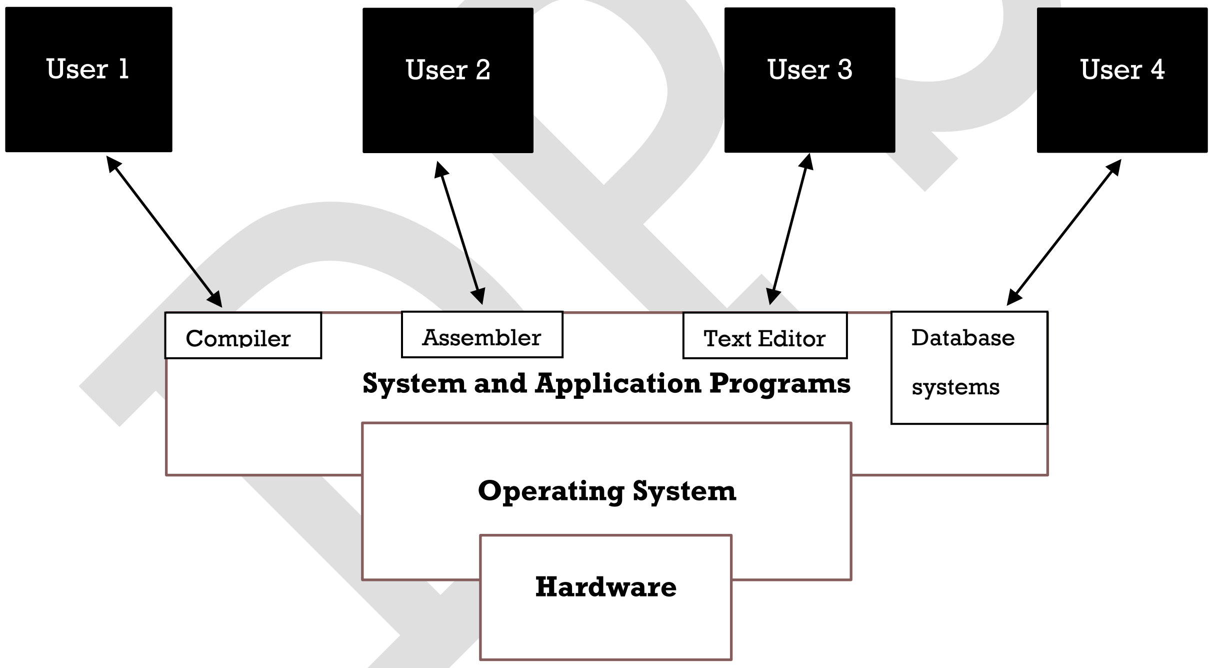
OPERATING SYSTEM

What is an Operating System?

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Operating System is a fully integrated set of specialized programs that handle all the operations of the computer. It controls and monitors the execution of all other programs that reside in the computer, which also includes application programs and other system software of the computer. Examples of Operating Systems are Windows, Linux, Mac OS, etc.



Why do we need an operating system?

The operating system helps in improving the computer software as well as hardware. Without OS, it became very difficult for any application to be user-friendly. Operating System provides a user with an interface that makes any application attractive and user-friendly. The operating System comes with a large number of device drivers that makes OS services reachable to the hardware environment. Each and every application present in the system requires the Operating System. The operating system works as a communication channel between system hardware and system software.

- *I f we don't have an operating system we have to write entire program to invoke I/O devices.*
- *It is one of the most important parts of the system and hence it is present in every device, whether large or small device*.

Objectives and Goals of Operating Systems

- Convenient to use: One of the objectives is to make the computer system more convenient to use in an efficient manner.
- User Friendly: To make the computer system more interactive with a more convenient interface for the users.
- Easy Access: To provide easy access to users for using resources by acting as an intermediary between the hardware and its users.
- Management of Resources: For managing the resources of a computer in a better and faster way.
- Controls and Monitoring: By keeping track of who is using which resource, granting resource requests, and mediating conflicting requests from different programs and users.
- Fair Sharing of Resources: Providing efficient and fair sharing of resources between the users and programs.

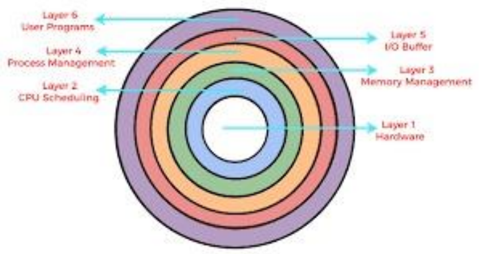
Functions of the Operating System

- Resource Management: The operating system manages and allocates memory, CPU time, and other hardware resources among the various programs and processes running on the computer.
- Process Management: The operating system is responsible for starting, stopping, and managing processes and programs. It also controls the scheduling of processes and allocates resources to them.
- Memory Management: The operating system manages the computer's primary memory and provides mechanisms for optimizing memory usage.
- Security: The operating system provides a secure environment for the user, applications, and data by implementing security policies and mechanisms such as access controls and encryption.
- Job Accounting: It keeps track of time and resources used by various jobs or users.
- File Management: The operating system is responsible for organizing and managing the file system, including the creation, deletion, and manipulation of files and directories.
- Device Management: The operating system manages input/output devices such as printers, keyboards, mice, and displays. It provides the necessary drivers and interfaces to enable communication between the devices and the computer.
- Networking: The operating system provides networking capabilities such as establishing and managing network connections, handling network protocols, and sharing resources such as printers and files over a network.

- User Interface: The operating system provides a user interface that enables users to interact with the computer system. This can be a Graphical User Interface (GUI), a Command-Line Interface (CLI), or a combination of both.
- Backup and Recovery: The operating system provides mechanisms for backing up data and recovering it in case of system failures, errors, or disasters.
- Virtualization: The operating system provides virtualization capabilities that allow multiple operating systems or applications to run on a single physical machine. This can enable efficient use of resources and flexibility in managing workloads.
- Performance Monitoring: The operating system provides tools for monitoring and optimizing system performance, including identifying bottlenecks, optimizing resource usage, and analyzing system logs and metrics.
- Time-Sharing: The operating system enables multiple users to share a computer system and its resources simultaneously by providing time-sharing mechanisms that allocate resources fairly and efficiently.
- System Calls: The operating system provides a set of system calls that enable applications to interact with the operating system and access its resources. System calls provide a standardized interface between applications and the operating system, enabling portability and compatibility across different hardware and software platforms.
- Error-detecting Aids: These contain methods that include the production of dumps, traces, error messages, and other debugging and error-detecting methods.
- Log management: Log management is the practice of continuously gathering, storing, processing, synthesizing and analyzing data from disparate programs and applications in order to optimize system performance, identify technical issues, better manage resources, strengthen security and improve compliance.

Layered Operating System

Layered Structure is a type of system structure in which the different services of the operating system are split into various layers, where each layer has a specific well-defined task to perform.



Generations of Operating System

The First Generation (1945 - 1955): Vacuum Tubes and Plugboards

Digital computers were not constructed until the second world war. Calculating engines with mechanical relays were built at that time. However, the mechanical relays were very slow and were later replaced with vacuum tubes. These machines were enormous but were still very slow.

These early computers were designed, built and maintained by a single group of people. Programming languages were unknown and there were no operating systems so all the programming was done in machine language. All the problems were simple numerical calculations.

By the 1950's punch cards were introduced and this improved the computer system. Instead of using plugboards, programs were written on cards and read into the system.

- Enormous
- Very large
- Very slow
- No OS
- Machine language]
- Simple numeric operations

The Second Generation (1955 - 1965): Transistors and Batch Systems

Transistors led to the development of the computer systems that could be manufactured and sold to paying customers. These machines were known as mainframes and were locked in air-conditioned computer rooms with staff to operate them.

The Batch System was introduced to reduce the wasted time in the computer. A tray full of jobs was collected in the input room and read into the magnetic tape. After that, the tape was rewound and mounted on a tape drive. Then the batch operating system was loaded in which read the first job from the tape and ran it. The output was written on the second tape. After the whole batch was done, the input and output tapes were removed and the output tape was printed.

- Comparatively small
- Paid systems available
- Batch OS

The Third Generation (1965 - 1980): Integrated Circuits and Multiprogramming

Until the 1960's, there were two types of computer systems i.e the scientific and the commercial computers. These were combined by IBM in the System/360. This used integrated circuits and provided a major price and performance advantage over the second generation systems

The third generation operating systems also introduced multiprogramming. This meant that the processor was not idle while a job was completing its I/O operation. Another job was scheduled on the processor so that its time would not be wasted.

- Small in size
- Computers are categorized into two scientific and general purpose
- Multiprogramming
- Reduction in CPU idle time

The Fourth Generation (1980 - Present): Personal Computers

Personal Computers were easy to create with the development of large-scale integrated circuits. These were chips containing thousands of transistors on a square centimeter of silicon. Because of these, microcomputers were much cheaper than minicomputers and that made it possible for a single individual to own one of them.

The advent of personal computers also led to the growth of networks. This created network operating systems and distributed operating systems. The users were aware of a network while using a network operating system and could log in to remote machines and copy files from one machine to another.

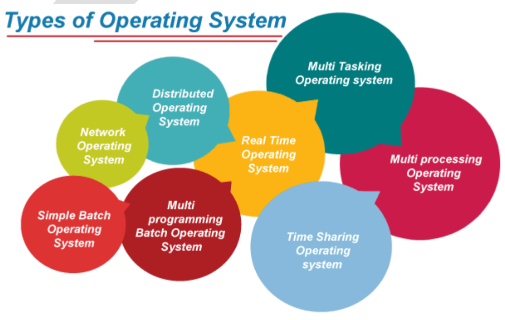
- VLSIC
- Easily available
- User-friendly
- Multiprogramming and multitasking

Fifth gen

- Advanced ai
- NLP
- Neural network

Types of Operating System

- ➢ Batch Operating System
- ➢ Multi-Programming System
- ➢ Multi-Processing System
- ➢ Multi-Tasking Operating System
- ➢ Time-Sharing Operating System
- ➢ Distributed Operating System
- ➢ Network Operating System
- ➢ Real-Time Operating System

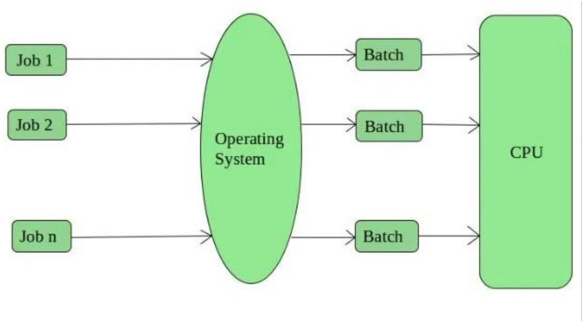

Types of Operating System
Multi Tasking Operating system
Distributed Operating System
Network Operating System
Real Time Operating System
Multi processing Operating System
Simple Batch Operating System
Multi programming Batch Operating System
Time Sharing Operating system

1. Batch Operating System

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

Advantages of Batch Operating System

- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in the queue.
- Multiple users can share the batch systems.
- The idle time for the 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 sometimes costly.
- The other jobs will have to wait for an unknown time if any job fails.

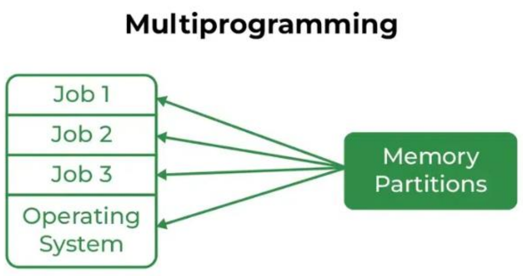
Examples of Batch Operating Systems: Payroll Systems, Bank Statements, etc.

2. Multi-Programming Operating System

Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. This is basically used for better execution of resources.

Advantages of Multi-Programming Operating System

- Multi Programming increases the Throughput of the System.
- It helps in reducing the response time.



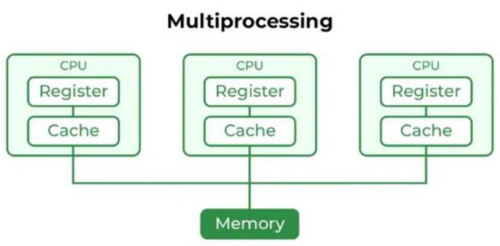
Disadvantages of Multi-Programming Operating System

- There is not any facility for user interaction of system resources with the system.

3. Multi-Processing Operating System

Multi-Processing Operating System is a type of Operating System in which more than one CPU is used for the execution of resources. It betters the throughput of the System.

Advantages of Multi-Processing Operating System



- It increases the throughput of the system.
- As it has several processors, so, if one processor fails, we can proceed with another processor.

Disadvantages of Multi-Processing Operating System

- Due to the multiple CPU, it can be more complex and somehow difficult to understand.

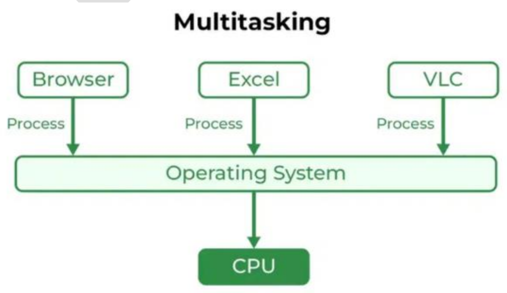
4. Multi-Tasking Operating System

Multitasking Operating System is simply a multiprogramming Operating System with having facility of a Round-Robin Scheduling Algorithm. It can run multiple programs simultaneously.

There are two types of Multi-Tasking Systems which are listed below.

- Primitive Multi-Tasking
- Cooperative Multi-Tasking

Advantages of Multi-Tasking Operating System



- Multiple Programs can be executed simultaneously in Multi-Tasking Operating System.
- It comes with proper memory management.

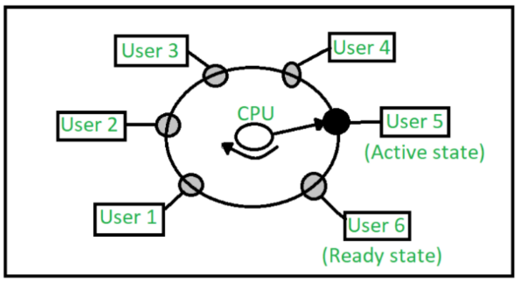
Disadvantages of Multi-Tasking Operating System

- The system gets heated in case of heavy programs multiple times.

5. Time-Sharing Operating Systems

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of the CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.

Advantages of Time-Sharing OS



- Each task gets an equal opportunity.
- Fewer chances of duplication of software.
- CPU idle time can be reduced.
- Resource Sharing: Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.
- Improved Productivity: Time-sharing allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
- Improved User Experience: Time-sharing provides an interactive environment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

Disadvantages of Time-Sharing OS

- Reliability problem.
- One must have to take care of the security and integrity of user programs and data.
- Data communication problem.
- High Overhead: Time-sharing systems have a higher overhead than other operating systems due to the need for scheduling, context switching, and other overheads that come with supporting multiple users.
- Complexity: Time-sharing systems are complex and require advanced software to manage multiple users simultaneously. This complexity increases the chance of bugs and errors.
- Security Risks: With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization to ensure the security of data and software.

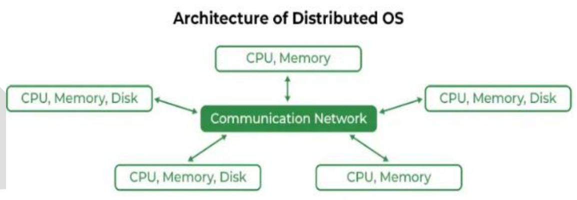
Examples of Time-Sharing OS with explanation

- IBM VM/CMS: IBM VM/CMS is a time-sharing operating system that was first introduced in 1972. It is still in use today, providing a virtual machine environment that allows multiple users to run their own instances of operating systems and applications.
- TSO (Time Sharing Option): TSO is a time-sharing operating system that was first introduced in the 1960s by IBM for the IBM System/360 mainframe computer. It allowed multiple users to access the same computer simultaneously, running their own applications.
- Windows Terminal Services: Windows Terminal Services is a time-sharing operating system that allows multiple users to access a Windows server remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.

6. Distributed Operating System

A distributed operating system is a type of operating system that manages resources and provides services across a network of interconnected computers. Unlike traditional operating systems that primarily focus on a single computer, distributed operating systems coordinate multiple computers to work together as a unified system. This allows for resource sharing, load balancing, fault tolerance, and efficient utilization of distributed computing resources.

Advantages of Distributed Operating System



- Failure of one will not affect the other network communication, as all systems are independent of 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 .

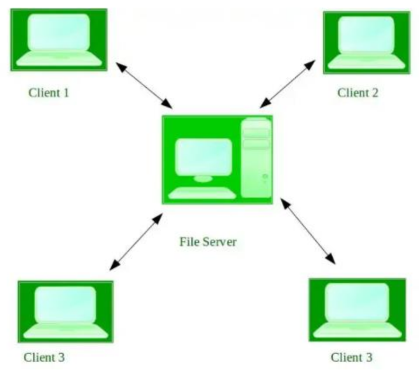
- Failure of the main network will stop the entire communication.
- To establish distributed systems the language is used not well-defined yet.
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well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

Examples of Distributed Operating Systems are LOCUS, etc.

7. Network Operating System

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access to 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 into the system.
- Server access is possible remotely from different locations and types of systems.

Disadvantages of Network Operating System

- Servers are costly.
- User has to depend on a central location for most operations.
- Maintenance and updates are required regularly.
- Examples of Network Operating Systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, BSD, etc.

8. Real-Time Operating System

These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called response time.

Real-time systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Types of Real-Time Operating System.

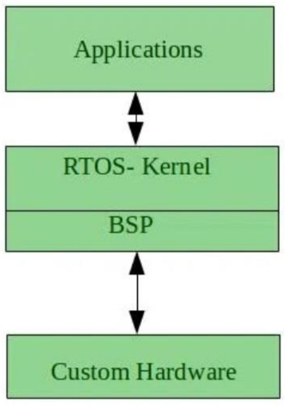
Hard Real-Time Systems:

Hard Real-Time OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of an accident. Virtual memory is rarely found in these systems.

Soft Real-Time Systems:

These OSs are for applications where time-constraint is less strict.

Advantages of RTOS



- Maximum Consumption: Maximum utilization of devices and systems, thus more output from all the resources.
- Task Shifting: The time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds in shifting from one task to another, and in the latest systems, it takes 3 microseconds.
- Focus on Application: Focus on running applications and less importance on applications that are in the queue.
- Real-time operating system in the embedded system: Since the size of programs is small, RTOS can also be used in embedded systems like in transport and others.
- Error Free: These types of systems are error-free.
- Memory Allocation: Memory allocation is best managed in these types of systems.

Disadvantages of RTOS

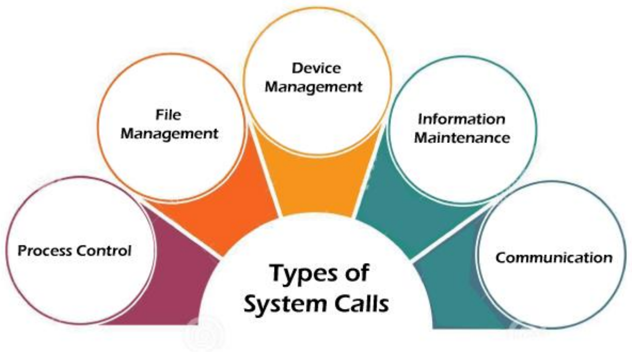
- Limited Tasks: Very few tasks run at the same time and their concentration is very less on a few applications to avoid errors.
- Use heavy system resources: Sometimes the system resources are not so good and they are expensive as well.
- Complex Algorithms: The algorithms are very complex and difficult for the designer to write on.
- Device driver and interrupt signals: It needs specific device drivers and interrupts signal to respond earliest to interrupts.
- Thread Priority: It is not good to set thread priority as these systems are very less prone to switching tasks.

Examples of Real-Time Operating Systems are Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

System calls

A system call is a mechanism used by programs to request services from the operating system (OS)

- Process management
- File management
- Device management
- Information maintenance
- Communication



Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

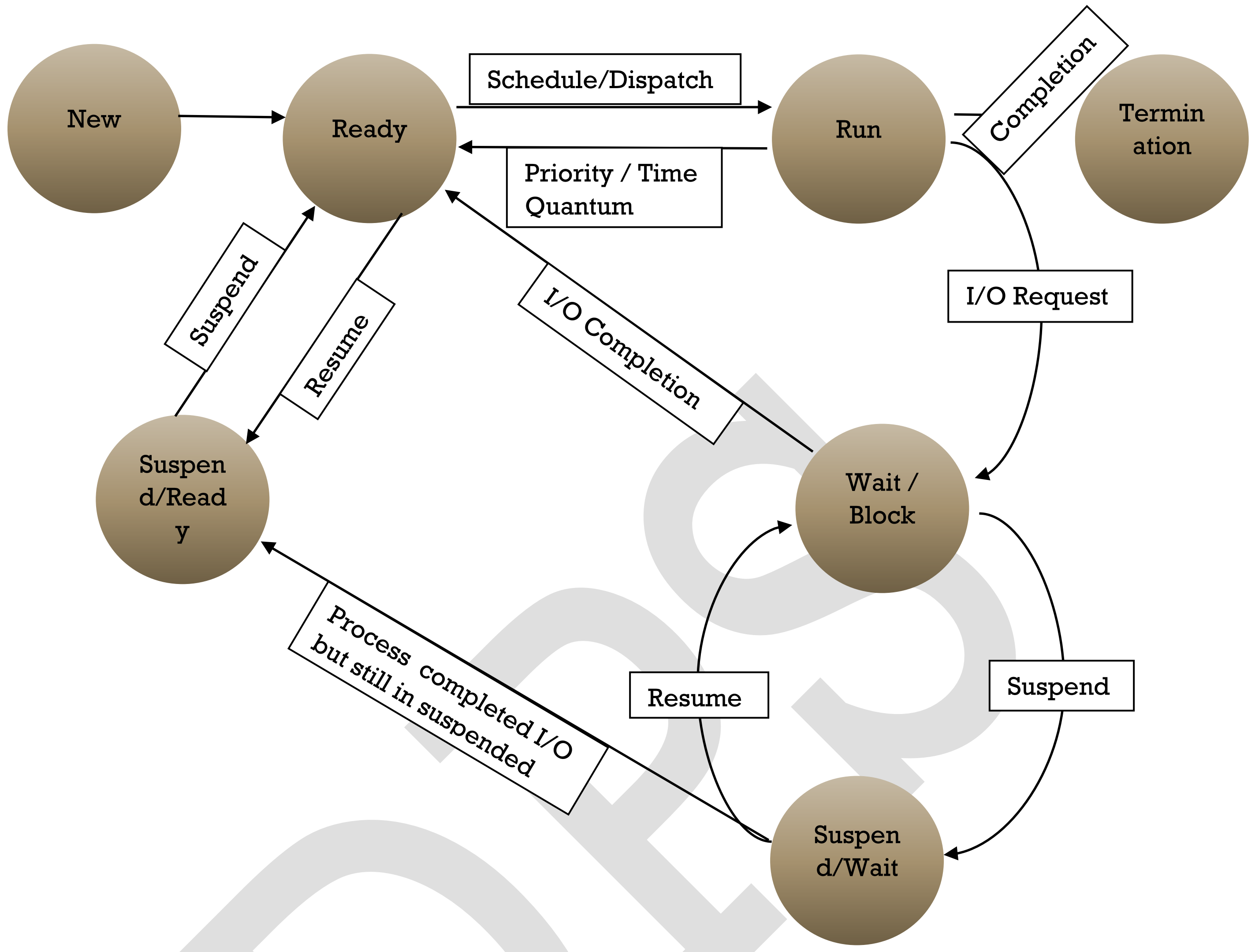
Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Fork – It is a system call which creates a child process which is as same as parents process and the working of both process is same . We can also create many fork .

- *Used for reading sub processes*

Process state

A process has several stages that it passes through from beginning to end. There must be a minimum of five states. Even though during execution, the process could be in one of these states, the names of the states are not standardized. Each process goes through several stages throughout its life cycle.



The states of a process are as follows:

- New (Create): In this step, the process is about to be created but not yet created. It is the program that is present in secondary memory that will be picked up by OS to create the process.
- Ready: New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue called ready queue for ready processes.
- Run: The process is chosen from the ready queue by the CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- Blocked or Wait: Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or waits for the state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.

- Terminated or Completed: Process is killed as well as PCB is deleted. The resources allocated to the process will be released or deallocated.
- Suspend Ready: Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought onto the main memory.
- Suspend wait or suspend blocked: Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.
- CPU and I/O Bound Processes: If the process is intensive in terms of CPU operations, then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.

How does a process move between different states in an operating system?

- A process can move between different states in an operating system based on its execution status and resource availability. Here are some examples of how a process can move between different states:
- New to ready: When a process is created, it is in a new state. It moves to the ready state when the operating system has allocated resources to it and it is ready to be executed.
- Ready to running: When the CPU becomes available, the operating system selects a process from the ready queue depending on various scheduling algorithms and moves it to the running state.
- Running to blocked: When a process needs to wait for an event to occur (I/O operation or system call), it moves to the blocked state. For example, if a process needs to wait for user input, it moves to the blocked state until the user provides the input.
- Running to ready: When a running process is preempted by the operating system, it moves to the ready state. For example, if a higher-priority process becomes ready, the operating system may preempt the running process and move it to the ready state.
- Blocked to ready: When the event a blocked process was waiting for occurs, the process moves to the ready state. For example, if a process was waiting for user input and the input is provided, it moves to the ready state.
- Running to terminated: When a process completes its execution or is terminated by the operating system, it moves to the terminated state.

Virtual machine

A virtual machine is a digital version of a physical computer . we can run different operating system , we can store data , we can compute , connect to network.

Or

A Virtual Machine (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine.

Difference Between Operating System and Kernal

Operating System	**Kernel**
Operating System is a system software.	Kernel is system software which is part of operating system.
Operating System provides interface between user and hardware.	Kernel provides interface between applications and hardware.
It also provides protection and security.	It's main purpose is memory management, disk management, process management and task management.
An operating system is a complete software package that includes a kernel and other system-level components such as device drivers, system libraries, and utilities.	The kernel, on the other hand, is the core of the operating system that manages system resources, such as the CPU, memory, and I/O devices.
the operating system provides a higher-level interface to the user, such as the GUI, command-line interface, and file system.	The kernel provides low-level services, such as memory management, process management, and device management, to other parts of the operating system
the operating system is a more complex system that includes a large number of components.	The kernel is a relatively small and simple component of the operating system,
the operating system provides a more general-purpose interface that	The kernel is often customized for specific hardware platforms or applications,

Operating System	**Kernel**
can be used on a wide range of hardware platforms.	
The operating system is designed to be portable across different hardware platforms,	the kernel is often platform-specific.
All system needs operating system to run.	All operating systems need kernel to run.
Type of operating system includes single and multiuser OS, multiprocessor OS, Realtime OS, Distributed OS.	Type of kernel includes Monolithic and Micro kernel.
It is the first program to load when computer boots up.	It is the first program to load when operating system loads.

Difference between User Mode and Kernel Mode

Terms	**User Mode**	**Kernel Mode**
Definition	User Mode is a restricted mode, which the application programs are executing and starts.	Kernel Mode is the privileged mode, which the computer enters when accessing hardware resources.
Modes	User Mode is considered as the slave mode or the restricted mode.	Kernel mode is the system mode, master mode or the privileged mode.
Address Space	In User mode, a process gets its own address space.	In Kernel Mode, processes get a single address space.
Interruptions	In User Mode, if an interrupt occurs, only one process fails.	In Kernel Mode, if an interrupt occurs, the whole operating system might fail.

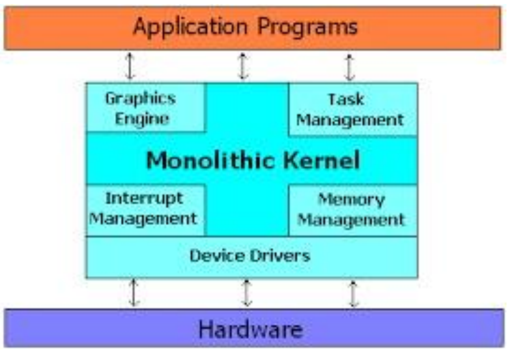
Restrictions	In user mode, there are restrictions to access kernel programs. Cannot access them directly.	In kernel mode, both user programs and kernel programs can access.

Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	The process is isolated.	Threads share memory.
8.	The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
9.	Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
10.	If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.

S.NO	Process	Thread
11.	The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
12.	Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behaviour of the other threads of the process.
13.	A system call is involved in it.	No system call is involved, it is created using APIs.
14.	The process does not share data with each other.	Threads share data with each other.

Monolithic kernel



A monolithic kernel is an operating system architecture where the entire operating system is working in kernel space. The monolithic model differs from other operating system architectures, such as the microkernel architecture, in that it alone defines a high-level virtual interface over computer hardware.

Example – linux , windows

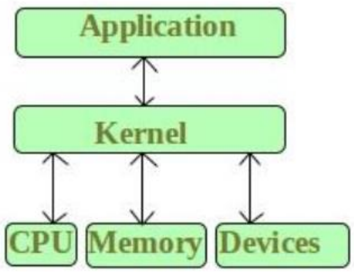
Advantages of Monolithic Kernel

- The execution of the monolithic kernel is quite fast as the services such as memory management, file management, process scheduling, etc., are implemented under the same address space.
- A process runs completely in single address space in the monolithic kernel.
- The monolithic kernel is a static single binary file.

Disadvantages of Monolithic Kernel

- If any service fails in the monolithic kernel, it leads to the failure of the entire system.
- The entire operating system needs to be modified by the user to add any new service

Microkernel

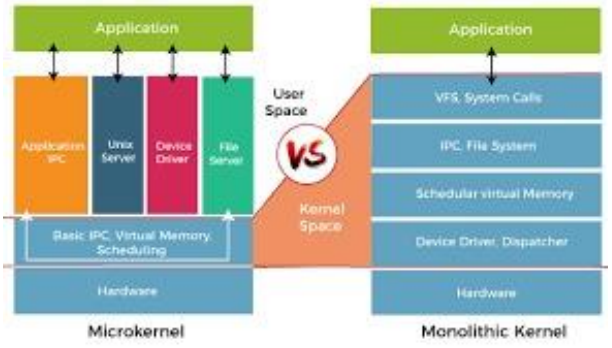


A microkernel is a type of operating system kernel that is designed to provide only the most basic services required for an operating system to function, such as memory management and process scheduling. Other services, such as device drivers and file systems, are implemented as user-level processes that communicate with the microkernel via message passing. This design allows the operating system to be more modular and flexible than traditional monolithic kernels, which implement all operating system services in kernel space.

The main advantage of a microkernel architecture is that it provides a more secure and stable operating system. Since only the most essential services run in kernel space, the attack surface of the operating system is reduced, making it more difficult for an attacker to exploit vulnerabilities. Additionally, if a user-level process crashes, it will not affect the stability of the entire system, since the microkernel is responsible only for managing processes and memory.

Another advantage of a microkernel architecture is that it makes the operating system more modular and flexible. Since services are implemented as user-level processes, it is easier to add, remove, or replace services without affecting other parts of the system. This makes it easier to customize the operating system to meet specific requirements.

However, there are also some disadvantages to a microkernel architecture. One major disadvantage is that message passing between user-level processes can be slower than direct system calls in a monolithic kernel. This can affect the performance of the operating system, especially in high-performance applications. Additionally, the modular design of a microkernel can lead to increased complexity, which can make it more difficult to develop and maintain the operating system.



Difference between monolithic kernal and microkernal

Basics	**Micro Kernel**	**Monolithic Kernel**
Size	Smaller in	Larger as OS and user both lie in the same address space.
Execution	Slower	Faster
Extendible	Easily extendible	Complex to extend
Security	If the service crashes then there is no effect on working on the microkernel.	If the process/service crashes, the whole system crashes as both user and OS were in the same address space.
Code	More code is required to write a microkernel.	Less code is required to write a monolithic kernel.
Examples	L4Linux, macOS	Windows, Linux BSD
Security	More secure because only essential services run in kernel mode	Susceptible to security vulnerabilities due to the amount of code running in kernel mode
Platform independence	More portable because most drivers and services run in user space	Less portable due to direct hardware access
Communication	Message passing between user-space servers	Direct function calls within kernel

Basics	Micro Kernel	Monolithic Kernel
Performance	Lower due to message passing and more overhead	High due to direct function calls and less overhead

Case Study: Evolution and Impact of the Linux Operating System

Introduction:

The Linux operating system is a prominent example of open-source software that has significantly impacted the world of computing. Developed initially by Linus Torvalds in 1991, Linux has evolved from a small project to a powerful and versatile operating system that powers a wide range of devices, from servers and supercomputers to smartphones and embedded systems. This case study explores the evolution, features, and impact of Linux on the technology landscape.

Evolution and Development:

1. Genesis and Growth: Linux originated as a personal project by Linus Torvalds, a Finnish computer science student. He aimed to create an operating system kernel that could run on his Intel 80386-based personal computer. Torvalds released the first version of the Linux kernel (0.01) in 1991, making it available under the GNU General Public License (GPL).

2. Collaborative Development Model: One of the most remarkable aspects of Linux is its collaborative development model. The Linux community, comprising developers from around the world, contributes to the kernel's development. This decentralized approach allows for rapid innovation, bug fixes, and the inclusion of new features.

3. Distributions and Variants: Over time, various distributions (distros) of Linux have emerged, each offering a unique combination of the Linux kernel and different software packages. Examples include Ubuntu, Debian, Red Hat, Fedora, and CentOS. These distros cater to different user needs, such as desktop computing, server hosting, and embedded systems.

Key Features:

1. Open Source Philosophy: Linux's open-source nature encourages collaboration, transparency, and user participation. Anyone can access, modify, and distribute the source code, leading to a vibrant ecosystem of developers and users.

2. Stability and Reliability: Linux is known for its stability, scalability, and robustness. It is widely used in server environments where uptime is critical.

3. Customizability: Users can tailor their Linux systems to their specific needs, installing only the software and components they require. This minimizes resource usage and enhances performance.

4. Security: Linux benefits from its open-source nature, allowing security vulnerabilities to be identified and patched quickly. Additionally, the separation of user and administrator privileges contributes to a more secure computing environment.

5. Versatility: Linux runs on various hardware architectures, from smartphones and embedded devices to mainframes and supercomputers. This versatility contributes to its widespread adoption.

Impact:

1. Server Dominance: Linux powers a significant portion of the server market. Major internet services, cloud providers, and data centers rely on Linux for its stability and performance.

2. Embedded Systems: Linux is used in a wide range of embedded systems, from smart TVs and routers to automotive infotainment systems. Its flexibility and customizability make it a preferred choice for such applications.

3. Smartphones and Devices: Android, the most popular mobile operating system, is based on the Linux kernel. Linux also forms the basis for various IoT (Internet of Things) devices.

4. Education and Learning: Linux has played a crucial role in education by providing students and enthusiasts with free access to a powerful operating system. Many universities and institutions teach operating system concepts using Linux.

5. Open Source Movement: Linux's success has helped promote the broader open-source software movement, emphasizing collaboration, transparency, and shared development.

Conclusion:

The Linux operating system stands as a testament to the power of open-source collaboration and the impact of a well-designed, versatile, and customizable software platform. Its journey from a personal project to a global force has shaped the technology landscape, influencing everything from servers and embedded systems to education and software development methodologies. As Linux continues to evolve, its significance in the world of computing remains profound.

Case Study: Evolution of the Windows Operating System

Introduction:

Windows is a series of operating systems developed by Microsoft Corporation. It has played a pivotal role in shaping the personal computing landscape and has undergone several iterations since its inception. This case study provides an overview of the evolution of the Windows operating system, highlighting key versions, features, successes, and challenges.

1. Windows 1.0 (1985):

Windows 1.0 marked Microsoft's first attempt at creating a graphical user interface (GUI) for MS-DOS. It introduced features like overlapping windows, a mouse-driven interface, and basic applications like Notepad and Paint. However, it had limited success due to hardware limitations and the lack of software compatibility.

2. Windows 3.0 (1990) and Windows 3.1 (1992):

These versions significantly improved the user interface, introduced Program Manager and File Manager, and gained popularity through compatibility with a wider range of software. Windows 3.x established Microsoft as a leading player in GUI-based operating systems.

3. Windows 95 (1995):

Windows 95 was a major milestone, featuring the iconic Start button, Taskbar, and the concept of plug-and-play hardware installation. It brought 32-bit architecture, preemptive multitasking, and long filenames. Windows 95 captured significant market share, establishing Microsoft's dominance in the operating system market.

4. Windows 98 (1998) and Windows ME (2000):

Windows 98 refined the Windows 95 interface and introduced features like improved USB support and integrated web browsing. Windows ME (Millennium Edition) focused on multimedia and home networking enhancements, but stability issues hindered its success.

5. Windows 2000 (2000) and Windows XP (2001):

Windows 2000 targeted business users with enhanced stability, security, and support for modern hardware. Windows XP merged the consumer and business lines, featuring a more visually appealing interface and improved networking. XP gained immense popularity and longevity, becoming one of the most widely used versions.

6. Windows Vista (2007):

Windows Vista aimed to improve security and the user experience with features like Aero interface and User Account Control. However, it faced criticism for its resource-intensive nature and compatibility issues, leading many users to stick with Windows XP.

7. Windows 7 (2009):

Windows 7 addressed many of Vista's shortcomings, delivering improved performance, streamlined UI elements, and enhanced compatibility. It became one of Microsoft's most well-received operating systems.

8. Windows 8 (2012) and Windows 8.1 (2013):

Windows 8 introduced a radical departure from the traditional desktop interface with the introduction of the Start Screen, optimized for touch devices. However, its dual-interface design was polarizing, with many users finding it challenging to use on non-touch devices. Windows 8.1 brought back some familiar elements and introduced improvements.

9. Windows 10 (2015):

Windows 10 focused on unifying the user experience across devices, introducing the concept of "Universal Windows Platform" apps. It also marked a shift to a continuous development model with regular updates. Windows 10 has been well-received for its performance, security enhancements, and integration of virtual assistants like Cortana.

10. Windows 11 (2021):

Windows 11 introduced a centered taskbar, redesigned Start Menu, and visual enhancements. It emphasized productivity and gaming, with features like Snap Layouts and DirectStorage for faster game loading. Windows 11 continued the trend of integrating Microsoft's ecosystem and enhancing touch and pen input.

Conclusion:

The evolution of the Windows operating system has been characterized by iterative improvements, changing UI paradigms, and adapting to the evolving needs of users and technology. Throughout its history, Windows has maintained its position as one of the most widely used and influential operating systems in the world, shaping the way people interact with computers and digital technology.