Unit 1 - INTRODUCTION

Computer System - Elements and organization; Operating System Overview - Objectives and Functions - Evolution of Operating System; Operating System Structures – Operating System Services - User Operating System Interface - System Calls – System Programs - Design and Implementation - Structuring methods.

OPERATING SYSTEM: INTRODUCTION

- **An operating system is software that manages a computer's hardware. It provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.**
- **The purpose of an operating system is to**
  - **Make the computer system convenient to use**
  - **Use the computer hardware in an efficient manner**
  - **Execute user programs and make solving user problems easier**

1.1 ELEMENTS / COMPONENTS OF COMPUTER SYSTEM

A computer system can be divided into four components:

1. The hardware,
2. The operating system,
3. The application programs, and
4. The users.

The hardware: provides the basic computing resources for the system Eg: the central processing unit (CPU), the memory, and the input/output (I/O) devices.

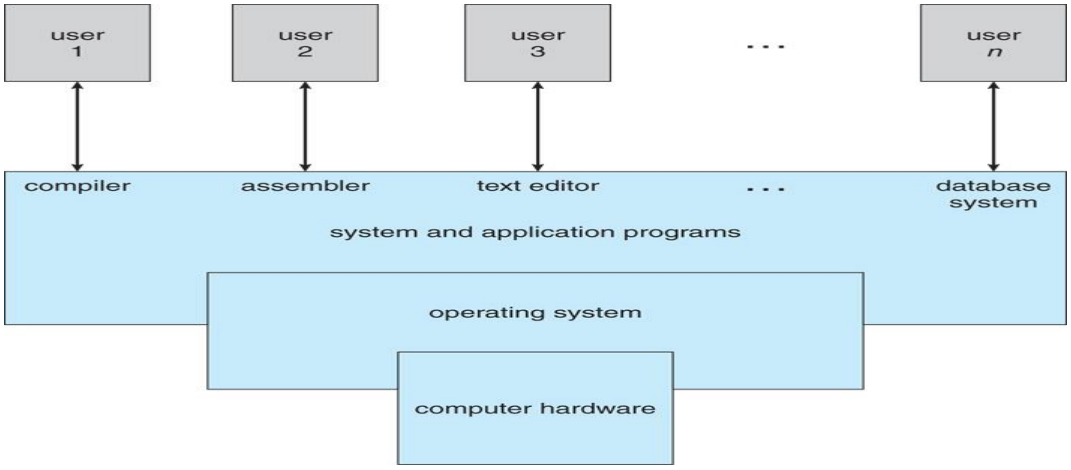


Fig : Abstract view of the components of the computer system

The application programs: define the ways in which these resources are used to solve users computing problems. Eg: word processors, spreadsheets, compilers, and Web browsers.

The operating system: controls the hardware and coordinates its use among the various application programs for the various users

The role of the operating system can be explored from 2 view points . They are:

1. User View 2. System View

i. User View

- The user's view of the computer varies according to the interface being used.
- Most computer users sit with Laptop/PC, consisting of a monitor, keyboard, and mouse. Such systems are designed for one user.
- Here, the operating system is designed for ease of use, less concentration on Performance and security.
- users interacting with mobile devices such as smartphones and tablets uses touch screen as the interface, where the user interacts with the system by pressing and swiping fingers across the screen.
- Many mobile devices also allow users to interact through a voice recognition interface, such as Apple's Siri.
- Some computers such as embedded computers are designed to run without user intervention. They are used in home devices and automobiles and have numeric keypads and may turn indicator lights on or off to show status.

ii. System View

- From the computer's point of view, we can view an operating system as a **resource allocator**.
- 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 manager of these resources.
- The operating system must decide how to allocate them to specific programs and operate the computer system efficiently and fairly.
- An operating system is a **control program**. A control program manages the execution of user programs to prevent errors and improper use of the computer.
- It is especially concerned with the operation and control of I/O devices.

Operating Systems

The operating system is the one program running at all times on the computer, usually called the **kernel**. Along with the kernel, there are two other types of programs:

- **System programs**, which are associated with the operating system but are not necessarily part of the kernel
- **Application programs**, which include all programs not associated with the operation of the system.

Today's OS for general purpose and Mobile includes

- **Middleware**, a set of software frameworks that provide additional services to application

For example, mobile operating systems, Apple's iOS and Google's Android.

The operating system always includes

- Kernel,
- Middleware frameworks
- System programs
- Application programs

1.2 COMPUTER SYSTEM- ORGANIZATION

In this section, we look at several parts of the structure of a computer system.

- **Computer System Operation**
- **Storage Structure**
- **I/O Structure**

1.2.1 Computer System Operation :

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common **bus** that provides access between components and shared memory.
- Each device controller is in charge of a specific type of device (for example, a disk drive, audio device, or graphics display).
- Depending on the controller, more than one device may be attached.
- 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.
- Typically, operating systems have a device driver for each device controller. The device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

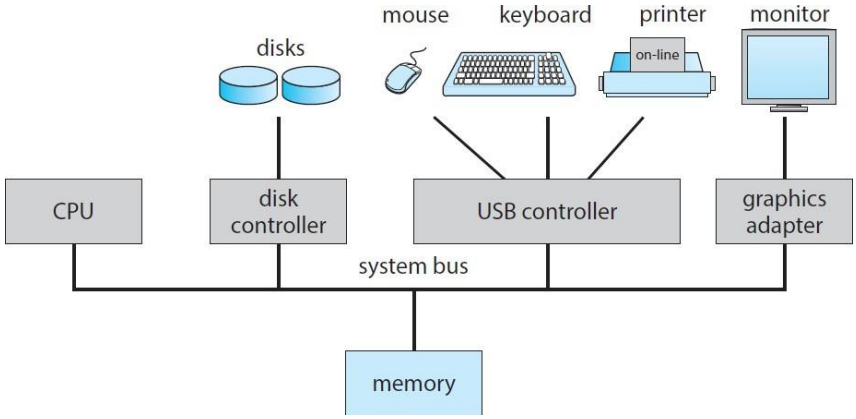


Fig: **A typical PC computer system**

- CPU moves data from/to main memory to/from local buffers.
- The occurrence of action in CPU is usually signaled by an interrupt
- **What is interrupt?**

  Interrupt is the event that can be caused by hardware or software that signals the processor to complete the ongoing instruction and immediately handle the Interrupt Service Routine (**ISR**). ISR contains the information for dealing with the interrupt.

Common Functions of Interrupts

- Interrupts are provided primarily as a way to improve processor utilization.
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**. Interrupt vector is an array that consists of interrupts and 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.
- **Interrupts are of three types: Hardware, software and trap**. Hardware Interrupts are generated by hardware devices to signal that they need some attention from the OS. Software Interrupts are generated by programs when they want to request a system call to be performed by the operating system. A trap or exception is a software-generated interrupt caused either by an error or a user request.
- An operating system is interrupt driven

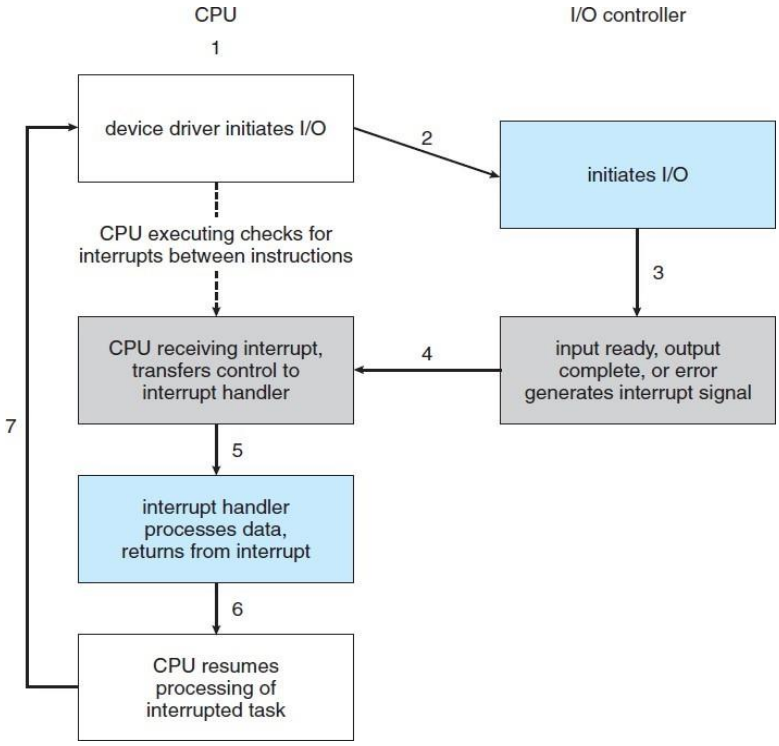


Fig : Interrupt-driven I/O cycle

CPUs have two interrupt request lines

i. Non Maskable Interrupts

Reserved for events such as unrecoverable memory errors.

ii. Maskable Interrupts

- The maskable interrupt is used by device controllers to request service.
- It can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted.

The following Figure illustrates the design of the interrupt vector for Intel processors. The events from 0 to 31, which are nonmaskable, are used to signal various error conditions. The events from 32 to 255, which are maskable, are used for purposes such as device-generated interrupts

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Table : Intel processor event-vector table.

1.2.1 STORAGE STRUCTURE

- The CPU can load instructions only from memory, so any programs must first be loaded into memory to run.
- General-purpose computers run most of their programs from rewritable memory, called main memory (also called random-access memory, or RAM).
  - Computers use other forms of memory as well. For example, the first program to run on computer power-on is a bootstrap program, which then loads the operating system. Since RAM is volatile, bootstrap program is not loaded in RAM, but it is loaded on EEPROM.
- The computer uses electrically erasable programmable read-only memory (EEPROM) and other forms of firmware storage that is infrequently written to and is non-volatile.
- All forms of memory provide an array of bytes. Each byte has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses.
- The load instruction moves a byte or word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory.

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**- it loses its contents when power is turned off or otherwise lost.
  - Typically random-access memory in the form of **Dynamic Random-access Memory (DRAM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Hard Disk Drive – 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
- Non-volatile memory (NVM) devices– faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular as capacity and performance increases, price drops

Hierarchy of Storage-device

- The wide variety of storage systems can be organized in a hierarchy according to storage capacity and access time.
- The five hierarchies in a system's memory are register, cache memory, main memory, magnetic disc, and magnetic tape.
- The top four levels of memory in the figure are constructed using semiconductor memory, which consists of semiconductor-based electronic circuits.
- NVM devices, at the fourth level, have several variants but are faster than hard disks. The most common form of NVM device is flash memory, which is popular in mobile devices such as smartphones and tablets. Increasingly, flash memory is being used for long-term storage on laptops, desktops, and servers as well.

- External or Secondary Memory
  - It consists of Magnetic Tape, Optical Disk, Magnetic Disk, i.e., it includes peripheral storage devices that are accessible by the system's processor via I/O Module.
- Internal Memory or Primary Memory
  - It consists of CPU registers, Cache Memory, and Main Memory. It is accessible directly by the processor.

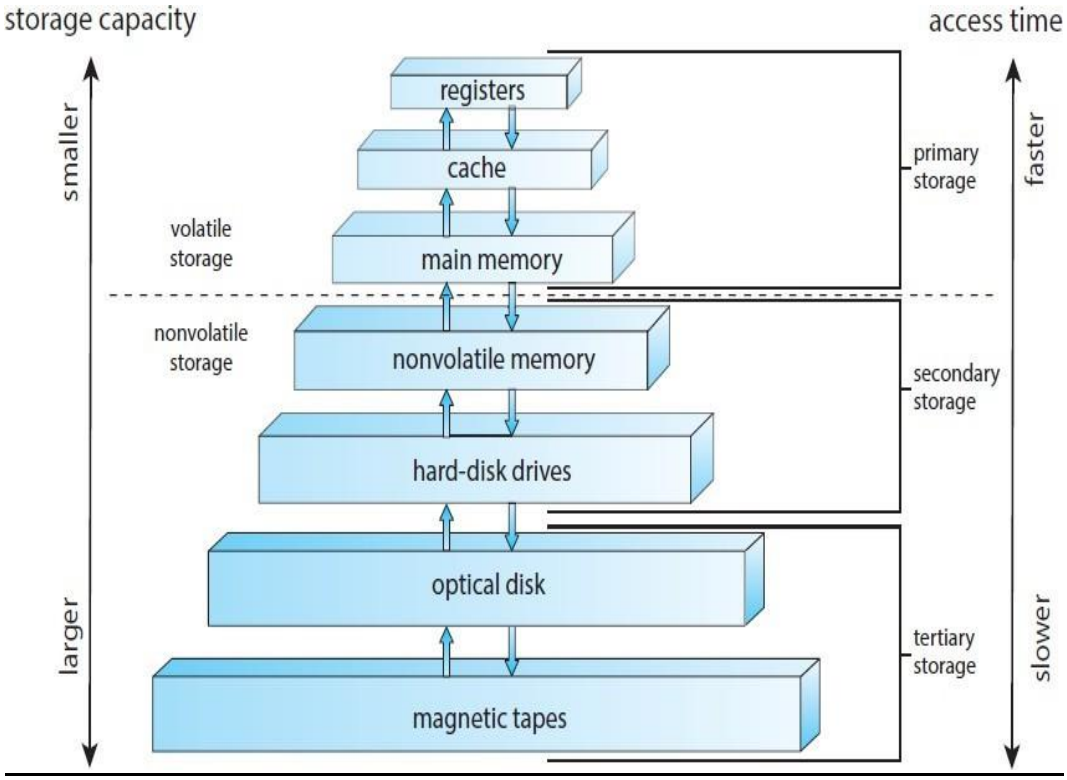


Fig : Storage Hiearchy

1.2.3 I/O STRUCTURE

- A large portion of operating system code is dedicated to managing I/O, because of its importance to the reliability and performance of a system and because of the varying nature of the devices.
- All the I/O devices are connected to each other by using common bus. CPU and main memory is also connected with this bus.
- Every I/O device uses a device controller to connect it to the computer's address and data bus.
- Various types of controllers are used in the computer system.
- Depending on the controller, more than one device can be attached. For eg , Small Computer System Interface (SCSI) controller can handle seven or more devices. Each device controller have its own buffer.
- Device controller manage the data transfer between peripheral device and its controller.
- There is a Device driver for every device controller.

➢ **I/O operation steps**

1. Device driver loads the registers within the device controller.
2. Device controller takes action according to the data loaded into the register.
3. Data is transferred from device to its local buffer with the help of device controller.
4. After completion of data transfer, the device controller sends an interrupt signal to device driver about data transfer completion operation.
5. The device driver then return control goes to operating system.

➢ This type of interrupt driven I/O is fine for small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O. To solve this problem, Direct memory Access (DMA) is used .

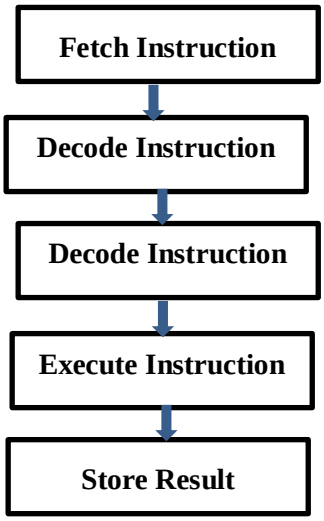
INSTRUCTION EXECUTION :

➢ The program which is to be executed is a set of instructions which are stored in memory. The Central Processing Unit (CPU) executes the instructions of the program to complete a task.

➢ The major responsibility of the instruction execution is with the CPU. The instruction execution takes place in the CPU registers.

➢ A special register contains the address of the instruction. The CPU "fetches" the instruction from memory at that address.

➢ The CPU "decodes" the instruction to figure out what to do. The CPU "fetches" any data (operands) needed by the instruction, from memory or registers.



- The CPU "executes" the operation specified by the instruction on this data. The CPU "stores" any results into a register or memory.
- The register used for instruction execution are as follows:

1. **Memory Address Register (MAR):** It specifies the address of memory location from which data or instruction is to be accessed (for read and write operation).

2. **Program Counter (PC):** It keeps track of the instruction which is to be executed next, after the execution of an on-going instruction.

3. **Instruction Register (IR):** Here the instructions are loaded before their execution.

1.3 OPERATING SYSTEM OVERVIEW

A computer system is a collection of hardware and software designed to provide an effective tool for computation.

Hardware generally refers to the electrical, mechanical and electronic parts that make up the computer (i.e Internal Architecture of the computer). However, the hardware is sophisticated, it cannot function properly without a proper driver which can drive it and bring it to the best advantage. For example, a car, even though sophisticated in features, it cannot function independently without being properly driven by an efficient driver.

Similarly, the hardware though technologically innovative and which presents enhanced features, which needs set of programs to bring it to operation. So, the driver that drives the hardware is software. *Software* refers to the set of programs written to provide services to the system. It gives life and meaning to the hardware and bring it to the operational level, which otherwise is a useless piece of metal.

Software is basically of two types

- Application software
- System software

Application software: set of programs written for a specific area of application.

Eg. Word processors, spreadsheets and database, etc.

System Software: set of programs written from the point of view of the machine *i.e., System software provides environment for execution of application software.*

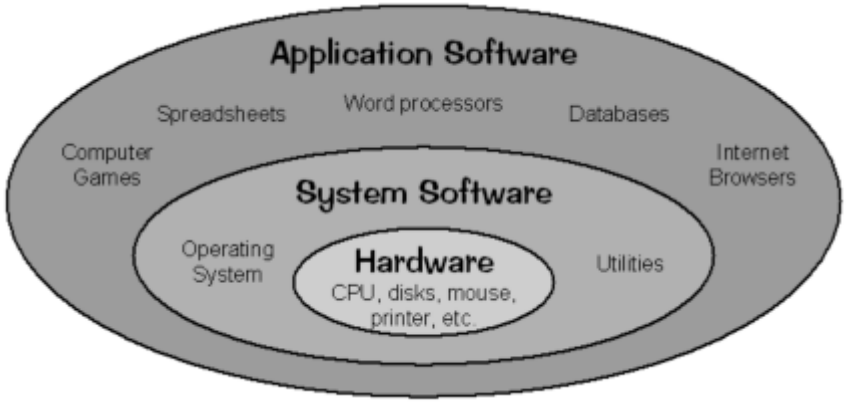


Fig 1.8 Computer System Components

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

Definition of an Operating System

An OS is defined as a System program that controls and coordinates the execution of the programs and acts as an interface between application user and the computer 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.

1.3.1 Objectives and Functions of OS

The OS has 3 main objectives

- Execute user programs and make solving user problems ***easier***
- Make the computer system ***convenient*** to use
- Use the computer hardware in an ***efficient*** manner

The OS performs various functions

1. ***Booting***: It is a process of starting the computer operating system starts the computer to work. It checks the computer and makes it ready to work.

2. ***Memory Management***: It is also an important function of operating system. The memory cannot be managed without operating system. Different programs and data execute in memory at one time. if there is no operating system, the programs may mix with each other. The system will not work properly.

3. ***Loading and Execution:*** A program is loaded in the memory before it can be executed. Operating system provides the facility to load programs in memory easily and then execute it.

4. ***Data security:*** Data is an important part of computer system. The operating system protects the data stored on the computer from illegal use, modification or deletion.

5. ***Disk Management***: Operating system manages the disk space. It manages the stored files and folders in a proper way.

6. ***Process Management:*** CPU can perform one task at one time. if there are many tasks, operating system decides which task should get the CPU.

7. ***Device Controlling:*** operating system also controls all devices attached to computer. The hardware devices are controlled with the help of small software called device drivers.

8. ***Providing interface:*** It is used in order that user interface acts with a computer mutually. User interface controls how you input data and instruction and how information is displayed on screen.
The operating system offers two types of the interface to the user:

- ✓ ***Graphical-line interface****:* It interacts with of visual environment to communicate with the computer. It uses windows, icons, menus and other graphical objects to issues commands.
- ✓ ***Command-line interface:*** it provides an interface to communicate with the computer by typing commands.

1.4 EVOLUTION OF OPERATING SYSTEM (COMPUTER SYSTEM ARCHITECTURE)

The evolution of operating system is explained at various stages:

i) Serial Processing
ii) Simple Batch Systems
iii) Multiprogramming Systems.
iv) Time sharing Systems
v) Multiprocessors Systems
vi) Real Time Systems
vii) Clustered Systems
viii) Distributed Systems
ix) Hand Held Systems

i) Early systems/ Serial processing (mid 1940s to mid 1950s)

Before 1950, the programmers directly interact with the hardware, no operating system was found at that time. If the programmer wishes to execute a program on those days, the following steps are necessary.

- Type the program on the punched cards.
- Feed the punched cards to a card reader.
- Submit to the computing machines, if there are any errors, the error was indicated by the lights.
- The programmer examines the register and main memory to identify the cause of an error.
- Take outputs on the printers.
- Then the programmer is ready for the next program.

ii) SIMPLE BATCH SYSTEMS (1960s)

- Before 1960, the computer is located in three different rooms, one room for card reader, another for program execution, and third for printing the result.
- This problem is solved by using batch processing.
- Batch operating system is the operating system which analyses the input and groups them into batches i.e. the same type of jobs batch together and execute at a time.
- The users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.
- The operator then sorts programs into batches with similar requirements and submit each group of job into computer.
- Jobs are processed in the order of submission i.e first come first served fashion
- OS keeps a number a jobs in memory and executes them without any manual information
- When job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.
- Batch system process a collection of jobs, called a batch. Batch is a sequence of user jobs. Job is a predefined sequence of commands, programs and data that are combined into single unit
- Example of Batch Operating System is as follows.
  - DOS (Disk operating system).
  - IBM OS/2.

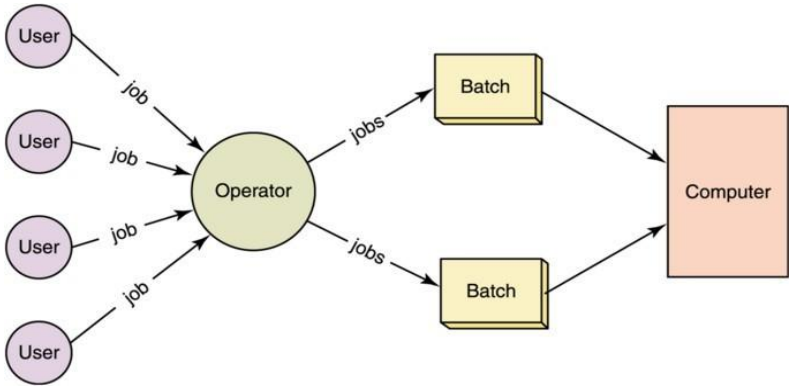


Fig : Batch Systems

Advantages of Batch System

- The overall time taken by the system to execute all the programs will be reduced.
- Can be shared between multiple users.

Disadvantages of Batch System

- Lack of interaction between the user and job.
- Difficult to provide the desired priority.
- Difficult to debug program

iii) MULTIPROGRAMMING OPERATING SYSTEMS (1970s)

- Execution of multiple jobs in an interleaved manner is known as multiprogramming
- In multiprogramming system, when one program is waiting for I/O transfer; there is another program ready to utilize the CPU. So it is possible for several jobs to share the time of the CPU.
- Jobs are organized in such a way that CPU always has one to execute. This increases CPU utilization by minimizing CPU idle time.
- Operating system loads multiple jobs into memory form job pool on the disk.
- The operating system then picks one job form memory and start executing it.
- When job need to perform some other activity, such as I/O, operating system simply picks up another job and starts executing it. When job needs to perform some other activity, the operating system switches to next job and so on. When I/O activity is finished it gets the CPU back.
- This type of execution of multiple processes is called concurrent execution.

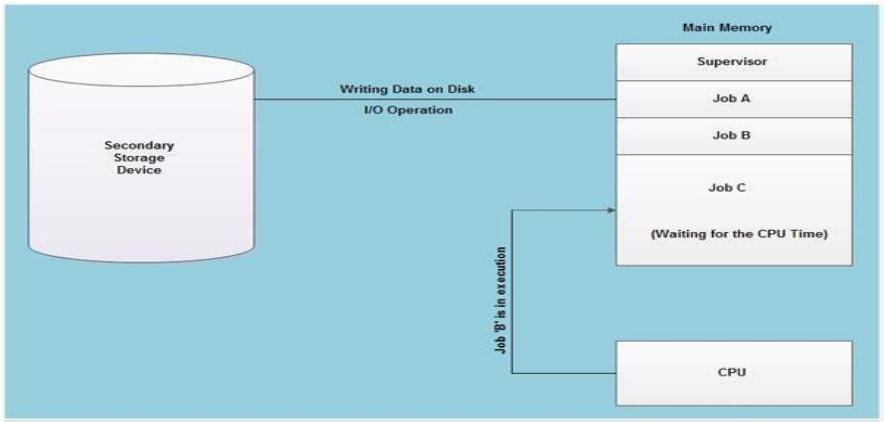


Fig: Working of Multiprogramming OS

➢ As shown in fig above, at the particular situation, job' A' is not utilizing the CPU time

because it is busy in I/ 0 operations. Hence the CPU becomes busy to execute the job 'B'. Another job C is waiting for the CPU for getting its execution time. So, in this state the CPU will never be idle and utilizes maximum of its time. A program in execution is called a "Process", "Job" or a "Task". The concurrent execution of programs improves the utilization of system resources and enhances the system throughput as compared to batch and serial processing.

Advantages :

- ➢ High and efficient CPU utilization.
- ➢ User feels that many programs are allotted CPU almost simultaneously.

Disadvantages :

- ➢ CPU scheduling is required.
- ➢ To accommodate many jobs in memory, memory management is required.

iv) TIME SHARING SYSTEMS (1970s)

- Time sharing refers to allocation of computer resources in time slots to several programs simultaneously.
- Users share these computer resources simultaneously.
- Time sharing systems uses CPU scheduling and multiprogramming.
- As the system switches rapidly from one user to other, a short time slot is given to each user for their execution.
- CPU time is divided among all the users on a scheduling basics. Round robin algorithm is used in Time sharing systems.
- The OS allocates a set of time to each user. When the time expires it passes control to the next user on the system.
- The time allowed is extremely small and the users are given an impression that they have their own CPU and they are the sole owner of the CPU.
- This shortest period of time during a user gets attention of the CPU is known as **time slice or a quantum**.

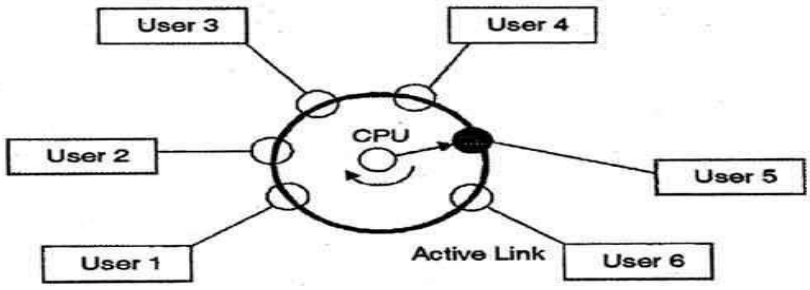


Fig Time Sharing Systems

- In the above figure user 5 is active but user 1, user 2, user 3, and user 4 are in waiting state whereas user 6 is in ready status.
- As soon as the time slice of user 5 is completed, the control moves on to the next ready user i.e., user 6. In this state user 2, user 3, user 4, and user 5 are in waiting state and user 1 is in ready state. The process continues in the same way and so on.

Advantages:

1. Since equal time quantum is given to each process, so each process gets equal opportunity to execute.
2. The CPU will be busy in most of the cases.

Disadvantage:

1. Process having higher priority will not get the chance to be executed first because the equal opportunity is given to each process.

v) MULTIPROCESSOR SYSTEMS (1980s)

- All the processor share common bus, clock, memory and peripheral devices.
- More Reliable i.e. if one processor fails then the remaining processor will share the work of the failed processor, thus preventing the failure of entire system
- Multiprocessor system is of two types
  - Symmetric multiprocessing
  - Asymmetric multiprocessing

Symmetric multiprocessor system (SMP)

- Symmetric multiprocessor system (SMP) is a multiprocessor system with a centralized shared memory called main memory operating under a single operating system with two or more processors
- Each processor shares a single copy of operating system.
- There is no Master-Slave relationship between processor
- Parallelism can be achieved by using multiprocessor operating systems

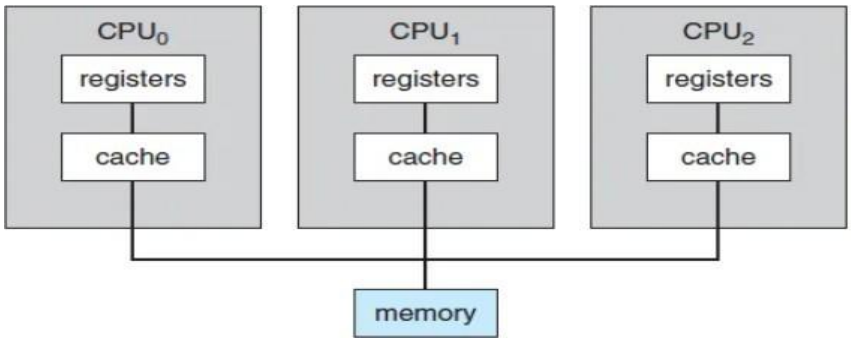


Fig : Symmetric Multiprocessor System

Asymmetric Multiprocessor

- Asymmetric Multiprocessing (AMP) allows a multiprocessor system to run multiple Operating Systems (OS) that are independent of each other. In other words, each CPU has its own private memory space, which contains the OS and the applications that are to run on that CPU.
- There is one master processor and the remaining are slave processors. Master processor controls the whole operations.
- Slave processor either wait for the master instructions to perform any task or have predefined tasks. This scheme defines master-slave relationship.
- Difficult to implement.
- Failure of the master processor will make entire system to fail.

vi) REAL TIME SYSTEMS –RTOS (1980s)

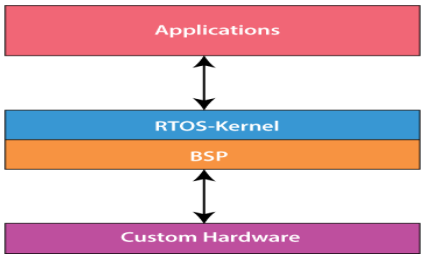
- When user gives input to a system, it must process within the time limit and the result is sent back. Such systems are called Real time systems
- Real time system fails if it does not give result within the time limits.
- Time constraint is the key parameter in real time systems. It controls autonomous system such as robots, satellites, air traffic control etc.
- Real time systems are of two types.
  - **Hard Real Time**
  - **Soft Real Time**

Hard Real Time System

- Critical task is completed within the time limit in hard real time system
- Hard real time system is purely deterministic and time constraint system. For example if the users expected the output for the given input in 10sec, then system should process the input data and give the output exactly by 10^{th} second.
- In the hard real time system meeting the deadline is very important if deadline is not met the system performance will fail.

Soft real time system

- In soft real time system, the meeting of deadline is not compulsory for every time for every task, but process should get processed and give the result. Even the soft real time systems cannot miss the deadline for every task or process. According to the priority it should meet the deadline or can miss the deadline.
- The best example for soft real time system is video surveillance (cctv), video player, virtual reality, etc



vii) CLUSTERED SYSTEMS (1980s)

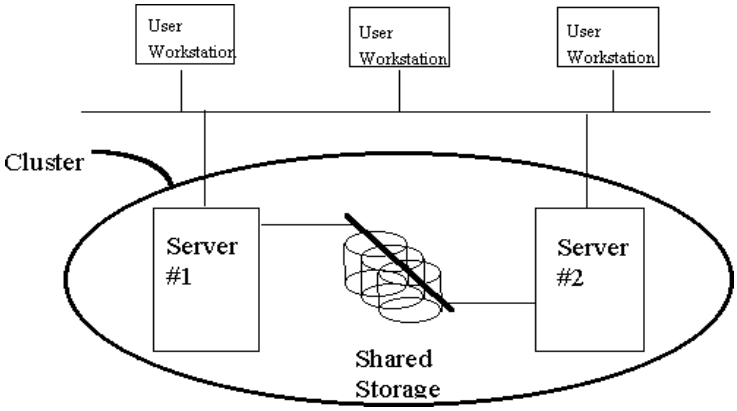
- Clustered system is a group of computer systems connected with a high speed communication link.
- Each computer system has its own memory and peripheral devices.
- Clustering provides high availability
- A clustered system uses multiple CPUs to complete a task.
- A layer of cluster software runs on cluster nodes. Each node can monitor one or more nodes over the LAN.
- The monitor machine can fail in some cases. The monitoring machine can take ownership of its storage. The monitoring machine can also restart applications that were running on the failed machine.
- The clustered system can be of the following forms:

Asymmetric Clustering:

- ❖ In this form, one machine is in hot standby mode and other machine is running the application. The hot standby machine performs nothing. It only monitors the server. It becomes the active server if the server fails.

Symmetric Clustering:

- ❖ In this mode, two or more machines run the applications. They also monitor each other at the same time. This mode is more efficient because it uses all available machines. It can be used only if multiple applications are available to be executed.



viii) DISTRIBUTED SYSTEMS (1990s)

- Distributed Operating System is a model where distributed applications are running on multiple computers linked by communications.
- A distributed operating system is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network.
- This system looks to its users like an ordinary centralized operating system but runs on multiple, independent central processing units (CPUs).

Advantages of distributed OS

- ✓ Resource sharing
  - ❖ Sharing of software resources such as software libraries, database and hardware resources such as hard disks, printers, CDROM.
- ✓ Higher reliability
  - ❖ Reliability refers to degree of tolerance against errors and component failures
- ✓ Availability
  - ❖ Fraction of time for which a system is available for use
  - ❖ Availability of hard disk can be increased by having multiple hard disks located at different sites. If one hard disk fails the program can use some other hard disk.
- ✓ Better price performance ratio
  - ❖ Reduction in price of microprocessor and increasing computing power gives good price-performance ratio
- ✓ Shortest response time and higher throughput
- ✓ Incremental growth

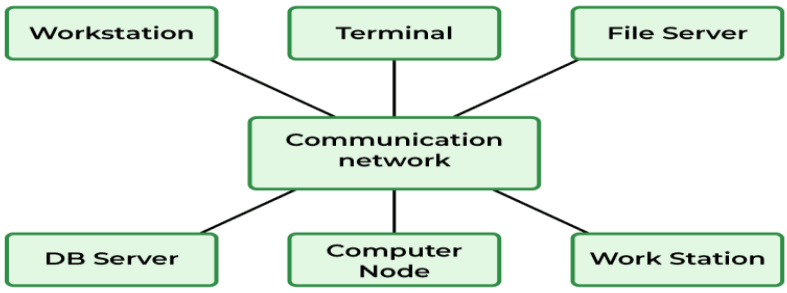


Fig : Distributed System

Advantages:

- Failure of one system will not affect the other network communication
- Since resources are being shared, computation is highly fast and durable

Disadvantages:

- Failure of the main network will stop the entire communication
- These types of systems are very expensive.

ix) HAND HELD SYSTEMS : (2000s)

- Very small, lightweight device (such as the Palm Pilot) which provides functionality similar to that of a laptop computer.
- Handheld system contains slow processor, small display and less memory.
- Physical size is the main constraint for handheld system.
- Operating system and application must manage memory more efficiently.
- Memory manager takes care of allocating and de-allocating of memory. Makes memory immediately free when task is completed
- Handheld system operates on battery and battery power is limited.
- To increase processor speed, battery capacity must be increased or recharged more frequently.
- Operating system used now a days are Blackberry, Android, Windows.

Note : Whenever several users share the system simultaneously, the situation can result in various security problems. Such problems are privacy, integrity and denial of service.
a) Privacy: One user can read the private data of another user.
b) Integrity: One user can corrupt the private data of another user.
c) Denial of Service: One user can prevent another user from getting anything done.

Difference between Time sharing OS and Multiprogramming OS

Sl.No	Time sharing OS	Multiprogramming OS
1	Processor time is shared with multiple users	Processor and memory underutilization problem is resolved and multiple programs runs on CPU.
2	It is based on the concept of time sharing	It is based on the concept of context switching
3	The idea is to allow multiple processes to run simultaneously via time sharing	The idea is to reduce the CPU idle time for as long as possible.
4	It takes less time to execute	It takes more time
5	Principle objective is to minimize response time.	Principle objective is to maximize processor use.

Difference between Symmetric and Asymmetric Multiprocessor

Symmetric Multiprocessor	Asymmetric Multiprocessor
Symmetric multiprocessing treats all processors are equals. An I/O can be processed on any CPU.	Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves.
Symmetric multiprocessing is easier to implement in operating systems.	Asymmetric multiprocessing is difficult to implement.

Single OS manages all processor cores simultaneously	Separate OS, or separate copy of same OS manage each core.
Master – slave concept is not used	It uses concept of master – slave concept.
The processors communicate with each other through shared memory	Shared memory is not used for communication.

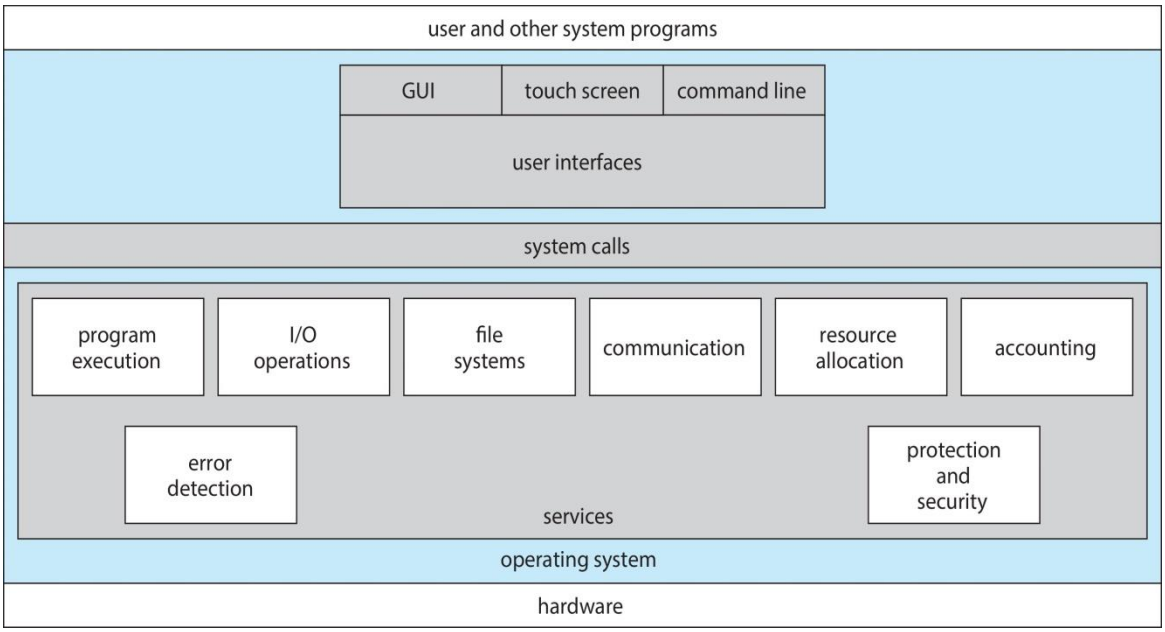
1.5 OPERATING SYSTEM SERVICES

An Operating System provides services to both the users and to the programs.

- ➢ It provides an environment for the execution of programs.
- ➢ It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system

- ❖ **Program execution**
- ❖ **I/O operations**
- ❖ **File System manipulation**
- ❖ **Communication**
- ❖ **Error Detection**
- ❖ **Resource Allocation**
- ❖ **Protection**
- ❖ **User interface**



Program execution

- ➢ The system must be able to load a program into memory and to run that program.
- ➢ The program must be able to end its execution, either normally or abnormally.

User interface.

- ➢ All operating systems have a user interface (UI).
- ➢ **A Graphical User Interface (GUI)**, this interface is a window system with a mouse that serves as a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text.
- ➢ Mobile systems such as phones and tablets provide a **Touch-Screen interface**, enabling users to slide their fingers across the screen or press buttons on the screen to select choices.

- **Command-Line Interface (CLI),** which uses text commands and a method for entering them. Some systems provide two or all three of these variations.

I/O Operation

- A running program may require I/O, which may involve a file or an I/O device.
- For specific devices, special functions may be desired (such as reading from a network interface or writing to a file system).
- 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

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error Detection

- Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware.
- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Allocation

- In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job.
- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection and security

- The OS ensures that all access to system resources is controlled.
- Security of the system from outsiders is done by means of a password, to gain access to system resources.
- It extends to defending external I/O devices, including network adapters, from invalid access attempts and recording all such connections for detection of break-ins.

Accounting (Logging)

- To keep track of which programs use how much and what kinds of computer resources.
- This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics.
- Usage statistics may be a valuable tool for system administrators who wish to reconfigure the system to improve computing services.

1.6 USER AND OPERATING SYSTEM INTERFACE

The user and operating system are connected with each other with the help of interface.

There are several ways to interface with the operating system.

Command line interface / Command Interpreters

- The command-line interface is an interface whenever the user needs to have different commands regarding the input and output and then a task is performed so this is called the command-line argument and it is used to execute the output and create, delete, list, print, copy, execute etc.
- Allows users to directly enter commands to be performed by the operating system.
- The command interpreter is a special program that is running when a process is initiated or when a user first logs on (on interactive systems).
- On systems with multiple command interpreters are known as shells.
- The main function of the command interpreter is to get and execute the next user-specified command.
- In one approach, the command interpreter itself contains the code to execute the command,
- An alternative approach, implements most commands through system programs.
- Command Line Interface advantages
  - Controls OS or application
  - faster management
  - Ability to store scripts which helps in automating regular tasks.
- Command Line Interface disadvantages:
  - The steeper learning curve is associated with memorizing commands and a complex syntax.
  - Different commands are used in different shells.

Graphical user interface

- Rather than entering commands directly via a command-line interface, users employ a mouse-based window and menu system characterized by a desktop metaphor.
- The user moves the mouse to position its pointer on images, or icons, on the screen (the desktop) that represent programs, files, directories, and system functions.
- Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory known as a folder or pull down a menu that contains commands
- The graphical user interface is used for playing games, watching videos, etc. these are done with the help of GUI because all these applications require graphics.
- The GUI is one of the necessary interfaces because only by using the user can clearly see the picture, play videos.
- The basic components of GUIs are :
  - Start menu with program groups
  - Taskbar which showing running programs
  - Desktop screen
  - Different icons and shortcuts.

Touch-Screen Interface

- A command-line interface or a mouse-and-keyboard system is impractical for most mobile systems, smartphones and handheld tablet computers use a touch-screen interface.

➢ Here, users interact by making gestures on the touch screen for example, pressing and swiping fingers across the screen.

➢ Although earlier smartphones included a physical keyboard, most smartphones and tablets now simulate a keyboard on the touch screen.

➢ The iPad and the iPhone use the Springboard touch-screen interface

Choice of interface

➢ System administrators who manage computers and power users who have deep knowledge of a system frequently use the command-line interface. The user interface can vary from system to system and even from user to user within a system.

➢ The design of a useful and intuitive user interface is therefore not a direct function of the operating system.

➢ The interface that is used with the help of OS for a particular task and that task can be performed with minimum possible time and the output is shown on the screen in that case we use the choice of interface.

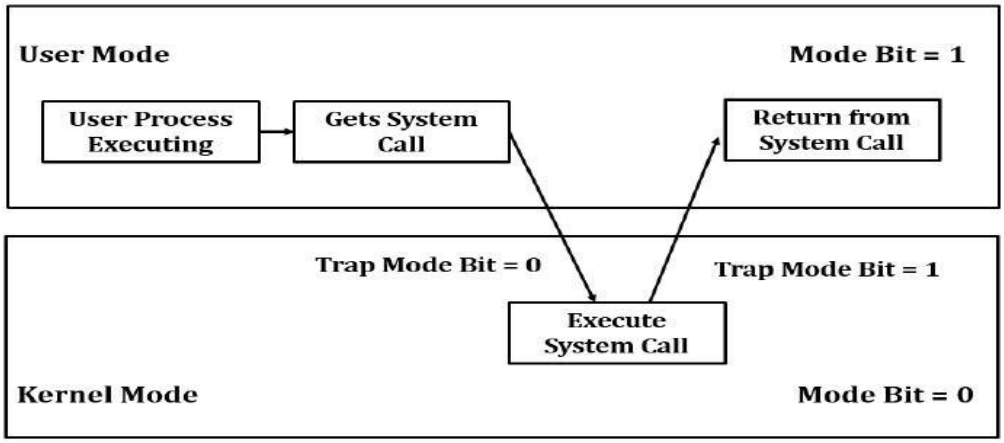
➢ The choice of interface means the OS checks the task and finds out which interface can be suitable for a particular task. So that type of interface is called the choice of interface and this can be done with the help of an OS.

1.7 SYSTEM CALLS

A system call is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS. System call offers the services of the operating system to the user programs via API (Application Programming Interface). System calls are the only entry points for the kernel system.

Working of System Call:

Step 1) The processes executed in the user mode till the time a system call interrupts it.
Step 2) After that, the system call is executed in the kernel-mode on a priority basis.
Step 3) Once system call execution is over, control returns to the user mode.
Step 4) The execution of user processes resumed in Kernel mode.



➢ Three most common APIs are

- ❖ Win32 API for Windows,
- ❖ POSIX API (all versions of UNIX, Linux, and Mac OS X), and
- ❖ Java API for the Java virtual machine (JVM)

Need of System Call:

- ❖ Reading and writing from files demand system calls.
- ❖ If a file system wants to create or delete files, system calls are required.
- ❖ System calls are used for the creation and management of new processes.
- ❖ Network connections need system calls for sending and receiving packets.
- ❖ Access to hardware devices like scanner, printer, need a system call.

Example for System Call

- ➢ Consider that there is a user program that will read the content of one file and will copy it into another file. Here the first information that the system requires is the name of two files, the first file from where the data has to be read (input file) and the second file where the data has to be copied (output file).Depending on the different types of operating systems a sequence of system calls will be required.
- ➢ A sequence of system calls in an interactive system will be:

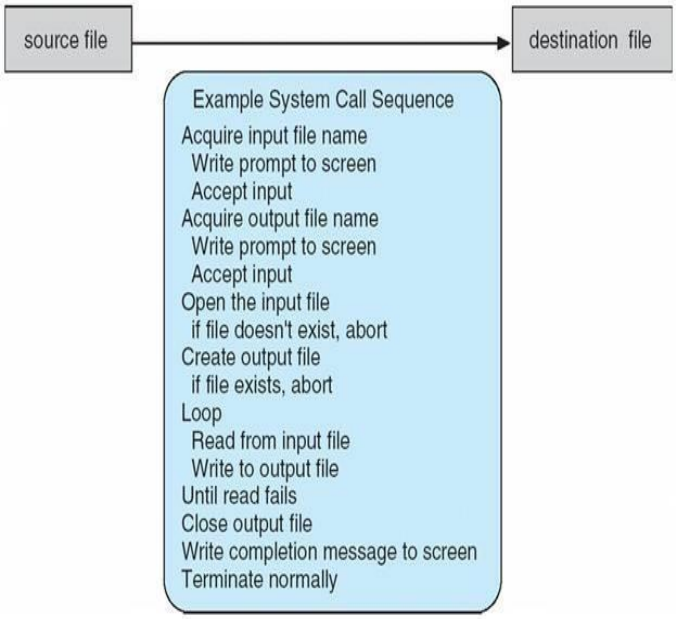


Fig: Examples of System Calls

Three general methods exist for passing parameters to the OS:

1. Parameters can be passed in registers.
2. When there are more parameters than registers, parameters can be stored in a block and the block address can be passed as a parameter to a register.
3. Parameters can also be pushed on or popped off the stack by the operating system.

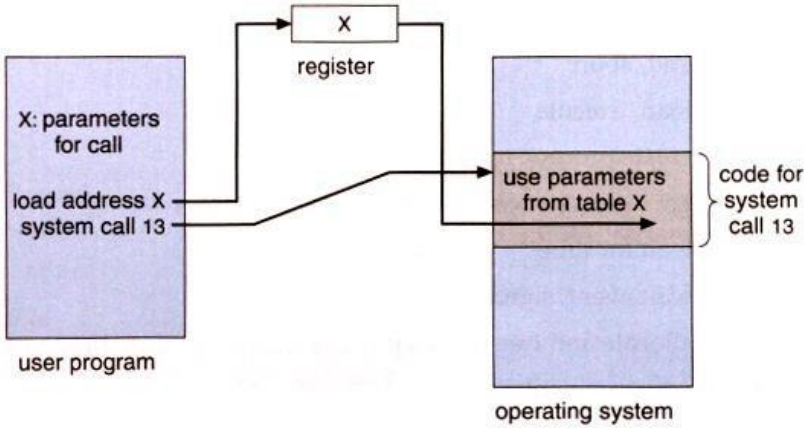


Fig: Passing Parameters to the Operating Systems

TYPES OF SYSTEM CALLS

- There are 6 different categories of system calls:
  - Process control,
  - File Management
  - Device management
  - Information maintenance
  - Communication
  - Protection

Process Control

- System call includes calls to create process, terminate process ,load, execute ,get process attributes, set process attributes,wait event, signal event, allocate and free memory.
- A running program needs to be able to halt its execution either normally (end()) or abnormally (abort()). If a system call is made to terminate the currently running program abnormally, or if the program runs into a problem and causes an error trap, a dump of memory is taken and an error message generated.
- The dump is written to a special log file on disk and may be examined by a debugger.
- A system program designed to aid the programmer in finding and correcting errors, or bugs to determine the cause of the problem.

File Management

- System calls under this category include calls to create file, delete file, open, close, read, write, reposition, get file attributes, set file attributes.

Device Management

- System calls under this category includes calls to request device, release device, read, write, reposition, get device attributes, set device attributes ,logically attach or detach devices.
- A process need several resources to execute main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available

Information Maintenance

- System calls under this category includes calls to return information about the system, such as 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.
- the operating system keeps information about all its processes, and system calls are used to access this information

Communication

- System calls under this category includes calls to create, delete communication connection, send, receive messages, transfer status information, attach or detach remote devices.
- There are two models of inter process communication, the message-passing model and the shared memory model.
- In a shared memory model, a process uses memory for communication. One process will create a memory partition which other processes can access. A shared segment can be attached multiple times by the same process
- In message passing communication is done either direct or indirect
  - In direct communication process address each other by their PID assigned to them by OS.
  - In indirect communication messages are sent and received via mailbox

Protection

- System calls under this category includes calls to get file permissions, set file permissions.
- Protection provides a mechanism for controlling access to the resources provided by a computer system. protection was a concern only on multiprogrammed computer systems with several users.
- However, with the advent of networking and the Internet, all computer systems, from servers to mobile handheld devices, must be concerned with protection.

	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() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Table: Examples of WINDOWS and UNIX system calls

Important System Calls Used in OS

1. wait()

- In certain systems, a process must wait for another process to finish running before continuing. When a parent process creates a child process, the parent process's execution is suspended until the child process has finished running.
- With a wait() system call, the parent process is automatically suspended. Control returns to the parent process once the child process has completed its execution.

2. fork()

- The fork system call in OS is used by processes to make copies of themselves. By using this system, the Call parent process can create a child process, and the parent process's execution will be halted while the child process runs.

3. exec()

- When an executable file replaces an earlier executable file within the context of an active process, this system call is executed. The original process identifier is still present even though a new process is not created; instead, the new process replaces the old one's stack, data, head, data, etc.

4. kill()

- The OS uses the kill() system call to urge processes to terminate by sending them a signal. A kill system call can have different meanings and is not always used to end a process.

5. exit()

- An exit system call is used when the program must be stopped. When the exit system call is used, the resources that the process was using were released.

1.8 SYSTEM PROGRAMS

- In addition to system calls, modern systems also provide variety of system programs. System programs provide a convenient environment for program development and execution. These programs act as an interface between the operating system and application programs.
- They provide an environment in which application programs can be developed and executed in a convenient manner.
- The system programs can be classified into following categories

File Management

- The System programs under this category provide commands, such as cut, copy, save, print, etc to perform various operations on files and directories.

File modification

- The system programs under this category allow creating or modifying the contents of a file stored on disk or some other storage devices.
- Text editor is a system program that belongs to this category. Vi is the text editor used in Unix operating systems.

Communications

- The system programs under this category enable communication among different users, processes, or systems by establishing virtual connection between them.
- With the help of these programs, user can send messages to other users, log on some remote systems or transfer data from other system to its own systems.

Status information

- The system programs under this category are used to present the status of the computer system, such as system date and time, number of users connected, CPU utilization, disk and memory usage, Configuration information, etc.

Programming Language Support

- Several programming languages support different system programs, such as compilers, assemblers and interpreters.
- These system programs are generally provided to users along with OS.

Program loading and execution

- After the user program has been compiled, it needs to be in main memory for execution.
- The task of loading a program into memory is performed by loader, a system program.
- The successful execution of the program also requires debugging, which is performed by debugger (a system program under this category).

1.9 OPERATING SYSTEM - DESIGN AND IMPLEMENTATION

- An operating system is a construct that allows the user application programs to interact with the system hardware. Operating system by itself does not provide any function but it provides an atmosphere in which different applications and programs can do useful work.

- There are many problems that can occur while designing and implementing an operating system.

Operating System Design Goals

- It is quite complicated to define all the goals and specifications of the operating system while designing it. The design changes depending on the type of the operating system i.e if it is batch system, time shared system, single user system, multi-user system, distributed system etc.

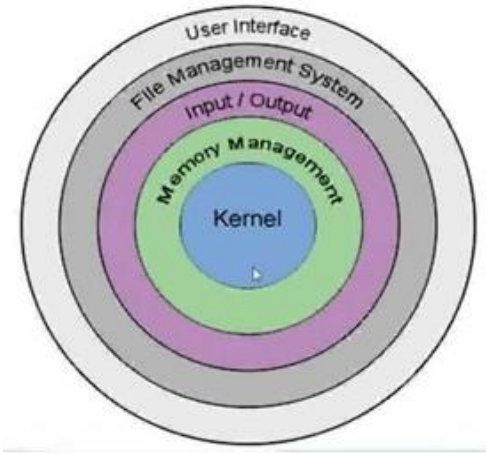


Fig : layered Operating System Design

- There are basically two types of goals while designing an operating system based on the requirements. These are

1. **User Goals - The user wants operating system should be convenient to use, easy to learn, reliable, safe, and fast**
2. **System Goals – Those who design the operating system requires that the system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient**

Operating System Mechanisms and Policies

- There is no specific way to design an operating system as it is a highly creative task. However, there are general software principles or policies that are applicable to all operating systems.
- A subtle difference between mechanism and policy is that mechanism shows how to do something, and policy shows what to do. Policies may change over time and this would lead to changes in mechanism. So, it is better to have a general mechanism that would require few changes even when a policy change occurs.
- For example - If the mechanism and policy are independent, then few changes are required in mechanism if policy changes. If a policy favors I/O intensive processes over CPU intensive processes, then a policy change to preference of CPU intensive processes will not change the mechanism.

Operating System Implementation

- The operating system needs to be implemented after it is designed. Earlier they were written in assembly language, but now higher-level languages are used. The first system not written in assembly language was the Master Control Program (MCP) for Burroughs Computers.

Advantages of Higher-Level Language

- There are multiple advantages to implementing an operating system using a higher-level language such as: the code is written more fast, it is compact and also easier to debug and understand. Also, the operating system can be easily moved from one hardware to another if it is written in a high-level language.

Disadvantages of Higher-Level Language

- Using high level language for implementing an operating system leads to a loss in speed and increase in storage requirements. However, in modern systems only a small amount of code is needed for high performance, such as the CPU scheduler and memory manager. Also, the bottleneck routines in the system can be replaced by assembly language equivalents if required.

1.10 STRUCTURING METHODS

- The operating systems are large and complex. A common approach is to partition the task into small components, or modules, rather than have one **monolithic** system.

- The structure of an operating system can be defined the following structures.

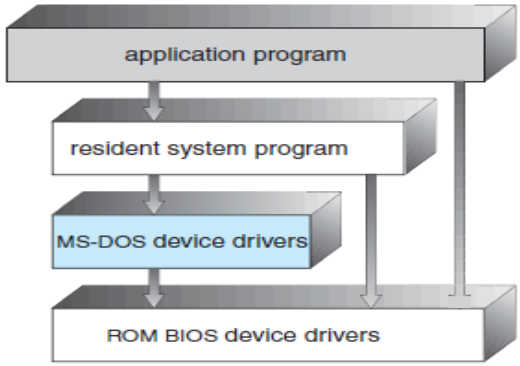
  - Simple structure (Monolithic Structure)
  - Layered approach
  - Microkernels
  - Modules
  - Hybrid systems

1. Simple structure: (Monolithic Structure)

- The Simple structured operating systems do not have a well defined structure. These systems will be simple, small and limited systems.

Example 1: MS-DOS.

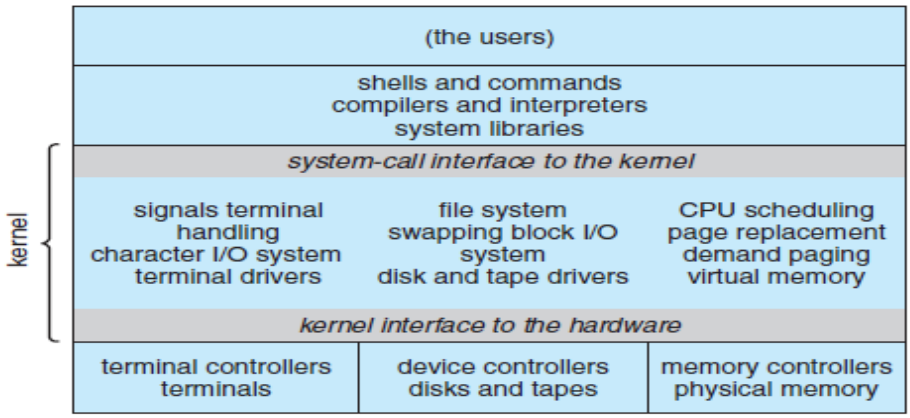
- MS-DOS – written to provide the most functionality in the least space
- Not divided into modules
- Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated
- In MS-DOS application programs are able to access the basic I/O routines. This causes the entire systems to be crashed when user programs fail.



MS DOS Layer Structure

Example 2 : Traditional UNIX OS Structure

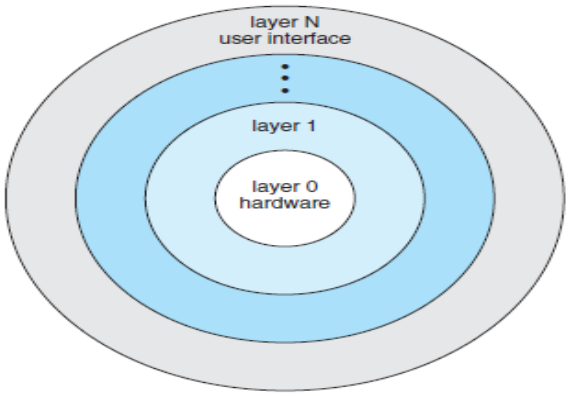
- It consists of two separable parts: the kernel and the system programs.
- The kernel is further separated into a series of interfaces and device drivers
- The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.
- This monolithic structure was difficult to implement and maintain.



Traditional UNIX OS Structure

2. Layered approach:

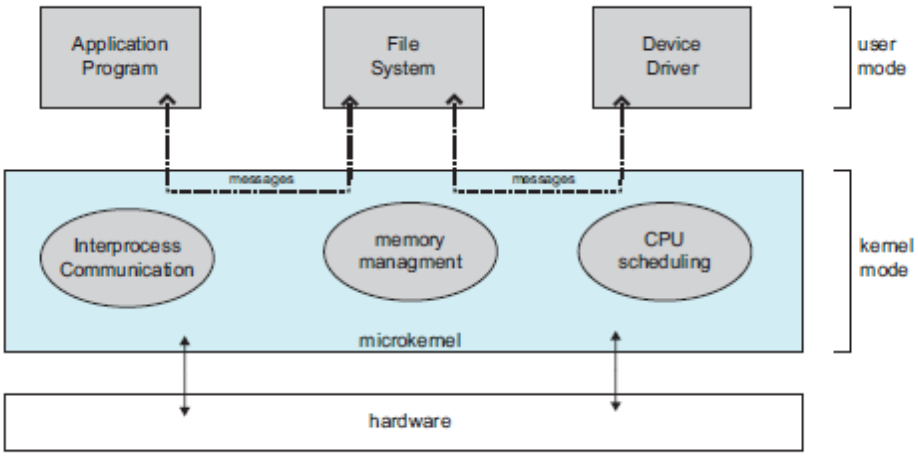
- A system can be made modular in many ways. One method is the **layered approach**, in which the operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.



- An operating-system layer is an implementation of an abstract object made up of data and the operations that can manipulate those data.
- The main advantage of the layered approach is simplicity of construction and debugging. The layers are selected so that each uses functions (operations) and services of only lower-level layers.
- Each layer is implemented only with operations provided by lower-level layers. A layer does not need to know how these operations are implemented; it needs to know only what these operations do.
- The major difficulty with the layered approach involves appropriately defining the various layers because a layer can use only lower-level layers.
- A problem with layered implementations is that they tend to be less efficient than other types.

3. Microkernels:

- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel** approach.
- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs.

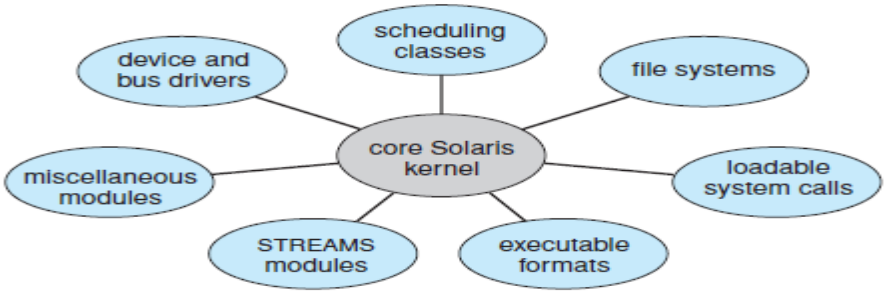


- Microkernel provide minimal process and memory management, in addition to a communication facility.
- The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space.
- The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel.
- One benefit of the microkernel approach is that it makes extending the operating system easier. All new services are added to user space and consequently do not require modification of the kernel.
- The performance of microkernel can suffer due to increased system-function overhead.

4. Modules:

- The best current methodology for operating-system design involves using **loadable kernel modules**
- The kernel has a set of core components and links in additional services via modules, either at boot time or during run time.
- The kernel provides core services while other services are implemented dynamically, as the kernel is running.
- Linking services dynamically is more comfortable than adding new features directly to the kernel, which would require recompiling the kernel every time a change was made.

Example: Solaris OS



- The Solaris operating system structure is organized around a core kernel with seven types of loadable kernel modules:
  - Scheduling classes
  - File systems
  - Loadable system calls

- Executable formats
- STREAMS modules
- Miscellaneous
- Device and bus drivers

5. Hybrid Systems:

- The Operating System combines different structures, resulting in hybrid systems that address performance, security, and usability issues.
- They are monolithic, because having the operating system in a single address space provides very efficient performance.
- However, they are also modular, so that new functionality can be dynamically added to the kernel.
- Example: Linux and Solaris are monolithic (simple) and also modular, IOS so that new functionality can be dynamically added to the kernel.
- There are Three hybrid systems:
    - the Apple Mac OS operating system
    - mobile operating systems—iOS and Android

Eg 1 : Mac OS

- The Apple Mac OS uses a hybrid structure. As shown in the figure it is a layered system.

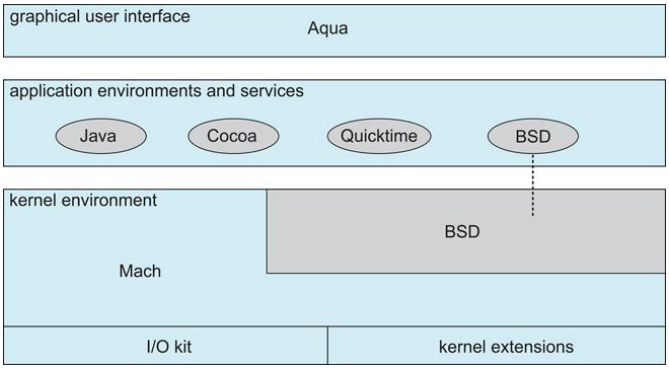


Fig : Mac Os Structure

- kernel consists of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called kernel extensions)

Eg 2 : Android OS : Google's Android OS uses a hybrid structure as shown below:

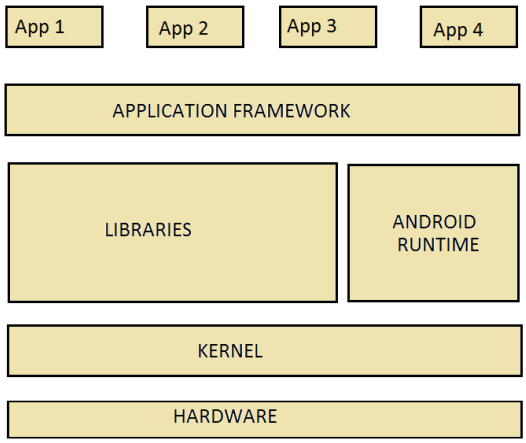


Fig : Google' Android Structure