Chapter 1 :Introduction of Operating System

1.1 Introduction and History of Operating System

- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs.
- The operating system is a vital component of the system software in a computer system.
- An operating system falls under the category of system software that performs all the fundamental tasks like file management, memory handling, process management, handling the input/output and governing and managing the peripheral devices like disk drives, networking hardware, printers, etc.

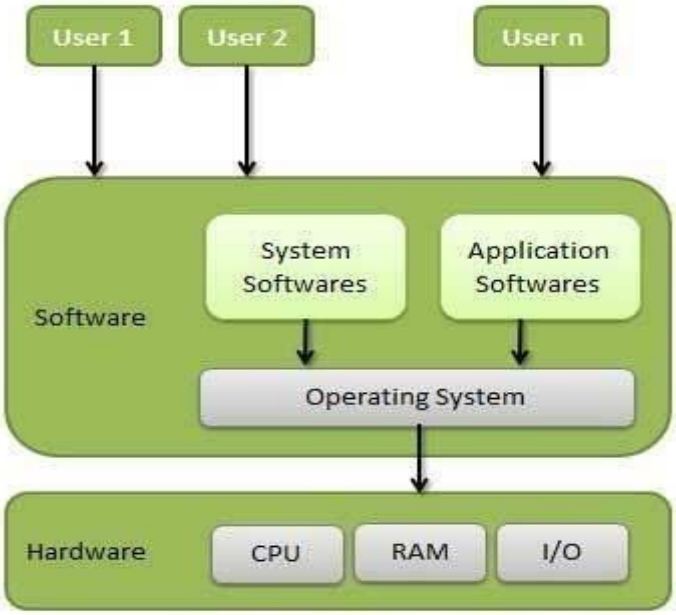


Fig: Operating System

Some examples of OS are Linux, Windows, OS X, Solaris, OS/400, Chrome OS, etc.

Objective/Goals of OS

An Operating System has a special program which controls the execution of application programs.

OS acts as an intermediary among applications and the hardware components. OS can be thought of as having three objectives. These are:

- **Convenience**: It makes a computer more suitable to use.
- **Efficiency**: It provides the computer system resources with efficiency and in easy to use format.
- **Ability to develop**: It should be built in such a way that it permits the efficient development, testing and installation of new system functions without interfering with service.

1.Operating system as an extended /virtual machine

An **Operating System (OS)** can be thought of as an **extended machine** because it hides the complexity of hardware and provides a simple, easy-to-use interface for users and applications. Here's a simple explanation:

Without an OS:

- Computers work with complicated hardware (processors, memory, disks, etc.).
- You would have to write complex code to directly control hardware every time you want to do something, like save a file or display text on the screen.

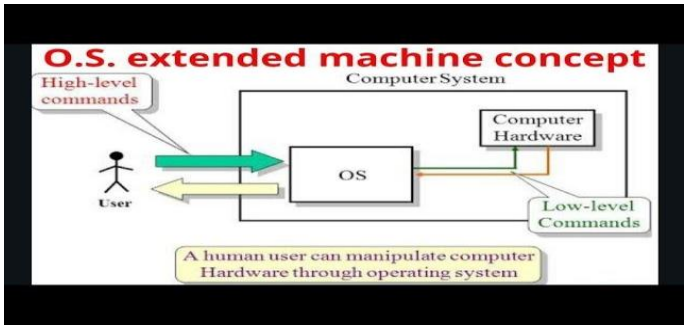
With an OS:

- The OS acts as a **middleman** between you (or your programs) and the hardware.
- It gives you simple tools and commands to perform tasks, like saving a file (Save button) or running a program (just click an icon).
- Instead of worrying about how the hardware works, you just focus on what you want to do.

Human Input: You type commands or click buttons (e.g., "Save File").

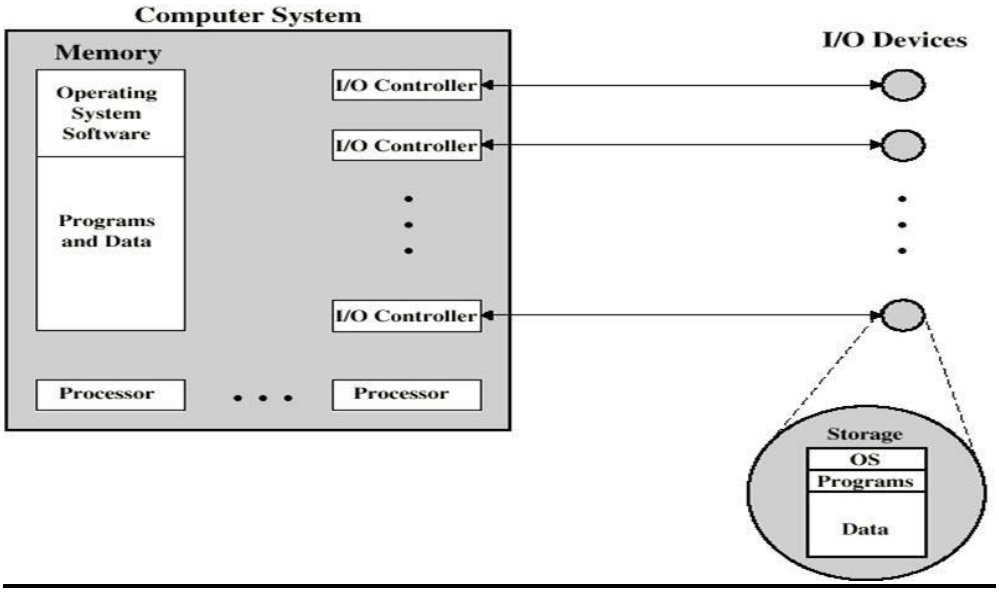
What the OS Does: The OS translates these into machine instructions (binary) that the hardware Understand.

Example: Clicking "Save" in a document editor sends a high-level instruction to the OS. The OS then tells the storage device (e.g., hard drive) to write data in binary form.



2.OS as a Resource Manager

- The concept of an operating system as providing abstractions to application programs is a top-down view. Alternatively, bottom-up view holds that the OS is there to manage all pieces of a complex system. Also, A computer consists of a set of resources such as processors, memories, timers, disks, printers and many others. The Operating System manages these resources and allocates them to specific programs.
- **As a resource manager**, the Operating system provides the controlled allocation of the processors, memories, I/O devices among various programs.
- Moreover, multiple user programs are running at the same time. The processor itself is a resource and the Operating System decides how much processor time should be given for the execution of a particular user program.
- The operating system also manages memory and I/O devices when multiple users are working.
- The primary task of OS is to keep the track of which programs are using which resources, to grant resource requests, to account for usage, and to resolve conflicting requests from different programs and users.
- Also, **Resource management** includes multiplexing (sharing) resources in two ways: in time and in space.



History of Operating Systems

1.First Generation (1940s - Early 1950s):

- No operating systems: Early computers were not equipped with any OS.
- Machine Language: All programming was done directly in machine language (binary code), meaning programmers had to write instructions that the hardware could directly understand.
- Manual control: Instead of an OS, machines were controlled using plugboards and physical rewiring, where operators set up connections manually to control the computer's operations.
- Usage: These computers were primarily used for simple mathematical calculations, and operating systems were not needed at the time because the tasks were straightforward**.**

2.Second Generation (1955-1965):

- Introduction of the first OS: In the 1950s, the first operating system, called GMOS, was developed by General Motors for IBM's 701 computer.
- Batch Processing: The OS in this period was called single-stream batch processing. It processed data in groups or batches rather than individually, which was more efficient.
- Mainframe computers: Large mainframe computers, such as IBM's machines, became popular in this era. These were extremely expensive and used mainly by large corporations and government agencies.

- Limited Access: The cost of these machines meant they were not accessible to individuals, and only skilled operators worked in dedicated computer rooms.

3.Third Generation (1965-1980):

- Multiprogramming: One of the major developments was multiprogramming, where a computer could run multiple programs at the same time. This made much better use of the computer's resources and kept the CPU active nearly all the time.
- Minicomputers: Smaller, more affordable minicomputers began to appear. The DEC PDP-1, released in 1961, was an early example of these systems. It cost $120,000—much less than the larger mainframes—and had only 4K of memory, but it became very popular.
- Growth of the industry: The availability of minicomputers led to the development of a new industry. Many small businesses and educational institutions could now afford these machines.
- Personal Computers (PCs): The success of these minicomputers laid the groundwork for the development of personal computers, which would take off in the 1980s.

4.Fourth Generation (1980-Present):

- Personal Computing: The 1980s marked the rise of personal computers that were affordable for individuals, unlike the expensive minicomputers of the previous generation. These PCs were much more accessible and began to be used in homes, schools, and small businesses.
- Microsoft and Windows: In 1981, Microsoft introduced MS-DOS, an early operating system for personal computers. Although MS-DOS was effective, it was not user-friendly due to cryptic command-line inputs. The Windows operating system, introduced in the 1980s, made computing easier by providing a graphical user interface (GUI) that allowed users to interact with their computers using windows, icons, and buttons.
    - Windows Evolution: Windows became hugely successful with versions like Windows 95, Windows XP (still widely used even years later), and later versions like Windows 7.
- Apple and Macintosh: At the same time, Apple introduced the Macintosh in 1984, with an intuitive graphical interface. This made personal computing even more popular and accessible to non-technical users. The Macintosh became a major competitor to Windows-based systems.
- Computing for Everyone: By the end of the 1980s and into the 1990s, operating systems like Windows and Macintosh made personal computers affordable and user-friendly, which led to their widespread use across households and businesses worldwide

1.2 Operating System Concepts and Functionalities

Operating System Concepts

1.2.1 Process

- A process is an instance of a program in execution.
- A program becomes process when executable file is loaded in the main memory.
- A process is defined as an entity which represents the basic unit of work to be implemented in the system.

1.2.2 Files

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

1.2.3 System Calls

- In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel. System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

1.2.3 The Shell

The **shell** in an operating system is a program that allows users to interact with the computer through commands. It serves as an interface between the user and the operating system, enabling the user to give instructions and receive feedback. Here's how it works in simple terms:

Key Functions of the Shell:

1. ***Command Interpreter****:*

- The shell **takes your input** (commands) and interprets them for the operating system. For example, if you type ls to list files, the shell translates that into a system command to display the files.

2. ***Interaction****:* ○ The shell provides a way for you to **communicate** with the operating system. You can ask the system to perform tasks like opening programs, managing files, or running scripts.

3. ***Two Modes of Operation****:*

- **Interactive Mode**: You type commands directly, and the shell executes them immediately. This is like talking to the computer and getting instant responses.
- **Script Mode**: You can write **shell scripts** (a series of commands) and save them in a file. These scripts can be run automatically without needing to type the commands each time.

Function of Operating System

Following are some of important functions of an operating System:

- **Processor management** which involves putting the tasks into order and pairing them into manageable size before they go to the CPU.
- **Memory management** which coordinates data to and from RAM (random-access memory) and determines the necessity for virtual memory.
- **Device management** which provides interface between connected devices.
- **Storage management** which directs permanent data storage.
- **Application** which allows standard communication between software and your computer.
- **User interface** which allows you to communicate with your computer.
- It provides **file management** which refers to the way that the operating system manipulates, stores, retrieves and saves data.
- **Error Handling** is done by the operating system. It takes preventive measures whenever required to avoid errors.

- **Security** − By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Control over system performance** − Recording delays between request for a service and response from the system.
- **Coordination between other softwares and users** − Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

1.3 Operating System Structure

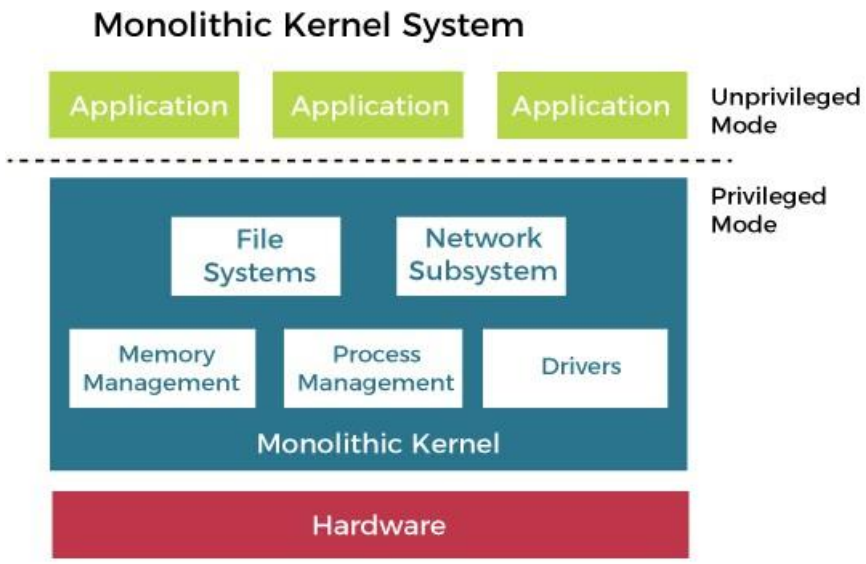
An operating system might have many structure. According to the structure of the operating system; operating systems can be classified into many categories.

Some of the main structures used in operating systems are:

1.3.1 Monolithic architecture of operating system

It is the oldest architecture used for developing operating system. Operating system resides on kernel for anyone to execute. System call is involved i.e. Switching from user mode to kernel mode and transfer control to operating system shown as event 1. Many CPU has two modes, kernel mode, for the operating system in which all instruction is allowed and user mode for user program in which I/O devices and certain other instruction are not allowed. Two operating system then examines the parameter of the call to determine which system call is to be carried out shown in event 2. Next, the operating system indexes into a table that contains procedure that carries out system call. This

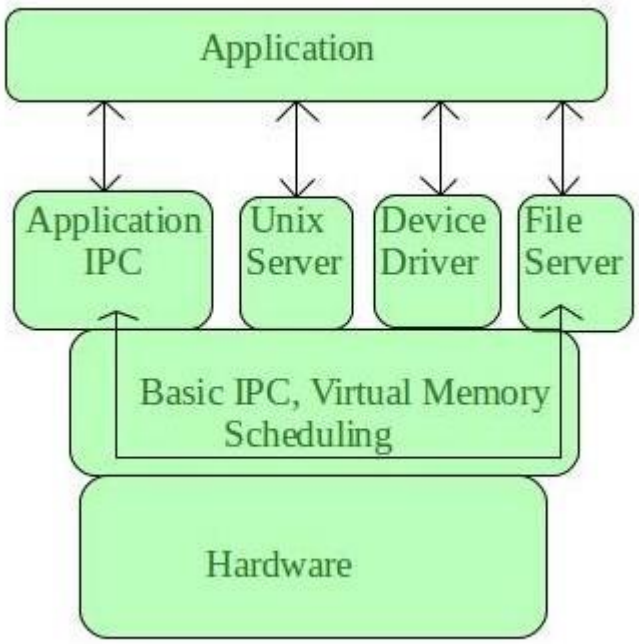
operation is shown in events. Finally, it is called when the work has been completed and the system call is finished, control is given back to the user mode as shown in event 4.



1.3.2 Microkernel Architecture

Since the kernel is the core part of the operating system, so it is meant for handling the most important services only. Thus in this architecture, only the most important services are inside the kernel and the rest of the OS services are present inside the system application program. Thus users are able to interact with those not-so-important services within the system application. And the microkernel is solely responsible for the most important services of the operating system they are named as follows: Inter Process-Communication

- Memory Management
- CPU-Scheduling

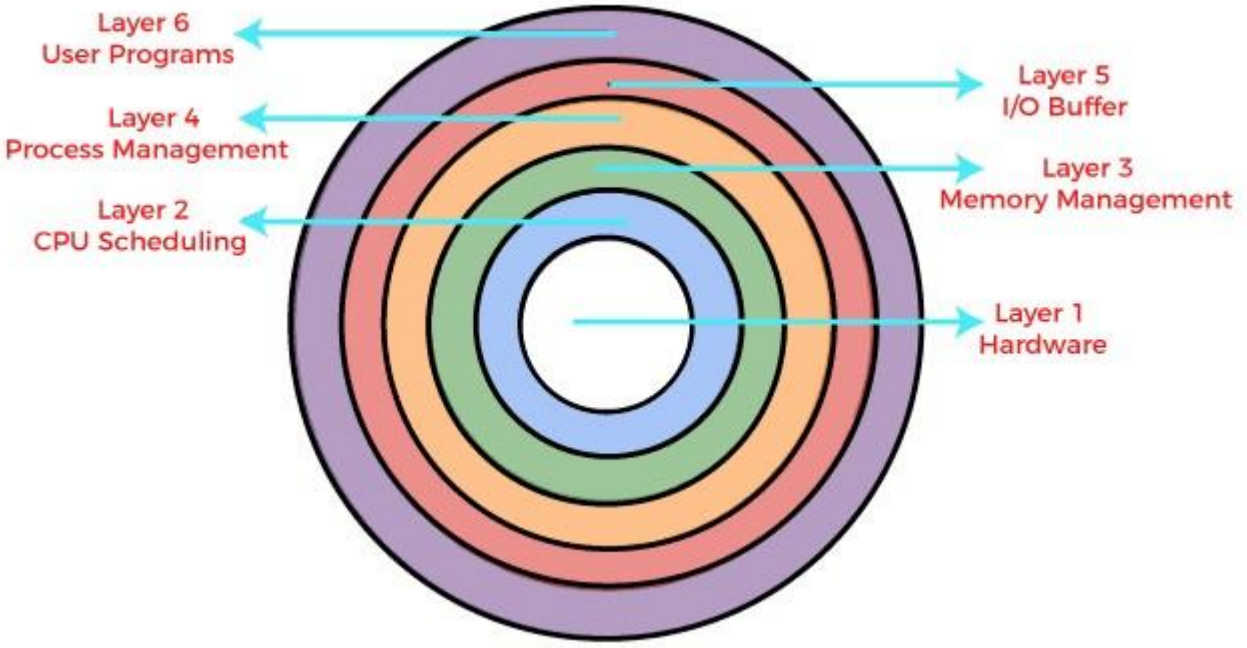


Features of Microkernel-Based Operating System

- **Minimal Core** : The kernel only manages basic tasks like CPU scheduling, memory management, and inter-process communication. All other services (like device drivers) run in user space, outside the kernel.
- **Modularity** : System services are separated into independent modules. This makes it easy to update or replace individual components without affecting the entire system.
- **Stability and Fault Isolation** : If a service like a device driver fails, it won't crash the whole system. Each service runs independently, so the core OS remains stable.
- **Ease of Maintenance** : Since services are independent, fixing bugs or adding new features doesn't require major changes to the kernel itself, making maintenance easier.

1.3.3 Layered Architecture of operating system

The layered Architecture of operating system was developed in 60's in this approach; the operating system is broken up into number of layers. The bottom layer (layer 0) is the hardware layer and the highest layer (layer n) is the user interface layer as shown in the figure.



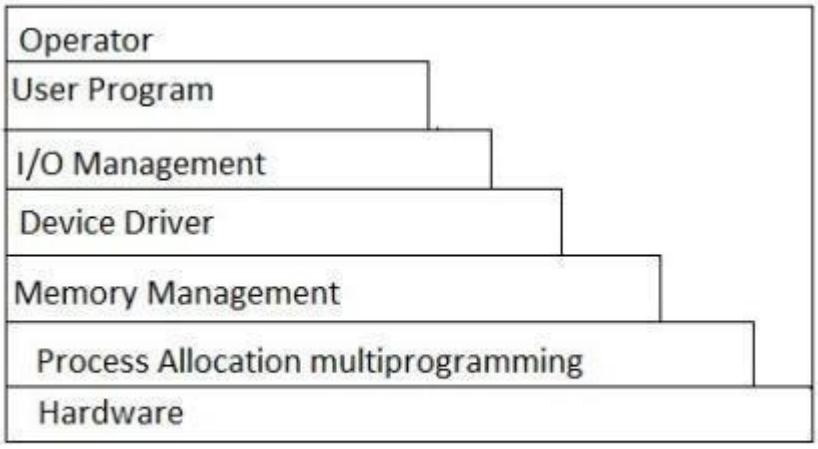


fig:- layered Architecture

The layered are selected such that each user functions and services of only lower level layer. The first layer can be debugged wit out any concern for the rest of the system. It user basic hardware to implement this function once the first layer is debugged., it's correct functioning can be assumed while the second layer is debugged & soon. If an error is found during the debugged of particular

layer, the layer must be on that layer, because the layer below it already debugged. Because of this design of the system is simplified when operating system is broken up into layer.

Os/2 operating system is example of layered architecture of operating system another example is earlier version of Windows NT.

The main disadvantage of this architecture is that it requires an appropriate definition of the various layers & a careful planning of the proper placement of the layer.

1.3.3 Virtual memory architecture of operating system

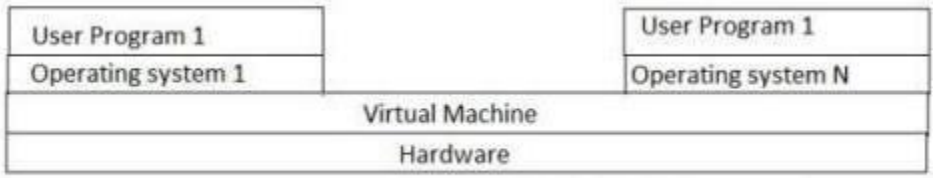


fig:- virtual memory architecture of os

Virtual machine is an illusion of a real machine. It is created by a real machine operating system, which make a single real machine appears to be several real machine. The architecture of virtual machine is shown above.

The best example of virtual machine architecture is IBM 370 computer. In this system each user can choose a different operating system. Actually, virtual machine can run several operating systems at once, each of them on its virtual machine.

Its multiprogramming shares the resource of a single machine in different manner.

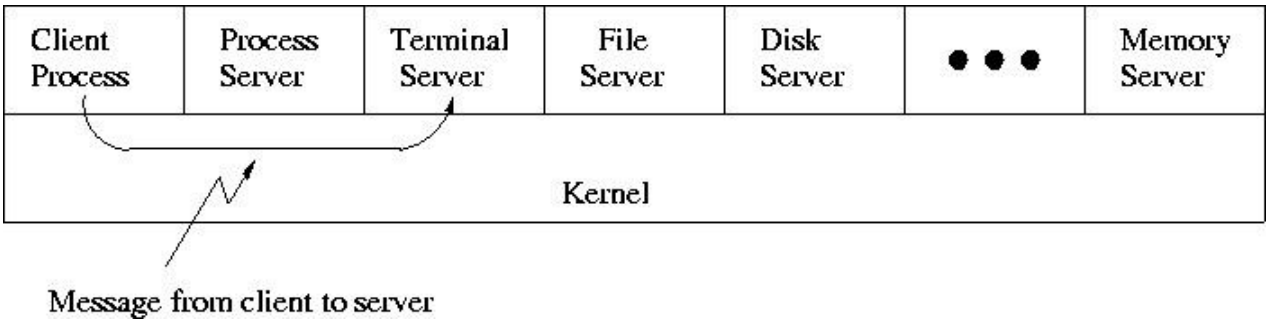
The concepts of virtual machine are:-

a. Control program (cp):- cp creates the environment in which virtual machine can execute. It gives to each user facilities of real machine such as processor, storage I/0 devices.
b. conversation monitor system (cons):- cons is a system application having features of developing program. It contains editor, language translator, and various application packages.

c. Remote spooling communication system (RSCS):- provide virtual machine with the ability to transmit and receive file in distributed system.
d. IPCS (interactive problem control system):- it is used to fix the virtual machine software problems.

1.3.4 Client/Server architecture of operating system

A trend in modern operating system is to move maximum code into the higher level and remove as much as possible from operating system, minimising the work of the kernel. The basic approach is to implement most of the operating system functions in user processes to request a service, such as request to read a particular file, user send a request to the server process, server checks the parameter and finds whether it is valid or not, after that server does the work and send back the answer to client server model works on request- response technique i.e. Client always send request to the side in order to perform the task, and on the other side, server gates complementing that request send back response. The figure below shows client server architecture.



In this model, the main task of the kernel is to handle all the communication between the client and the server by splitting the operating system into number of ports, each of which only handle some specific task i.e. file server, process server, terminal server and memory service.

Another advantage of the client-server model is it's adaptability to user in distributed system. If the client communicates with the server by sending it the message, the client need not know whether it was send a ……. Is the network to a server on a remote machine? As in case of client, same thing happens and occurs in client side that is a request was send and a reply come back

Kernel mode and User mode

In any modern operating system, the CPU is actually spending time in two very distinct modes to make sure it works correctly:

Kernel Mode

In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system. Crashes in kernel mode are catastrophic; they will halt the entire PC.

User Mode

In User mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory. Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable. Most of the code running on your computer will execute in user mode.

Necessity of Dual Mode (User Mode and Kernel Mode) in Operating System

The lack of a dual mode i.e user mode and kernel mode in an operating system can cause serious problems. Some of these are:

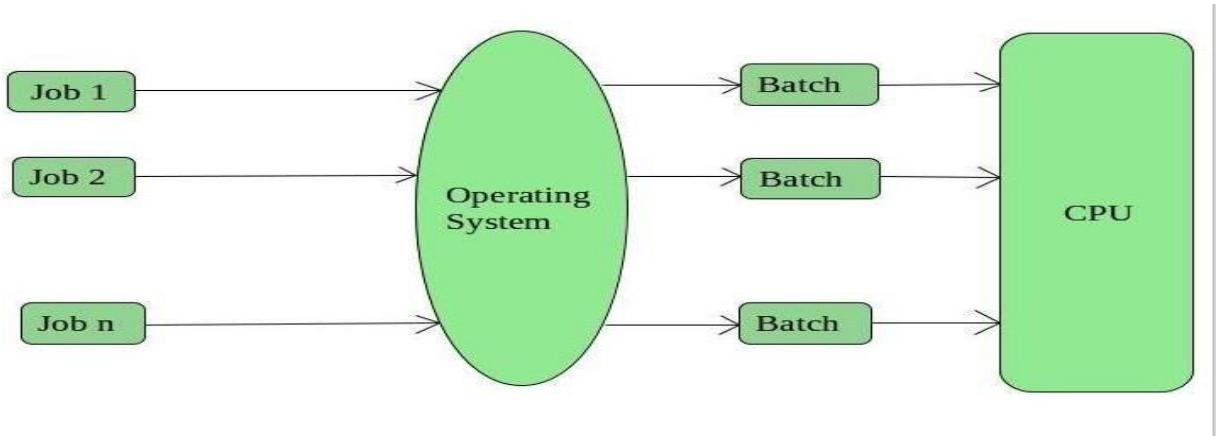
- A running user program can accidentally wipe out the operating system by overwriting it with user data.
- Multiple processes can write in the same system at the same time, with disastrous results.

1.4 Types and evolution of Operating System

Operating systems are there from the very first computer generation and they keep evolving with time. Some of the important types of operating systems which are most commonly used are:

i. Batch Operating System

This type of operating system do not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required by any job to complete. Processors of the batch systems knows how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

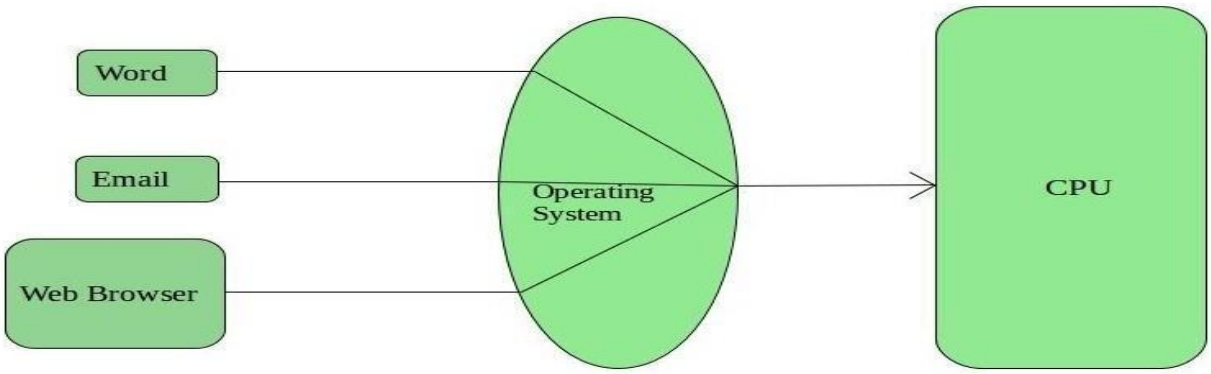
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometime costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System: Payroll System, Bank Statements etc.

ii. Time-Sharing Operating Systems

Each task has given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task

can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.



Advantages of Time-Sharing OS:

- Each task gets an equal opportunity
- Less chances of duplication of software
- CPU idle time can be reduced

Disadvantages of Time-Sharing OS:

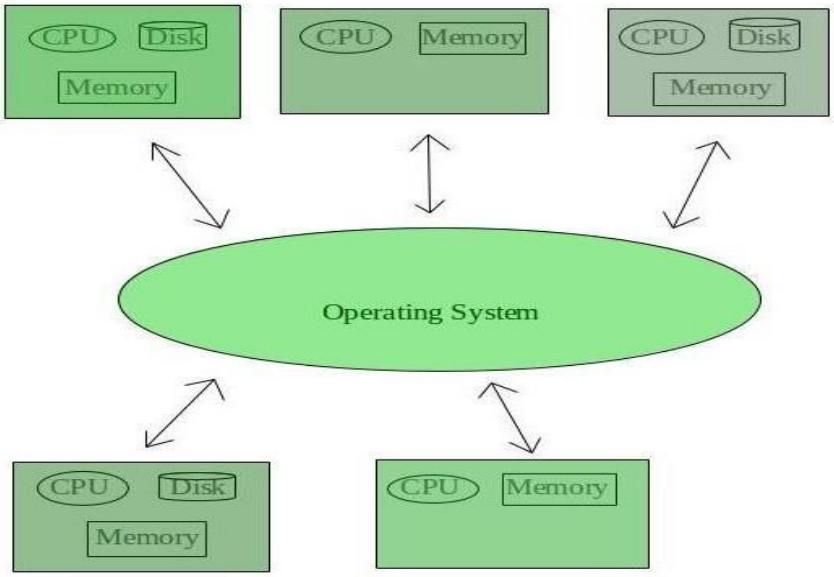
- Reliability problem
- One must have to take care of security and integrity of user programs and data Data communication problem

Examples of Time-Sharing OSs are: Multics, Unix etc.

iii. Distributed Operating System

These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world and, that too, with a great pace. Various autonomous interconnected computers communicate each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred as **loosely coupled systems** or distributed systems. These systems processors differ in sizes and functions. The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but

on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

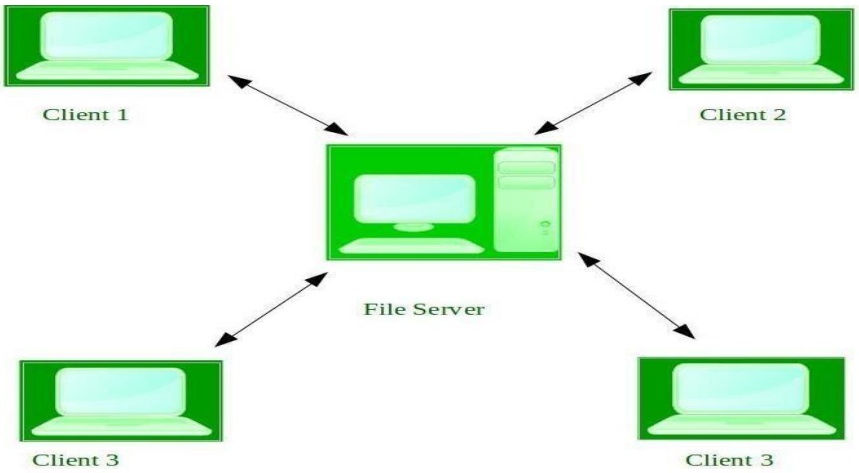
Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which are used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.

Examples of Distributed Operating System are- LOCUS,AMOEBA etc.

iv. Network Operating System

These systems runs on a server and provides the capability to manage data, users, groups, security, applications, and other networking functions. These type of operating systems allows shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that's why these computers are popularly known as **tightly coupled systems**.



Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated to the system
- Server access are possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on central location for most operations Maintenance and updates are required regularly

Examples: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD etc.

v. Parallel Operating System

- Parallel operating systems are used to interface multiple networked computers to complete tasks in parallel.
- The architecture of the software is often a UNIX-based platform, which allows it to coordinate distributed loads between multiple computers in a network.
- Parallel operating systems are able to use software to manage all of the different resources of the computers running in parallel, such as memory, caches, storage space, and processing power.
- Parallel operating systems also allow a user to directly interface with all of the computers in the network
- A parallel operating system works by dividing sets of calculations into smaller parts and distributing them between the machines on a network. To facilitate communication between the processor cores and memory arrays, routing software has to either share its memory by assigning the same address space to all of the networked computers, or distribute its memory by assigning a different address space to each processing core.

vi. Real Time operating System

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the **response time**. So in this method, the response time is very less as compared to online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

Advantages of Real Time Operating System:

- Maximum Consumption
- Task Shifting
- Error Free

- 24-7 systems

Disadvantages of Real Time Operating System:

- Limited Tasks
- Use heavy system resource
- Complex Algorithms
- Expensive

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

vii. Multitasking, multiprogramming and multiprocessing Multitasking

Both the memory and CPU time is shared among the tasks It performs multiple tasks concurrently. It gives a illusion that all the processes or tasks are performed simultaneously. For example, in your desktop you can listen to music, download things simultaneously. The concept is that the CPU time is shared equally among the processes or tasks. Again take an example that 3 tasks are loaded into the main memory. Consider that the time allocated for each program is 5 micro seconds. Now the first task is executed. After 5 microseconds irrespective whether the task gets completed or not it gets switched to the second task. Similarly, after 5 seconds it goes to the third task. Thus the CPU time is shared equally among the tasks.

Multiprogramming

The concept of multiprogramming is that more than one program that is to be executed by the processor is loaded into the memory.

Say we have 2 programs loaded into the memory. The first program that is loaded is getting executed. At one point of time it requires input from the user or waiting for some data. During the waiting time, the CPU is idle. Instead of wasting the time, the CPU will now begin to execute the second program. Meanwhile the first program once it receives the required data, will again get the CPU time and get executed blocking or pausing the execution of the second program. After the completion of the first program, the second program is executed from where it was paused. The concept was introduced to maximize the CPU usage.

Multiprocessing

The term multiprocessing is introduced around the modern times when they started to use more than one processor in a single computer (Remember the terms like dual core, quad core, octa core processor). These processors share some things in common like memory, peripherals etc. By sharing the memory and peripherals they are able execute different task simultaneously.

In general,

Multiprogramming: takes place in a system where it has a single processor.

Multitasking: single processor or sometimes multiprocessor **Multiprocessing**: multiprocessor.

Feature	Multiprogramming	Multitasking	Multithreading	Multiprocessing
Definition	Running multiple programs on a single CPU	Running multiple tasks (applications) on a single CPU	Running multiple threads within a single task (application)	Running multiple processes on multiple CPUs (or cores)
Resource Sharing	Resources (CPU, memory) are shared among programs	Resources (CPU, memory) are shared among tasks	Resources (CPU, memory) are shared among threads	Each process has its own set of resources (CPU, memory)
Scheduling	Uses round-robin or priority-based scheduling to allocate CPU time to programs	Uses priority-based or time-slicing scheduling to allocate CPU time to tasks	Uses priority-based or time-slicing scheduling to allocate CPU time to threads	Each process can have its own scheduling algorithm
Memory Management	Each program has its own memory space	Each task has its own memory space	Threads share memory space within a task	Each process has its own memory space
Context Switching	Requires a context switch to switch between programs	Requires a context switch to switch between tasks	Requires a context switch to switch between threads	Requires a context switch to switch between processes
Inter-Process Communication (IPC)	Uses message passing or shared memory for IPC	Uses message passing or shared memory for IPC	Uses thread synchronization mechanisms (e.g., locks, semaphores) for IPC	Uses inter-process communication mechanisms (e.g., pipes, sockets) for IPC

Scheduling

Scheduling is the method by which task specified by some means is assigned to resources that complete the task. The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Objective and criteria of process scheduling

- Max CPU utilization [Keep CPU as busy as possible] ▪ Fair allocation of CPU.
- Max throughput [Number of processes that complete their execution per time unit]
- Min turnaround time [Time taken by a process to finish execution]
- Min waiting time [Time a process waits in ready queue] ▪ Min response time

Preemptive vs Non-Preemptive Scheduling

1. Preemptive Scheduling:

Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on preemptive scheduling are: Round Robin (RR), Shortest Job First (SJF basically non preemptive) and Priority (non-preemptive version), etc.

2. Non-Preemptive Scheduling:

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Algorithms based on preemptive scheduling are: Shortest Remaining Time First (SRTF), Priority (preemptive version), etc.

Comparison of preemptive and Non preemptive scheduling

BASIS	PREEMPTIVE SCHEDULING	NON PREEMPTIVE SCHEDULING
Basic	The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process cannot be interrupted till it terminates or switches to waiting state.
Starvation	If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Overhead	Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Flexibility	Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Cost	Preemptive scheduling is cost associated.	Non-preemptive scheduling is not cost associative.