of them and also sends them next executable program.
- It is suitable for general purpose computers , as it cannot run multiple processes in parallel.

Multi-Processor Systems

- Multi Processor Systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

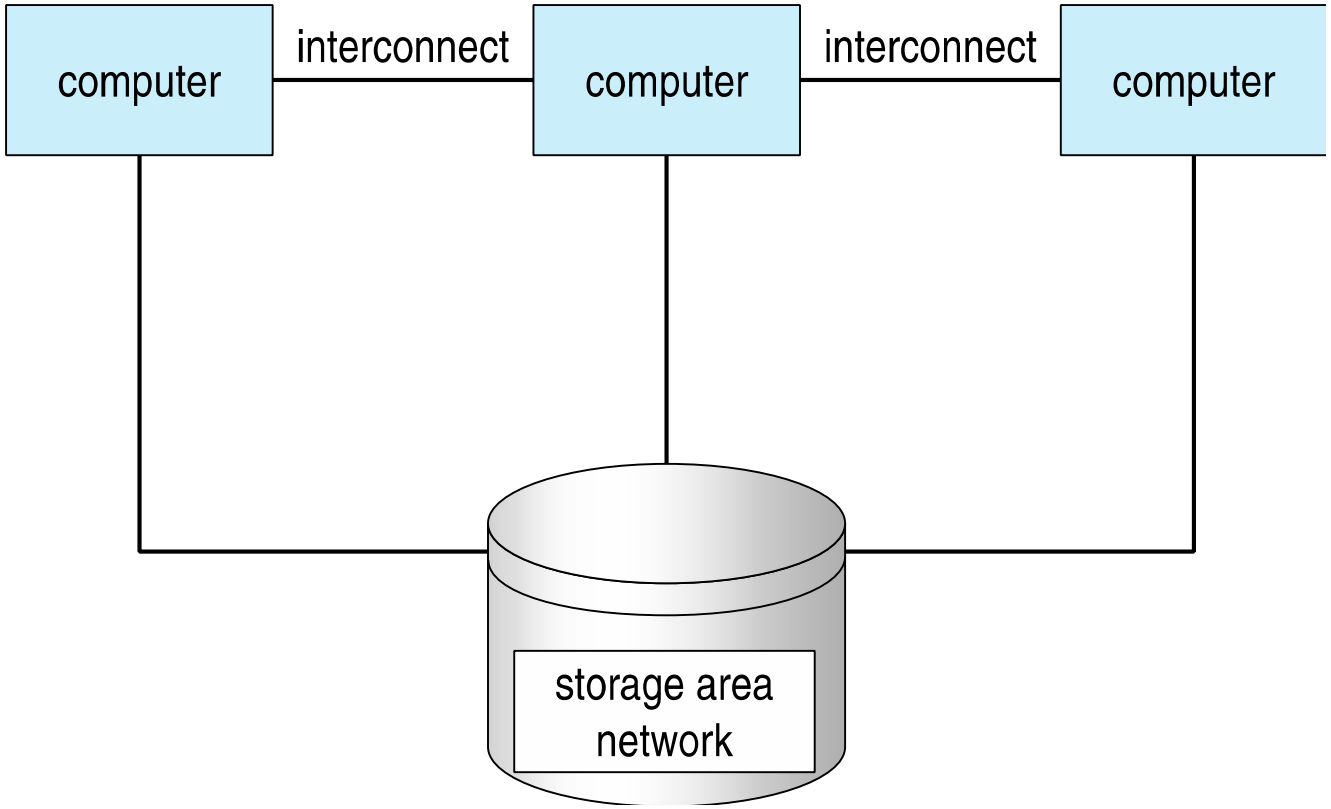
- Multiprocessor systems have three main advantages
  - **Increased throughput**-more work done in less time
  - **Economy of scale**-Common bus, peripherals, memory, power etc
  - **Increased reliability-** Failure of one will not affect the functionality of the system

- The multiple-processor systems in use today are of two types. Some systems use
- **Asymmetric multiprocessing-** all processors not treated equally. So each processor is assigned a specific task
- **Symmetric multiprocessing-** All processors are treated equally in which each processor performs all tasks within the operating system.



Clustered Systems

- Another type of multiprocessor system is a **clustered system, which gathers** together multiple CPUs.
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**

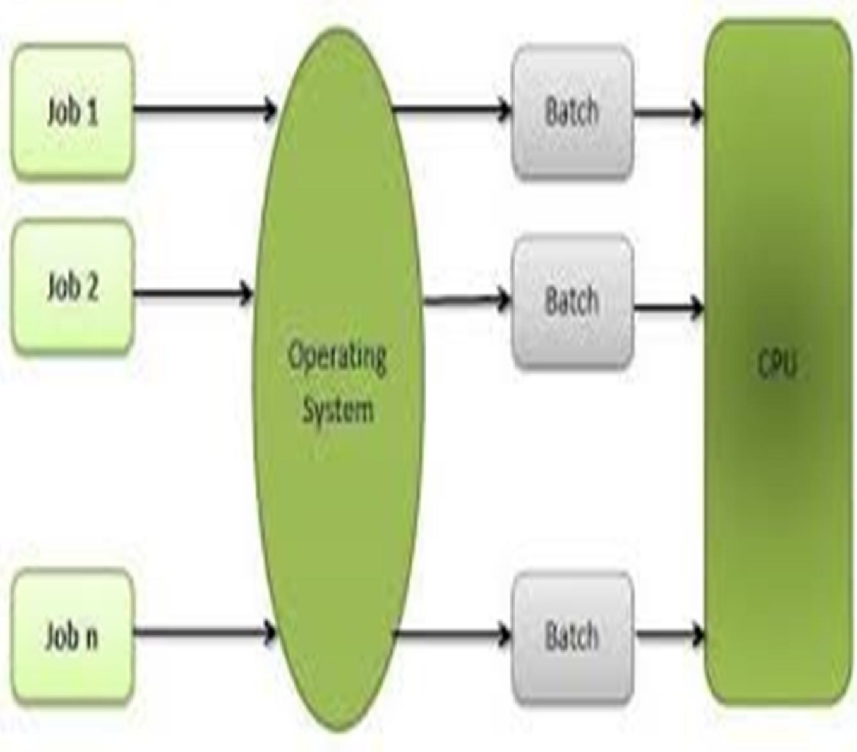

computer
interconnect
computer
interconnect
computer
storage area
network

1.3.2 Operating System Architecture

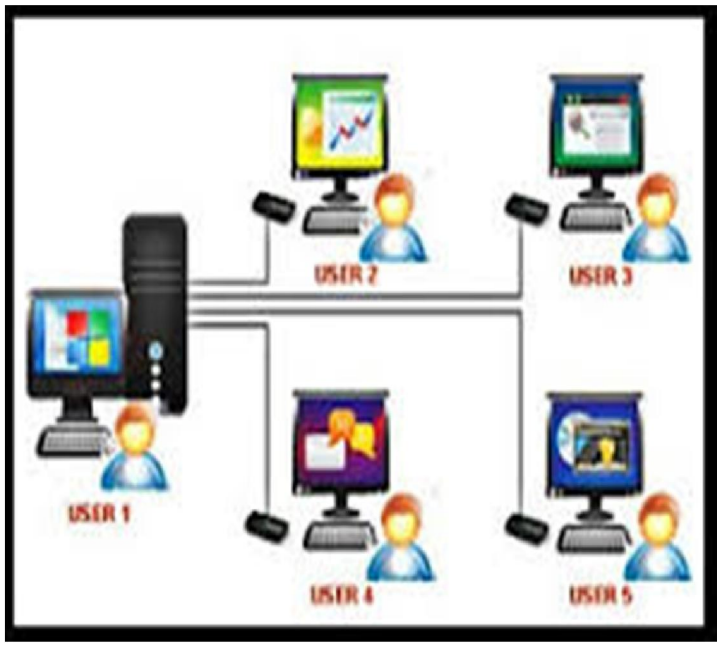
- **Multiprogramming (**Batch system) needed for efficiency
  - Single user program cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job

- **Multitasking** (Time sharing) 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

Batch System



Time sharing System

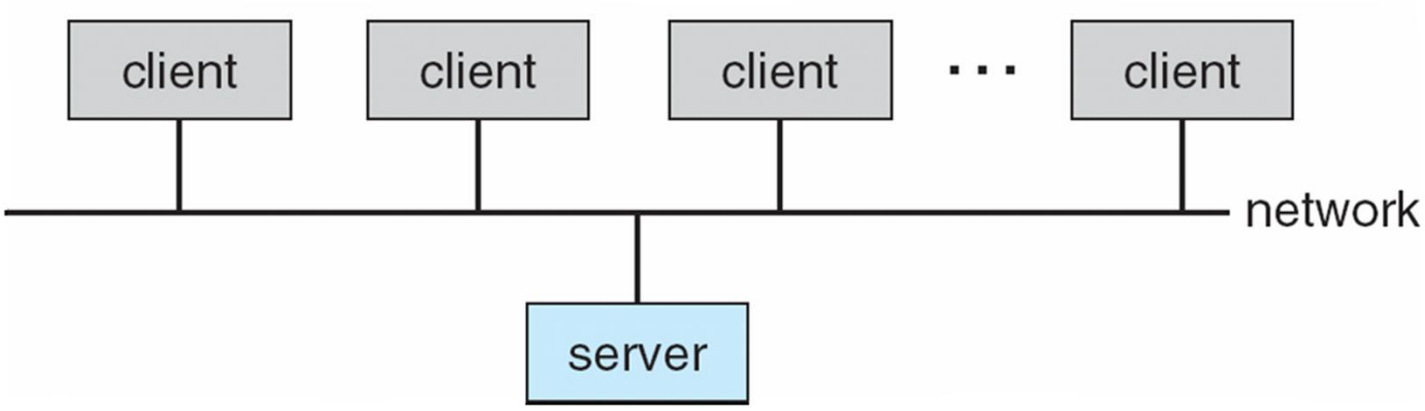


1.3.3 Computing Environments

- Types of Computing Environments
  - Traditional Computing
  - Client Server Computing
  - Peep-to-Peer Computing
  - Web Based Computing
- Traditional computing
  - Office environment & Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Client-Server Computing

- 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



- P2P Computing
  - 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**
- Web Based Computing
  - Web has become ubiquitous (found everywhere)
  - More devices becoming networked to allow web access

1.3.4 Special Purpose Systems

- The different special Purpose Systems are:
  - Real Time embedded Systems
  - Multimedia Systems
  - Hand Held Systems
- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
- Multimedia systems
  - Streams of data must be delivered according to time restrictions
- Handheld systems
  - PDAs, smart phones, limited CPU, memory, power
  - Reduced feature set OS, limited I/O

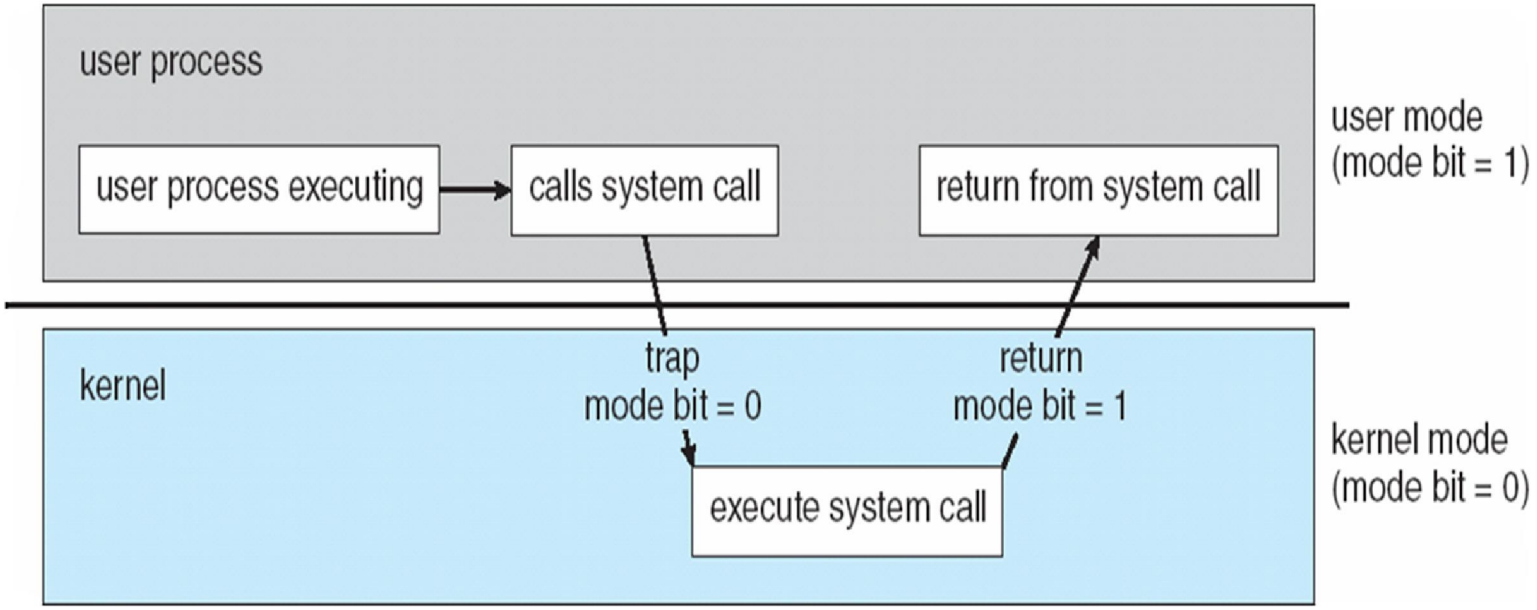
1.3.5 Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains.
- A network, in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality
- Different types of networks are:
  - **Local Area Network** (**LAN**)
  - **Wide Area Network** (**WAN**)
  - **Metropolitan Area Network** (**MAN**)
- Network Operating System provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

1.4 Operating System Operations

- Interrupt driven by hardware/software
- Software error or request creates **exception or trap**
  - Eg: Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- OS should provide Computer Protection against these sort of errors
- **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

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt



1.5 Functionalities of OS

- **The major functionalities performed by an operating system are:**
  - Process Management
  - Memory Management
  - Storage Management
  - Protection and Security

1.5.1 Process Management

- A program does nothing unless its instructions are executed by a CPU. A process is a unit of work in a system.
- A process needs resources – CPU time, memory, Files, I/O devices etc. The resources are allocated while it is running.
- Program is a ***passive entity***, process is an ***active entity***.
- 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

1.5.2 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 it is needed
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes move data into and out of memory
  - Allocating and de-allocating memory space as needed

1.5.3 Storage Management

- OS provides uniform, logical view of information storage
  - Different devices, same view
  - 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 directories
    - 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
  - Disk is slow, its I/O is often a bottleneck
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling

- **Caching**
- Caching is an important principle of computer systems. Information is normally kept in some storage system
- In addition, internal programmable registers, such as index registers, provide a high-speed cache for main memory.
- Because caches have limited size, cache management is an important design problem
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache

1.5.4 Protection & 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
  - includes 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**) 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 right

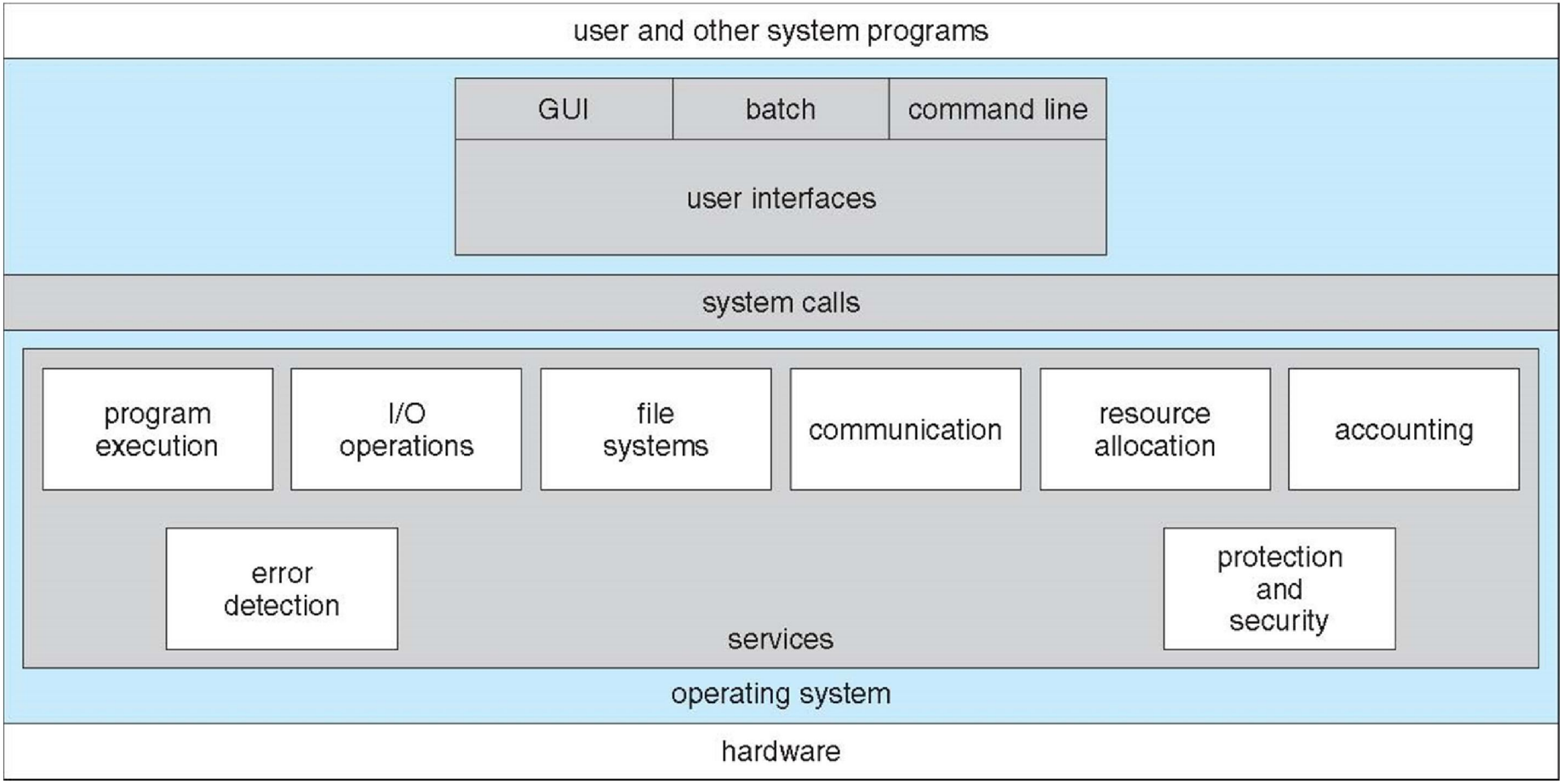
1.6 Operating System Services

- **Services helpful for the user**
  - **User interface** - Almost all operating systems have a user interface (UI).
    - Varies between **Command-Line (CLI)**, Batch Interface, **Graphics User Interface (GUI)**
  - **Program execution** - The system must be able to load a program into memory and to run that program, end 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
  - **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.

- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - 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
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

- **For Enhancing the efficiency of the system**
- **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
  - Many types of resources - 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
- **Accounting -** 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
  - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

Operating System Services



1.14 OS Interface

- Command Line Interface (CLI) or **command interpreter** allows direct command entry
  - Sometimes implemented in kernel, sometimes by systems program
  - Sometimes multiple flavors implemented – **shells**
  - 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

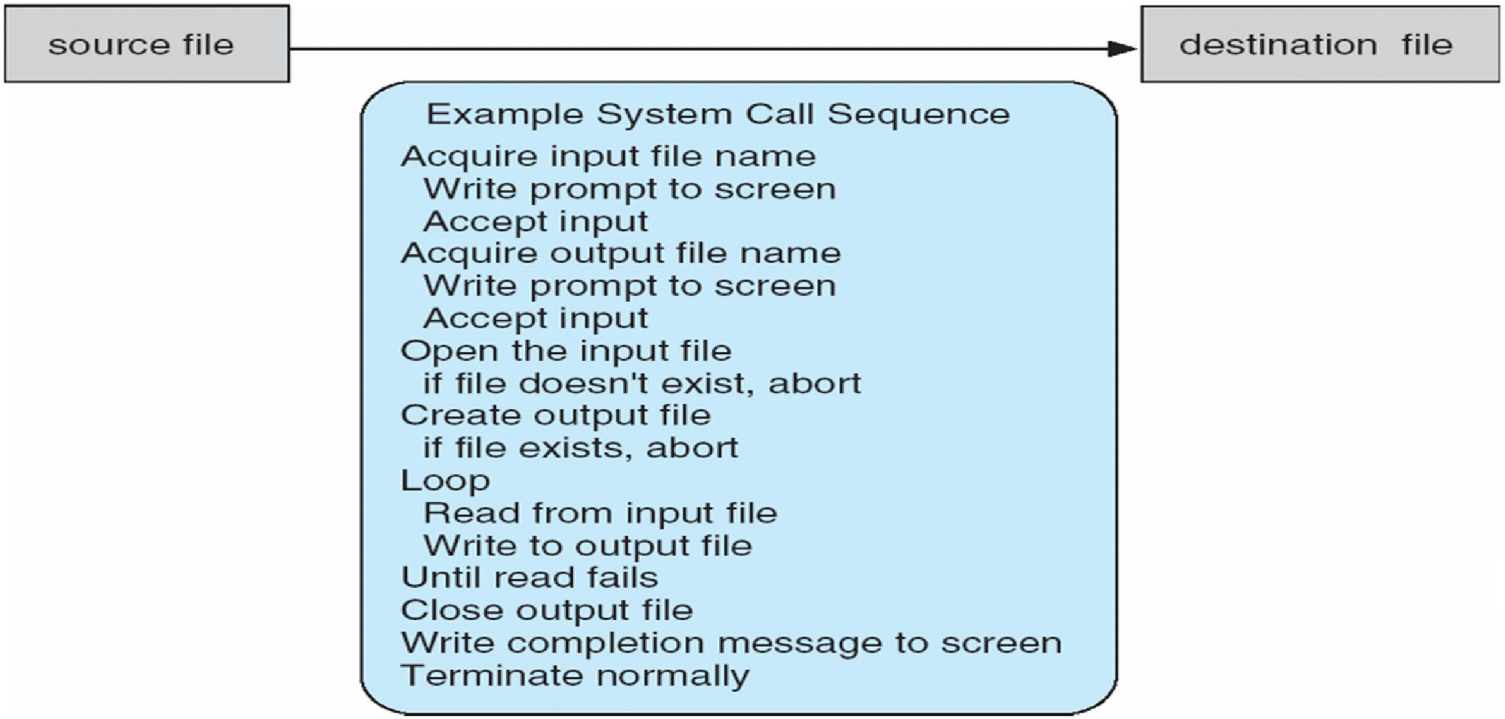
1.14 OS Interface

- 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)

1.7 System Calls

- The OS provides the services to the programming interfaces only through system calls
- These system calls are written in C/C++ and sometimes assembly language too. Now written in Java also.

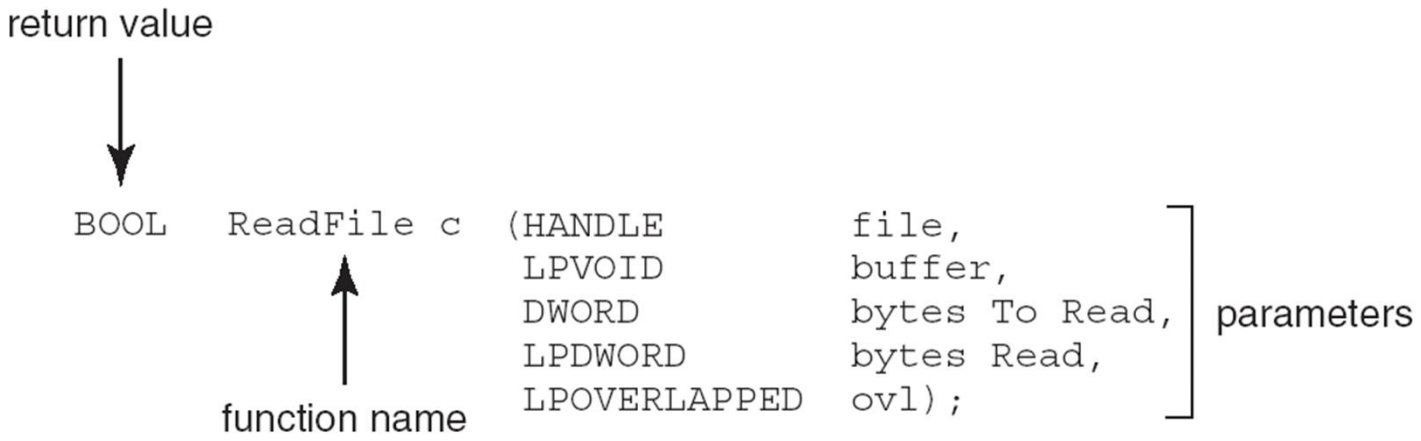


-Contd

- Systems Execute thousands of system calls per second.
- Most programmers never see this detail. Programmers design the program according to API. (Application Programming Interfaces)
- API are set of functions that are available to the application including the parameters to the functions and return values the programmer can expect
- The most common API are Win32 API, JavaAPI and POSIX API
- API defines a proper way for a developer to request services from another software/program
- The **System Call** is the request for running any program and for performing any operation on the system.

Standard API

- Consider the ReadFile() function in the
- Win32 API—a function for reading from a file



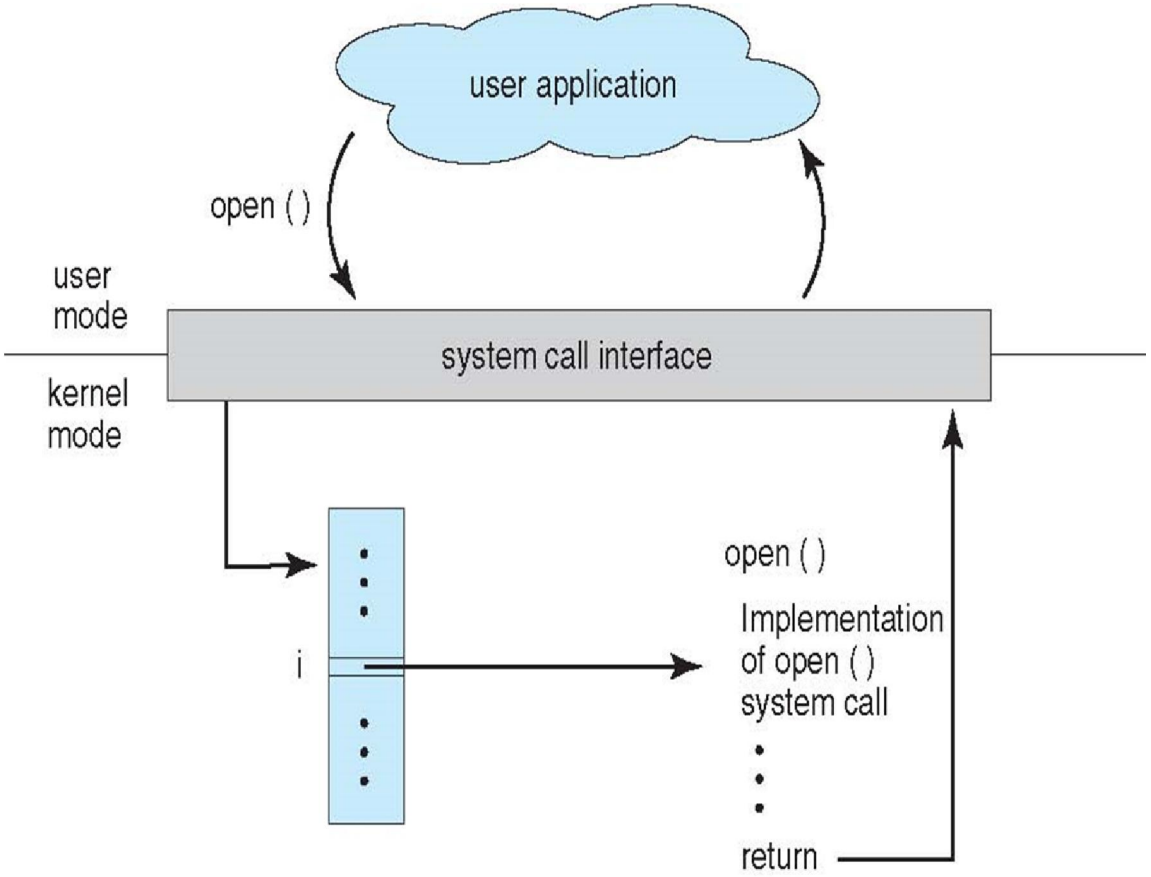
- A description of the parameters passed to ReadFile()
  - HANDLE file—the file to be read
  - LPVOID buffer—a buffer where the data will be read into and written from
  - DWORD bytesToRead—the number of bytes to be read into the buffer
  - LPDWORD bytesRead—the number of bytes read during the last read
  - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

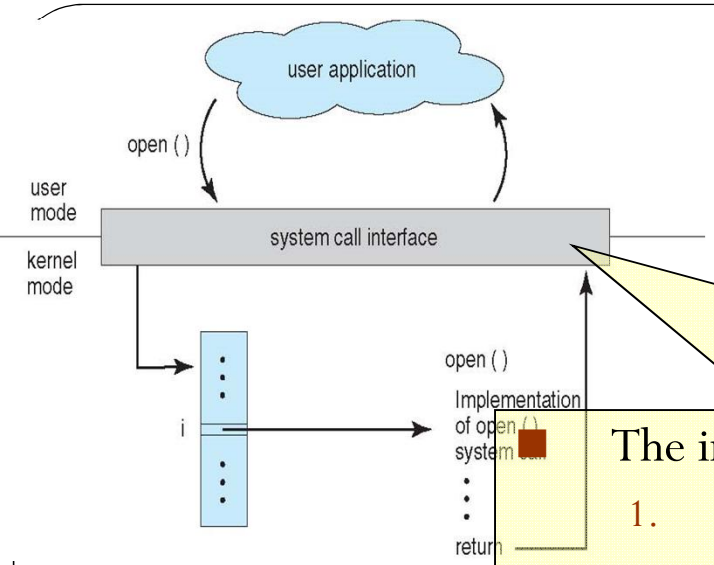
System Call Implementation

- A system library routine is called first
- It transforms the call to the system standard (native API) and traps to the kernel
- Control is taken by the kernel running in the system mode
- According to the service **"code"(indexed number)**, the interface invokes the responsible part of the Kernel
- Depending on the nature of the required service, the kernel may block the calling process
- After the call is finished, the calling process execution resumes obtaining the result (success/failure) as if an ordinary function was called

System Call Implementation

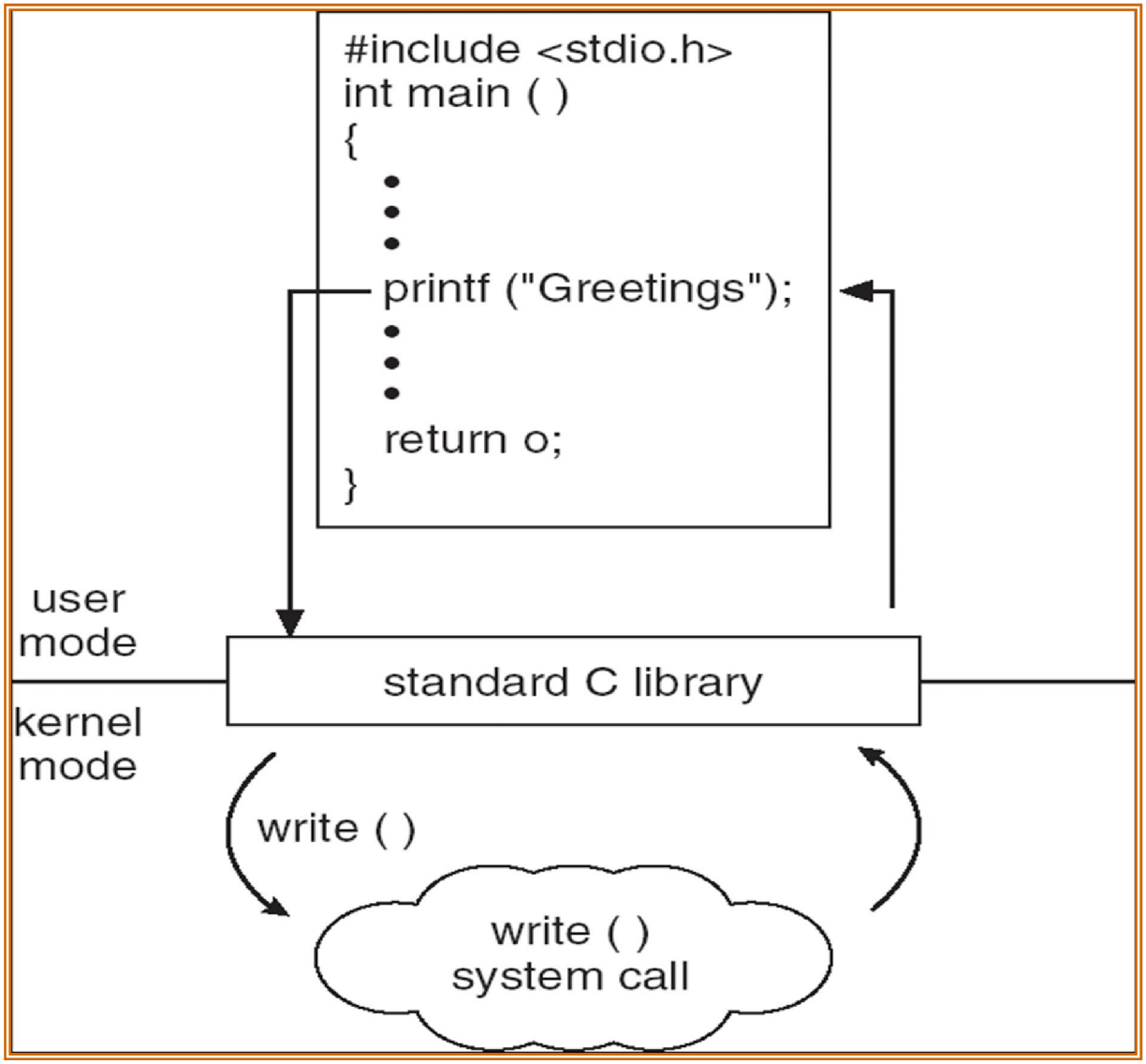
- Parameter values passed will be stored in registers, block/table of memory, maintained in a stack according to the design of OS.





The interface to the services provided by the OS has two parts:

1. Higher language interface – a part of a system library
   - Executes in user mode
   - Implemented to accept a standard procedure call (API)
   - Traps to the Part 2
2. Kernel part
   - Executes in system mode
   - Implements the required system service
   - May cause blocking the caller (forcing it to wait)
   - After completion returns back to Part 1 (may report the success o the call)


#include <stdio.h>
int main ( )
{
printf ("Greetings");
return o;
}
user mode
kernel mode
standard C library
write ( )
write ( )
system call

1.8 Types of System Calls

- System calls can be grouped roughly into six major categories:
  - Process Control
  - File Manipulation
  - Device Manipulation
  - Information Maintenance
  - Communication
  - Protection

Process Control

- create process, terminate process
- end, abort
- load, execute
- 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 file
- read, write, reposition
- get and set file attributes

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 and set process, file, or device attributes

Communication

- create, delete communication connection
- send, receive messages
- create and gain access to memory regions
- transfer status information
- attach and detach remote devices

Protection

- Control access to resources
- Get and set permissions
- Allow and deny user access

Some System Calls – in Windows and Unix

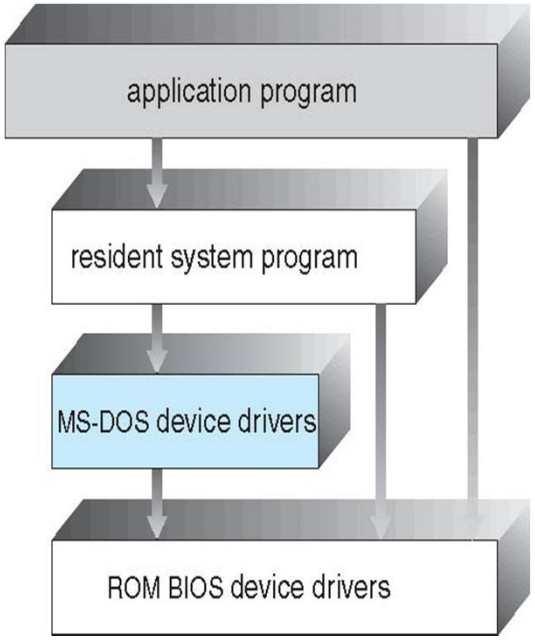
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

1.9 System Programs

- System programs provide a convenient environment for program development and execution.
- They can be divided into:
  - File manipulation
  - Status information
  - File Modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs

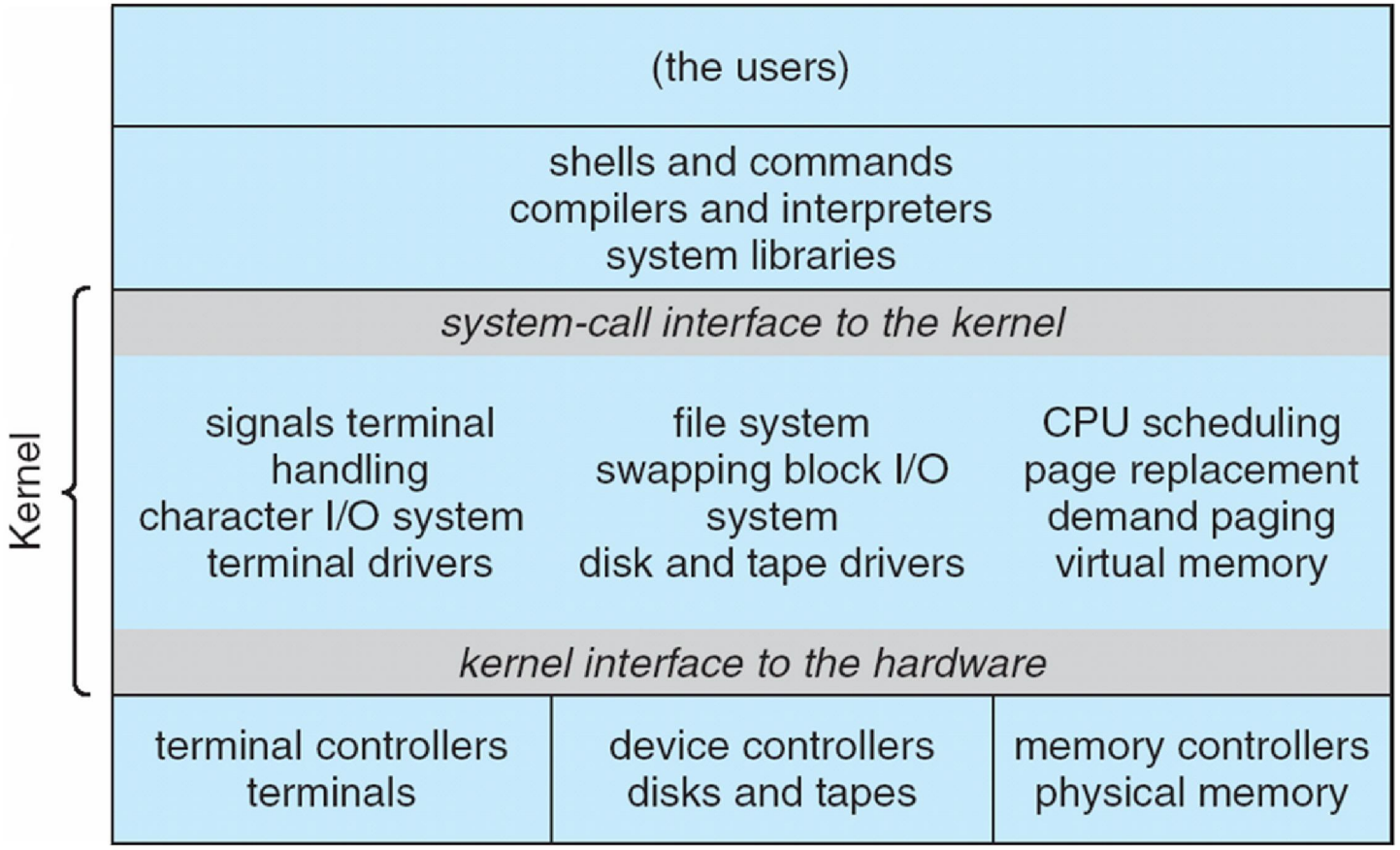
1.10 OS Structure

- **Simple Structure**
  - MS-DOS – written to provide the most functionality in the least space
    - Not divided into modules
    - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

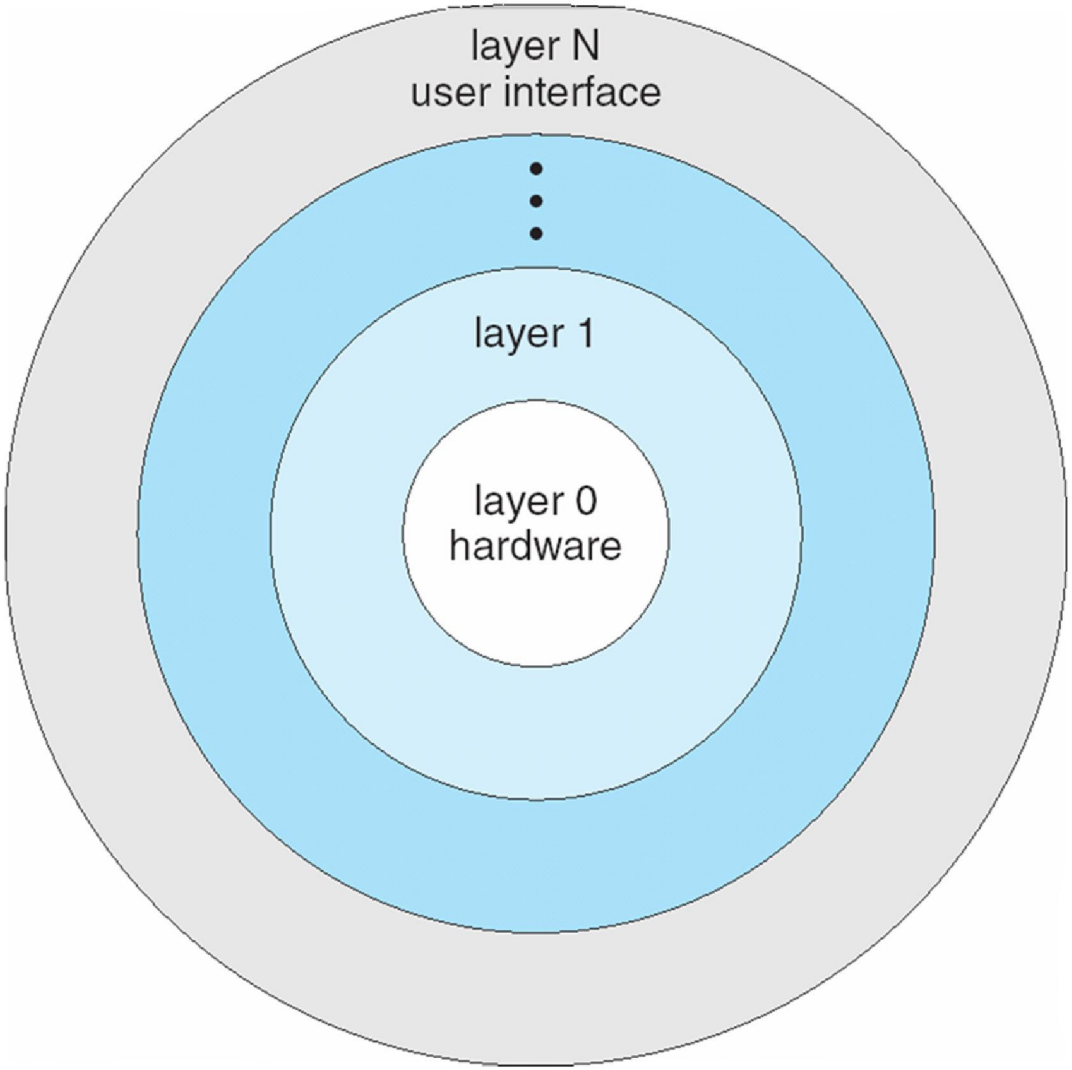


- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
  - Shell
  - Kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling,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

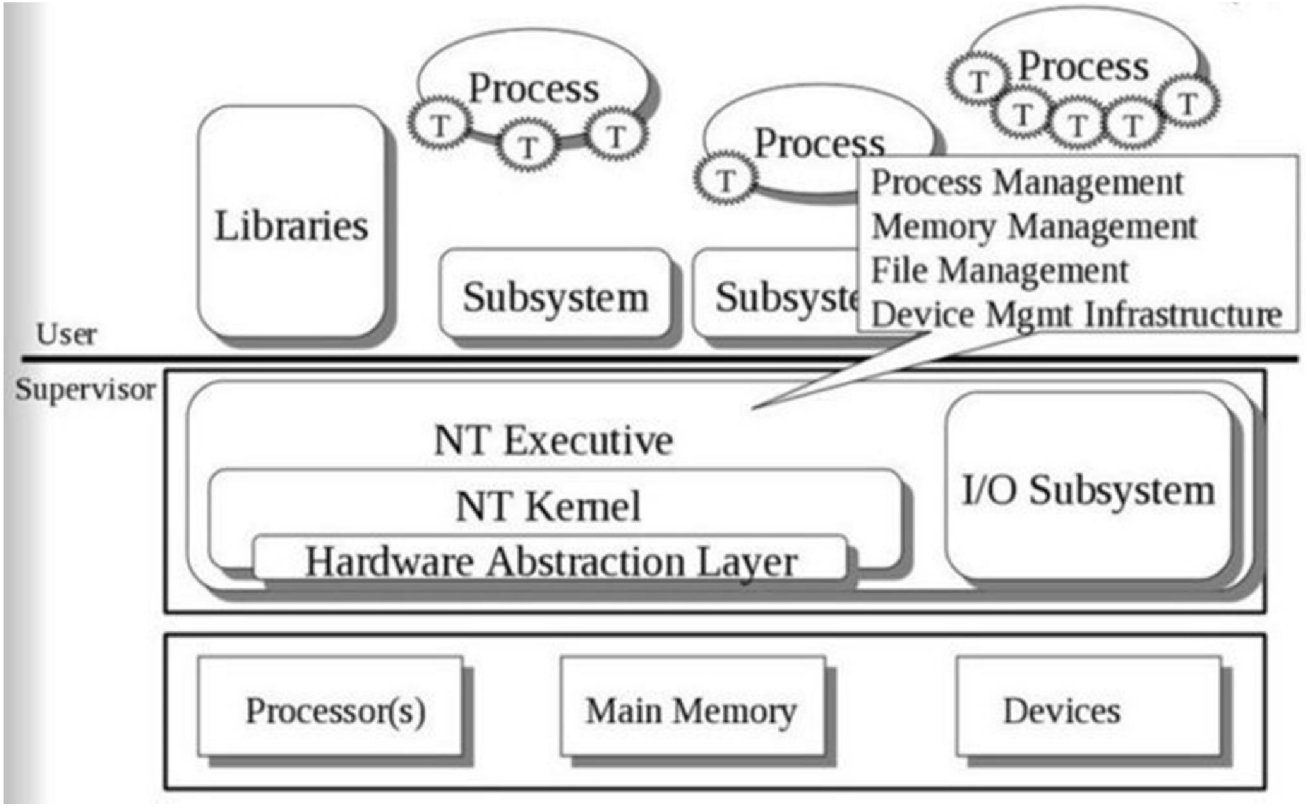
UNIX Structure



- **Layered Approach**
  - The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
  - With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers
  - Advantage: Simplicity of Construction and Debugging
  - Disadvantage: fails if appropriate planning not done. Each layer adds overhead to the lower layers


layer N
user interface
layer 1
layer 0
hardware

Windows NT Structure



- **Microkernel**
- First introduced by Mach in mid 1980's

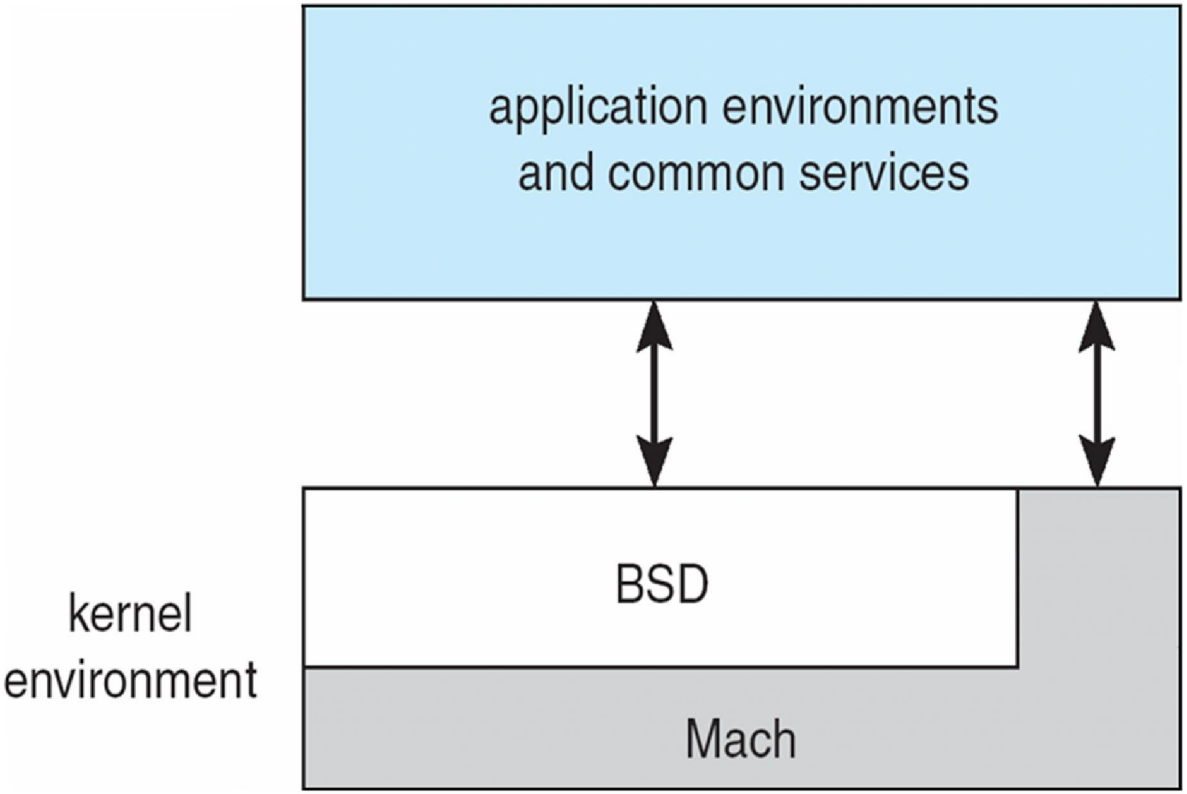
- Moves as much from the kernel into "*user*" space

- Communication takes place between user modules using message passing

- Benefits
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure

- Disadvantage
  - Performance overhead of user space to kernel space communication

Mac OS 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 Structure