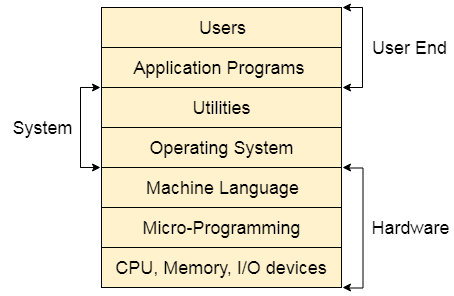
**UNIT-1:**

**INTRODUCTION:**

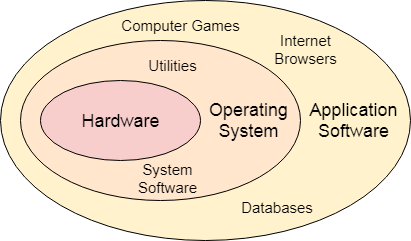
1. Computer system:

* Process is a program which is in execution.
* A Computer System consists of:

1. Users (people who are using the computer)
2. Application Programs (Compilers, Databases, Games, Video player, Browsers, etc.)
3. System Programs (Shells, Editors, Compilers, etc.)
4. Operating System ( A special program which acts as an interface between user and hardware )
5. Hardware ( CPU, Disks, Memory, etc)



1. Operating system:

* Operating System can be defined as an interface between user and the hardware. It provides an environment to the user so that, the user can perform its task in convenient and efficient way.
* In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.
* We need a system which can act as an intermediary and manage all the processes and resources present in the system.
* 
* An Operating System can be defined as an interface between user and hardware. It is responsible for the execution of all the processes, Resource Allocation, [CPU](https://www.tpointtech.com/cpu-full-form) management, File Management and many other tasks.
* The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.
* What does an operating system do?

1. Process Management
2. Process Synchronization
3. Memory Management
4. CPU Scheduling
5. File Management
6. Security

* Function of Operating system:

1. **Process Management**

* Manages processes in the system (creation, execution, termination).
* Handles multitasking (running multiple processes at the same time).
* Provides mechanisms like scheduling and synchronization.

1. **Memory Management**

* Allocates and deallocates memory to programs when needed.
* Keeps track of each byte in a computer’s memory.
* Ensures no process uses another process's memory (protection).

1. **File System Management**

* Manages files and directories on storage devices.
* Handles file creation, deletion, reading, writing, and permission settings.
* Provides structure (hierarchical format) for file organization.

1. **Device Management**

* Manages all hardware devices (input/output devices).
* Controls communication between devices and programs.
* Uses drivers to ensure hardware and software compatibility.

1. **Security and Access Control**

* Protects data and resources from unauthorized access.
* Implements password protection, encryption, and permission settings.

1. **User Interface**

* Provides a way for users to interact with the system (CLI or GUI).
* Makes system usage easier and efficient.

1. **Resource Allocation**

* Allocates CPU, memory, files, and I/O devices to various tasks.
* Ensures fair distribution of resources to all processes.

1. **Error Detection and Handling**

* Detects errors in hardware, software, and memory.
* Provides mechanisms to recover from failures.

1. **Networking**

* Manages network connections and data exchange.
* Provides networking protocols and services like file sharing.

1. **System Performance Monitoring**

* Tracks system performance (CPU utilization, memory usage).
* Helps in improving system efficiency.

1. Goals of Operating system:
   1. Convenience

* To make the computer system easy to use for the user.
* Provides an interface between the user and hardware.
  1. Efficiency
* To utilize system resources (CPU, Memory, I/O Devices) in the best possible way.
* Ensures maximum output with minimum resources.
  1. Ability to Evolve
* Should be designed in a way that it can be upgraded and modified easily.
* Supports new hardware and software updates.
  1. Resource Management
* Manages all hardware and software resources efficiently.
* Allocates resources fairly to all users and programs.
  1. Security and Protection
* Protects data and resources from unauthorized access.
* Provides user authentication, file permissions, and encryption.
  1. Error Detection and Recovery
* Detects errors in hardware and software.
* Tries to recover and continue smooth functioning.
  1. Multiprogramming
* Supports running multiple programs at the same time.
* Helps in better CPU utilization.
  1. Fairness
* Ensures that all processes and users get fair access to system resources.

1. User VS system view of operating system:
2. User View of Operating System:

* The Operating System is an interface, hides the details which must be performed and present a virtual machine to the user that makes it easier to use.
* Operating System provides the following services to the user.
* Execution of a program
* Access to I/O devices
* Controlled access to files
* Error detection (Hardware failures, and software errors).

1. Hardware View of Operating System:

* The Operating System manages the resources efficiently in order to offer the services to the user programs.
* Operating System acts as a resource manager:
* Allocation of resources
* Controlling the execution of a program
* Control the operations of I/O devices
* Protection of resources
* Monitors the data

1. System View of Operating System:

* Operating System is a program that functions in the same way as other programs. It is a set of instructions that are executed by the processor.
* Operating System acts as a program to perform the following.
* Hardware upgrades
* New services
* Fixes the issues of resources
* Controls the user and hardware operations.

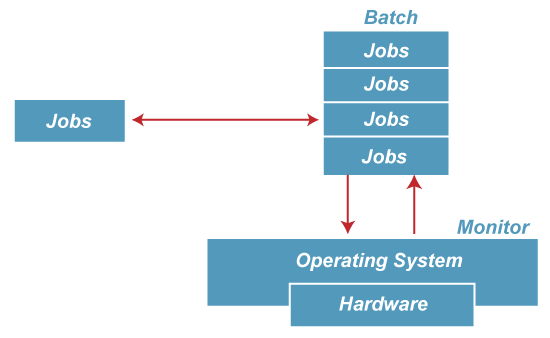
1. Advantages and disadvantages of operating system:

* Advantages are-
  1. **User-Friendly Interface** – Provides an easy-to-use interface (GUI) for users to interact with the computer.
  2. **Multitasking** – Allows multiple applications to run simultaneously.
  3. **Resource Management** – Efficiently manages CPU, memory, storage, and input/output devices.
  4. **Security & Protection** – Provides user authentication, data encryption, and access control.
  5. **Device Management** – Manages peripheral devices like printers, scanners, and USBs.
  6. **Process Scheduling** – Allocates CPU time to multiple processes for smooth execution.
  7. **File Management** – Helps in creating, modifying, deleting, and organizing files.
  8. **Error Handling** – Detects and handles system errors, preventing crashes.
  9. **Networking Support** – Allows multiple systems to connect and communicate over networks.
  10. **Hardware Abstraction** – Provides an abstraction layer to interact with hardware without complexity.
* **Disadvantages are**-
  1. **System Crash** – A failure in the OS can cause the whole system to crash.
  2. **Security Vulnerabilities** – OS can be targeted by viruses, malware, and hackers.
  3. **Complexity** – Some operating systems have a steep learning curve.
  4. **Hardware Dependency** – Some OS are designed for specific hardware.
  5. **High Cost** – Commercial OS like Windows and macOS can be expensive.
  6. **Resource Consumption** – Some OS require high RAM and storage, affecting performance on low-end devices.
  7. **Software Compatibility Issues** – Some software may not be compatible with all OS.
  8. **Frequent Updates** – Regular updates may consume bandwidth and time.
  9. **Limited Customization** – Closed-source OS have limited flexibility in modification.
* **Applications are**-
  1. **Personal Computers** – Used in desktops and laptops (e.g., Windows, macOS, Linux).
  2. **Mobile Devices** – Powers smartphones and tablets (e.g., Android, iOS).
  3. **Servers** – Manages web, database, and cloud servers (e.g., Linux, Windows Server).
  4. **Embedded Systems** – Used in smart devices, ATMs, and IoT devices (e.g., RTOS).
  5. **Supercomputers** – Used in high-performance computing (e.g., Unix, Linux).
  6. **Gaming Consoles** – Used in PlayStation, Xbox, and Nintendo Switch.
  7. **Industrial Control Systems** – Used in automation and robotics.
  8. **Healthcare Systems** – Manages medical equipment and hospital databases.
  9. **Banking & Finance** – Used in ATMs and online banking applications.
  10. **Networking Devices** – Used in routers, modems, and switches.

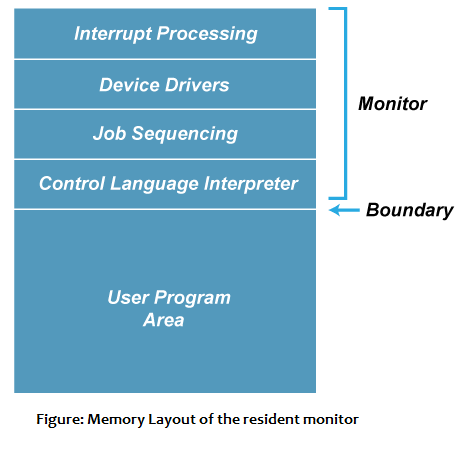
1. Types of operating system:

### Batch Operating System

* In the 1970s, Batch processing was viral.
* In this technique, similar types of jobs were batched together and executed in time. People were used to having a single computer, which was called a mainframe.
* In the Batch operating system, access is given to more than one person; they submit their respective jobs to the system for execution.
* The system puts all of the jobs in a queue on the basis of first come, first serve and then executes the jobs one by one. The users collect their respective output when all the jobs get executed.



* The purpose of this operating system was mainly to transfer control from one job to another as soon as the job was completed. It contained a small set of programs called the resident monitor that always resided in one part of the main memory. The remaining part is used for servicing jobs.



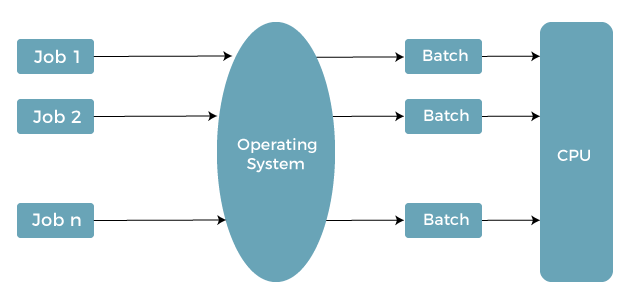
### Advantages of Batch OS

* It has increased efficiency, like a resident monitor that eliminates CPU idle time between jobs to enable it to smoothen the switching of tasks.
* The automatic handling of the processing of jobs means that users should not take control to intervene in whatever is going on while running or executing tasks.
* The resources, such as the CPU and the memory, are utilized and not left idle.
* It goes well with the payroll as well because it keeps processing jobs in batches without any time wastage and hard work.
* It can also perform even very huge complex jobs without any intermission because it simply streams without any intermission, even when the task is excessively intensive.
* Group jobs as 'batches' remove most of the manual setup that had to be done in between individual tasks, thereby saving time.
* It logs and deals with errors at the time when the batch is over. This allows the system to run even without intermissions.

### 🡺Disadvantages of Batch OS

1. **Starvation**

* Batch processing suffers from starvation.
* **For Example:**



* There are five jobs J1, J2, J3, J4, and J5, present in the batch. If the execution time of J1 is very high, then the other four jobs will never be executed, or they will have to wait for a very long time. Hence the other processes get starved.

1. **Not Interactive**

* Batch Processing is not suitable for jobs that are dependent on the user's input. If a job requires the input of two numbers from the console, then it will never get it in the batch processing scenario since the user is not present at the time of execution.

1. **Delayed Output**

* Since the jobs are submitted in batches, the output is not produced in time. Such a condition can be rather inconvenient for time-critical jobs.

1. **Difficult to Debug**

* An error is found only after the entire batch has been processed, which makes it even harder to locate and fix an issue in real-time.

1. **It Requires Knowledge of Job Scheduling**

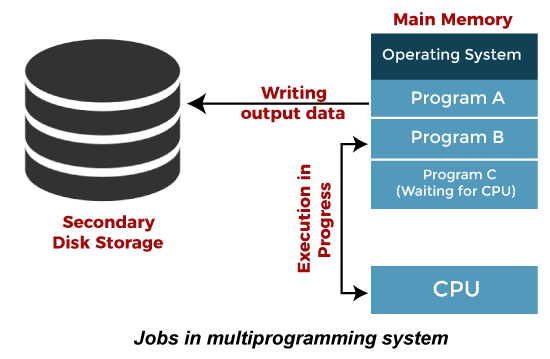
* The users or the system administrator should know well about the behavior of the system as well as dependencies among tasks.

1. **Large Jobs Cause Delays**

* If a batch contains a large job, then problems may occur because the processing of all the subsequent jobs is delayed. This, therefore, slows down the overall system performance.

## Multiprogramming Operating System

* Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.
* In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



### Advantages of Multiprogramming OS

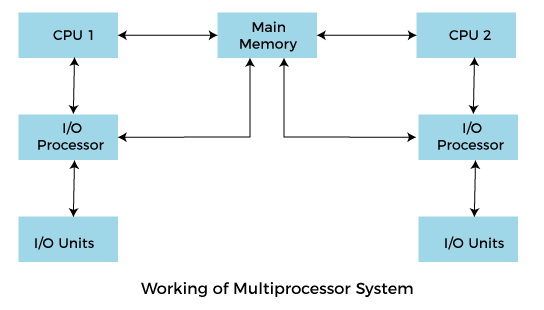
* Throughout the system, it increased as the CPU always had one program to execute.
* Response time can also be reduced.
* Multiprogramming maximizes the utilization of resources like memory, I/O devices, and processing power since more than one program can be kept alive at any time.
* Since several jobs are being processed in parallel, significantly more tasks could be completed within a certain amount of time, thus enhancing the overall throughput of the system.
* During times when a program is waiting for I/O operations, the processor does not go idle since it jumps on to another task to continue processing.
* The system can support both short and long tasks to be executed in parallel, which makes for a more dynamic and productive processing environment.

### Disadvantages of Multiprogramming OS

* Multiprogramming systems provide an environment in which various systems resources are used efficiently, but they do not provide any user interaction with the computer system.
* Multiple programs increase system complexity, as the operating system needs to manage multiple processes, memory management, and scheduling.
* It requires more memory compared to less sophisticated operating systems because multiple programs run in memory simultaneously.
* The operating system has to switch between the running processes continuously, and this leads to scheduling overhead and reduces performance.
* As several concurrent operations access shared resources simultaneously, the system is likely to experience deadlocks: two or more processes waiting for each other for further actions.
* Resource contention due to competition for scarce resources could degrade performance.

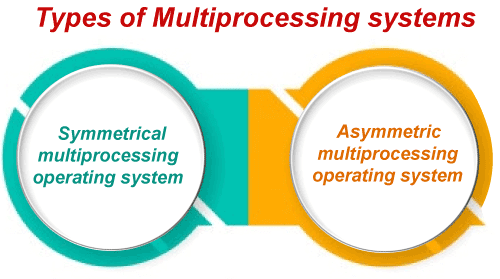
## Multiprocessing Operating System

* In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.
* In multiprocessing, several processors execute procedures in collaboration. You can even split a procedure into parts that run in parallel.
* Since several processors work concurrently, more tasks are completed faster.
* Multiprocessing enhances the speed at which the system executes since tasks scatter the processors, minimizing the waiting period for procedures.
* If one processor is faulty, the others will continue running, hence enhancing system reliability.
* Such an operating system is suited for highly computationally intensive environments needing high processing power, for example, scientific simulations, real-time data processing, and server environments



* **Working**

In a Multiprocessing OS, concurrent execution of processes by multiple processors is done. The processes are divided and allotted to different processors for parallel computing. Thus, system performance is enhanced due to high throughput, good resource allocation, smooth distribution of workload on the processors, and fault tolerance. If one of its processors fails, then the system can continue and reallocate its tasks to other processors, and thus, the operation can be done smoothly.



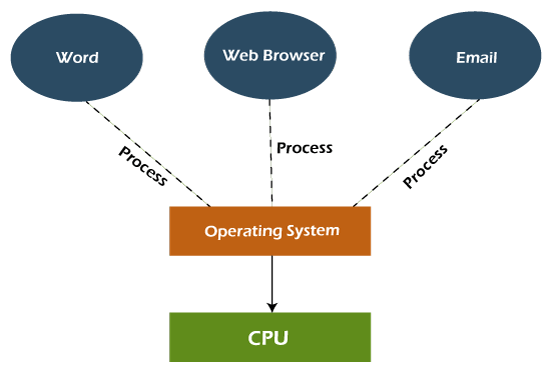
### Advantages of Multiprocessing Operating System:

* Increased Reliability: Because multiple processors are present, if one fails, others may take over it, thus stabilizing the system.
* Increased Throughput: More jobs can be processed at the same time by these multiple processors than by a single processor, thereby increasing the speed of execution.
* Efficient Resource Utilization: It acts more productively upon the utilization of resources like CPU, memory, and I/O devices.
* Parallelism: Many processes can run parallel, which increases the speed of execution to a large extent.
* Scalability: As the workload increases, more processors can be added to enhance performance.

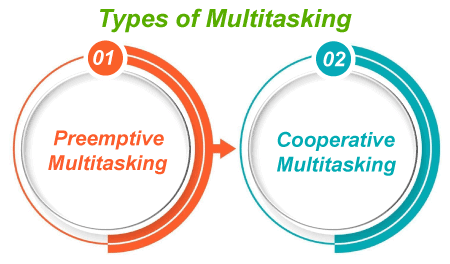
### Disadvantages of Multiprocessing Operating System:

* **Complexity:** There is a rise in the complexity of the system while dealing with a number of processors and proper distribution of tasks.
* **Increased Cost:** Hardware equipment added in multiprocessing systems increases their cost
* **Communication Overhead:** Generally, communication among processors results in overhead and provides a slight reduction in efficiency.
* **Software Compatibility Problems:** Generally, most software is not designed to operate properly with multiprocessing systems.

### Multitasking Operating System



* Multiprogramming extended the multitasking operating system concept, which enabled various programs to run simultaneously.
* It provided a user with the ability to do more than one task on the same system at a given time by sharing various system resources like CPU time among different processes. This assures efficient use of the CPU because it shifts between tasks quickly; hence, it seems that they are running parallel.
* There are multitasking systems in almost all personal computers where, for example, a user might run a web browser, a music player, and a document editor all at once.



**i. Cooperative Operating System:**

* In a co-operating system, the tasks voluntarily relinquish control of the CPU so other tasks can execute. A process runs until it completes or explicitly relinquishes control. An operating system has a relatively small amount of control over when a process will context switch. The operating system must rely on the processes to co-operate with each other. Such a system may lead to problems if a process does not yield control and causes others to be blocked.

**ii. Preemptive Operating System:**

* In this system, the operating system has full control over how it allocates the CPU. In this system, it can interrupt the process in the middle of its execution and switch to another based on previously defined criteria for scheduling such as priority, time slices, or deadlines. This means that within such a system, better multitasking and responsiveness are ensured, and there is no event of allowing any single process to monopolize the CPU.

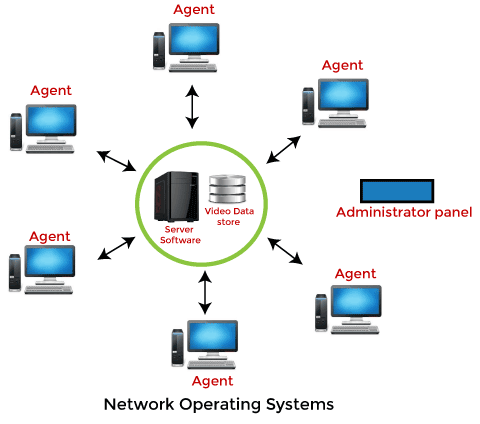
### Advantages of the Multitasking Operating System:

* This system can handle multiple users or tasks at once; therefore, it is best utilized in multi-user environments.
* The memory is allocated dynamically and efficiently to various tasks so that there is optimal usage of system resources.
* More applications can be run at the same time, which increases productivity because tasks are executed concurrently.
* It provides for rapid shifts between tasks, therefore, shortening the response time for the user.
* Because the system keeps running by switching between tasks, CPU time and other system resources are consumed more effectively.

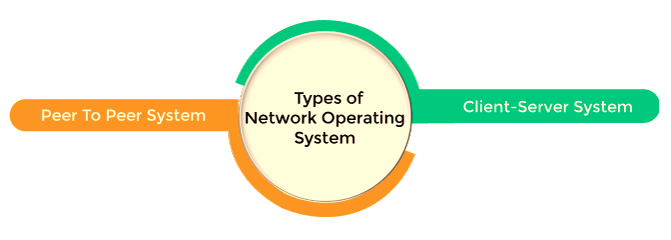
### Disadvantages of Multitasking Operating System:

* The multitasking environment makes a number of processors busier at the same time; that is, the CPU generates more heat.
* The managing of different tasks all together requires a more sophisticated algorithm, which becomes complex to administer.
* Running too many applications at the same time can stress the system to a point where performance degrades.
* In this scenario, multiple tasks will be competing for the same resources, thereby delaying them.
* Multitasking systems usually require more powerful hardware, especially in terms of memory and processing power, to run without a hitch.

## Network Operating System



* A NOS, which is an abbreviation for network operating system, is an operating system that exists to allow computers to communicate and share resources over a network, almost like the backbone of networking, ensuring that different devices (like computers, printers, and even servers) can always connect and interact seamlessly.
* NOS allows devices not only to communicate but also to share software applications, files, and hardware so everyone in the network can have what they need without a complicated setup. Examples of network operating systems are Microsoft Windows Server, Linux, and Novell NetWare.
* The enterprise saves money since resources like printers or storage drives do not have to be duplicated for every user as they are shared, making the system in its whole aspect easier and cheaper to operate. It is the tool that keeps the network running smoothly, making it possible for connected devices to work together in collaboration and productivity.



### Advantages of Network Operating Systems

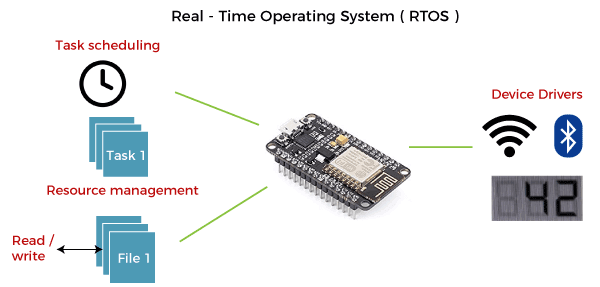
* Since network applications are split between the clients and the servers, this minimizes the total number of communications on the network, thus increasing performance.
* Configuring as well as maintaining an NOS is less expensive than other elaborate systems since shared resources reduce duplication needed.
* A NOS grants centralized control over data, security, and resource management. That means that administrations may easily manage a large network.
* Scaling of the systems can be made easily in terms of organization growth. Easly, new clients or servers can be added without reconfiguring the system.
* It shares resources like printers, files, and applications, which reduces hardware and software redundancy.

### Disadvantage of Network Operating System

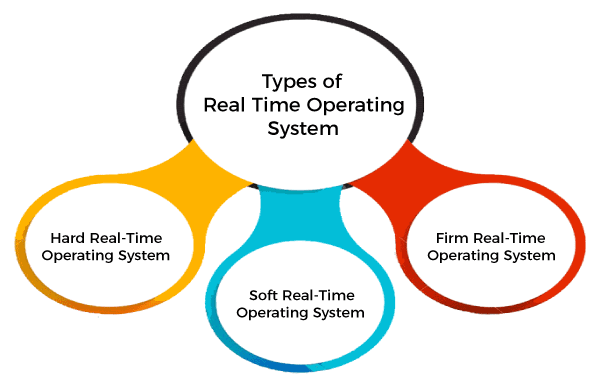
* If one node or server fails, it affects the whole system, and network functions will be interrupted. Hence, reliability is important.
* Security needs to be robust so that unauthorized access is restricted. Complex security measures demand constant monitoring and updating.
* The skilled network administrators involved will handle system performance, security configurations, and troubleshooting.
* When the network size is too large, and traffic is heavy, it degrades with time if it's not monitored and maintained, which is constantly demanding attention.
* This is because an attacker, once able to gain access to one server, comes close to achieving his goal for multiple resources contained in the entire network.

## Real Time Operating System

* A Real-Time Operating System is conceived as a design for tasks that need to be completed under very strict time constraints. In such systems, each task has a clear deadline; failing to achieve completion by that deadline can, in turn, bring about significant consequences. Completion by the deadline may result in either a critical failure or something no longer of relevance, again depending on the application, thus leading to inefficiencies or even dangers in its application.
* Real-time systems are widely used in areas where timing is the issue, for example, the manufacturing of medical equipment, automotive controls, or even industrial automation. For instance, in an anti-lock braking system (ABS) in a car, the response must come in milliseconds in order to assure safety.



* The application of a Real-Time system exists in the case of military applications. If you want to drop a missile, then the missile is supposed to be dropped with a certain precision.



**i. Hard Real-Time Systems**

* In hard real-time systems, it is absolutely impossible to fail to meet a deadline. The consequence of missing a deadline may be disastrous, even failing the system or loss of life.
* Examples: Flight navigation and control aircraft, Medical devices, like pacemakers. It is required that strict guarantees be established so that the jobs will definitely be executed on time. Failure can be very disastrous.

**ii. Soft Real-Time Systems**

* In this system type, occasional missed deadlines may not result in catastrophic failure but can degrade the performance or usability of the system.
* Examples: Video streaming (occasional delays or buffering are acceptable) and online transaction systems (like bank ATMs, where slight delays are tolerable).
* The system may still be functionally operational and workable even if some deadline misses do occur, though the performance would be adversely affected. This has focused on efficiency and the removal of delays rather than absolute precision.

**iii. Firm Real-Time Systems**

* In between hard and soft real-time systems, there are firm real-time systems. In that case, there is no crash, but the result of the task is worthless when it doesn't meet the deadline. There's a minimum kind of penalty, but the value of the completion of the task goes significantly down if it is not made in time.
* Examples: Automated stock trading systems (late transactions lose relevance) and some e-commerce systems (for example, offers run out).

### Advantages of Real-Time Operating System:

* Real-time applications are quite easy to design, develop, and implement under a real-time operating system.
* Maximum utilization of devices and systems in a Real-Time Operating System.
* Fast response to events.
* High reliability in the performance of time-critical operations.
* Strict scheduling ensures the predictable execution of tasks.

### Disadvantages of Real-Time Operating System:

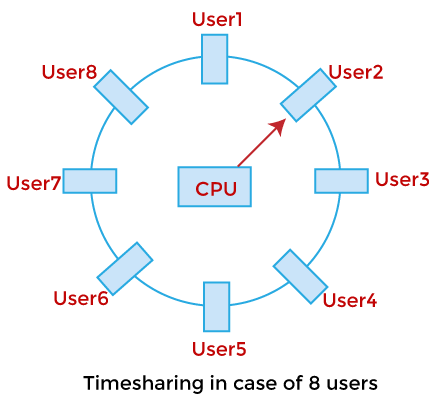
* Real-time operating systems are very expensive to design.
* Real-time operating systems are very resource-intensive and consume critical CPU cycles.
* Less multitasking support.
* Lacks adaptability to new functions.
* At times, it demands specific hardware.

## Time-Sharing Operating System

## A Time-Sharing Operating System allows multiple users to access the system concurrently, and this occurs by allocating a small time slice or quantum to each task. The CPU switches between the tasks so rapidly that the users feel their programs have been running concurrently. This kind of operating system is most effective in systems where multiple users require access to the central system in real-time or near real-time.

## Here's how it works:

* **Scheduling:** Operating systems use scheduling algorithms in order to allow a particular block of CPU time, referred to as a time slice, for each user or process. After each time slice, the CPU shifts to the following process in the queue.
* **Multiprogramming:** The TSOS uses multiprogramming, where several programs are resident in the memory at any given time. This allows several programs to be executed simultaneously.
* **Context Switching:** Whenever the CPU needs to switch to another task, the operating system saves the state of the current process and loads the state of the next process. It is a process referred to as context switching. That lets a task be resumed exactly where it left off.



* A time-sharing operating system allows many users to be served simultaneously, so sophisticated CPU scheduling schemes and Input/output management are required.
* Time-sharing operating systems are very difficult and expensive to build.

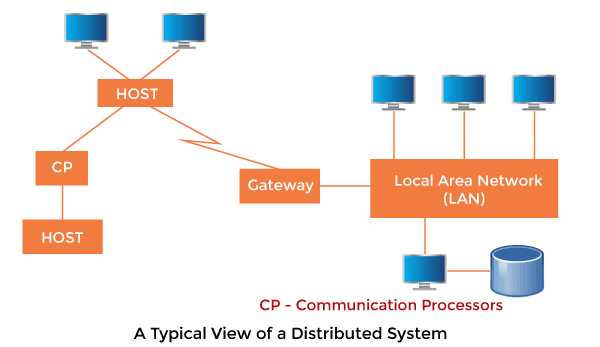
### Advantages of Time-Sharing Operating System

* The time-sharing operating system facilitates effective utilization and sharing of resources.
* This system helps decrease CPU idle and response time.
* It allows various users to access and interact with their programs at the same time, and it leads to greater productivity.
* Time-sharing assures better memory management because it swaps programs into and out of main memory efficiently.
* An interactive computing environment provides users with real-time access to their programs and files.

### Disadvantages of Time-Sharing Operating System

* Data transmission rates compared to others are very high
* The integrity and security of the user programs loaded in memory and data have to be ensured since many users access the system concurrently.
* Implementation as well as management of time-sharing systems is more complicated than others since there is a tendency for task scheduling and memory management.
* Since more and more users get hooked to the system, it degrades due to resource contention.
* However, with time, context switches between tasks will incur overheads, thereby impacting the overall efficiency of the system.

## Distributed Operating System

* The Distributed Operating system is not installed on a single machine; it is divided into parts, and these parts are loaded on different machines. A part of the distributed Operating system is installed on each machine to make their communication possible.
* Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.
* 

### Advantages of Distributed Operating System

* The distributed operating system offers resource sharing.
* This is a fault-tolerant system.
* Scalability is achieved and easily new nodes can be added to the system.
* Distributed task execution improves its performance.
* Parallel processing helps in increasing the speed of job execution and enhances efficiency in getting results.

### Disadvantages of Distributed Operating System

* Protocol overhead might control computation costs.
* The managing of the system is complex as it works in distributed.
* Security may prove difficult on multiple nodes.
* The system highly depends on network stability for it to run without hassles.

1. Interactive operating system

* An interactive operating system is an OS that is designed to facilitate real-time interaction between the user and the system. This means that the user provides inputs, which the system processes instantly to produce immediate outputs.
* The primary feature of such systems is a graphical user interface (GUI), where users can see the changes resulting from their inputs almost instantly.
* Characteristics of Interactive Operating Systems:

1. Real-time interaction: The system responds instantly to user inputs.
2. User-friendly interface: They generally have a GUI that makes them easier for the average user to operate.
3. Multitasking capability: They allow multiple tasks or processes to be performed simultaneously.

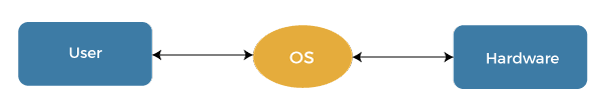
### Examples of Interactive Operating Systems

1. Microsoft Windows: Known for its user-friendly GUI, this OS has been a staple on personal computers for decades.
2. MacOS: Apple's operating system, known for its sleek design and intuitive user interface.
3. Linux (with GUI): While Linux can be a command-line interface (CLI) based OS, many distributions come with a GUI, making it an interactive OS.
4. Android/iOS: These OS used on smartphones and tablets are prime examples of interactive OS due to their highly responsive touch interfaces.

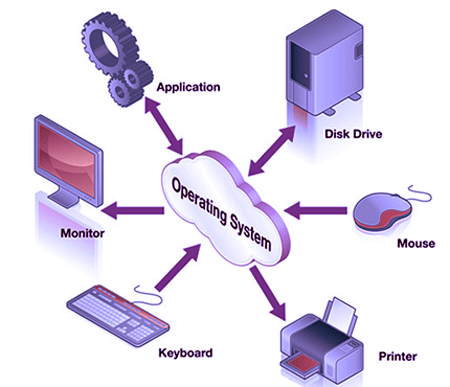
* Importance of Interactive Operating Systems

1. Interactive operating systems have revolutionized the way users interact with their computers and other digital devices.
2. They've made computers more accessible to non-technical users and have facilitated the development of complex, interactive applications.
3. Moreover, the real-time feedback provided by these systems allows users to immediately see the results of their actions, making them essential for tasks that require constant user interaction, such as video editing, game playing, and real-time communication.
4. Operating system services:

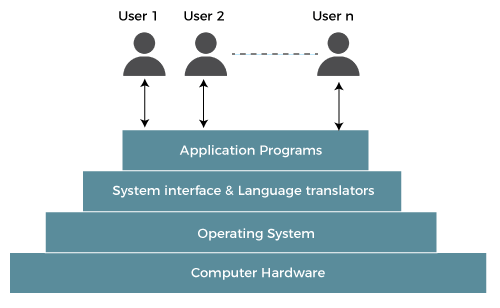
* An Operating System provides an interaction between the users and computer hardware.
* A user is a person sitting at the computer terminal concerned about the application rather than the architecture of the computer. The user never interacts with the hardware directly. To get the services of the hardware, he has to request through the operating system.



* The operating system is a primary resource manager. It manages the hardware, including processors, memory, Input-Output devices, and communication devices.



* The operating system operates either in kernel mode or user mode. Compilers and editors run in user mode, whereas operating system code runs in kernel mode.



* Operating system services:
  1. Program execution
  2. Control Input/output devices
  3. Program creation
  4. Error Detection and Response
  5. Accounting
  6. Security and Protection
  7. File Management
  8. Communication
  9. **Program execution**
* To execute a program, several tasks need to be performed. Both the instructions and data must be loaded into the main memory. In addition, input-output devices and files should be initialized, and other resources must be prepared.
* The Operating structures handle these kinds of tasks. The user now no longer should fear the reminiscence allocation or multitasking or anything.
  1. **Control Input/output devices**
* As there are numerous types of I/O devices within the computer system, and each I/O device calls for its own precise set of instructions for the operation. The Operating System hides that info with the aid of presenting a uniform interface. Thus, it is convenient for programmers to access such devices easily.
  1. **Program Creation**
* The Operating system offers the structures and tools, including editors and debuggers, to help the programmer create, modify, and debugging programs.
  1. **Error Detection and Response**
* An Error in a device may also cause malfunctioning of the entire device. These include hardware and software errors such as device failure, memory error, division by zero, attempts to access forbidden memory locations, etc. To avoid error, the operating system monitors the system for detecting errors and takes suitable action with at least impact on running applications.
* While working with computers, errors may occur quite often. Errors may occur in the:
* Input/ Output devices: For example, connection failure in the network, lack of paper in the printer, etc.
* User program: For example: attempt to access illegal memory locations, divide by zero, use too much CPU time, etc.
* Memory hardware: For example, Memory error, the memory becomes full, etc.
* To handle these errors and other types of possible errors, the operating system takes appropriate action and generates messages to ensure correct and consistent computing.
  1. **Accounting**
* An Operating device collects utilization records for numerous assets and tracks the overall performance parameters and responsive time to enhance overall performance. These personal records are beneficial for additional upgrades and tuning the device to enhance overall performance.
  1. **Security and Protection**
* Operating device affords safety to the statistics and packages of a person and protects any interference from unauthorized users. The safety feature counters threats, which are published via way of individuals out of doors the manage of the running device.
* For Example:

When a user downloads something from the internet, that program may contain malicious code that may harm the already existing programs. The operating system ensures that proper checks are applied while downloading such programs.

* If one computer system is shared amongst a couple of users, then the various processes must be protected from another intrusion. For this, the operating system provides various mechanisms that allow only those processes to use resources that have gained proper authorization from the operating system. The mechanism may include providing unique users ids and passwords to each user.
  1. **File management**
* Computers keep data and information on secondary storage devices like magnetic tape, magnetic disk, optical disk, etc. Each storage media has its capabilities like speed, capacity, data transfer rate, and data access methods.
* For file management, the operating system must know the types of different files and the characteristics of different storage devices. It has to offer the proportion and safety mechanism of documents additionally.
  1. **Communication**
* The operating system manages the exchange of data and programs among different computers connected over a network. This communication is accomplished using message passing and shared memory.

1. System call:

* A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.
* A system call can be written in assembly language or a high-level language like **C** or **Pascal**. System calls are predefined functions that the operating system may directly invoke if a high-level language is used.
* A system call is a method for a computer program to request a service from the kernel of the [operating system](https://www.tpointtech.com/os-tutorial) on which it is running.
* A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.
* The **Application Program Interface (API)** connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services.
* The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.
* **User vs Kernel Mode:** In user mode, programs can't directly access hardware resources; they request through system calls the operations in kernel mode. System calls to preserve the security of the system through its accessibility resources, which an OS can get. This is because system calls mainly limit the direct hardware control to the OS.

## How are system calls made?

* When a computer software needs to access the operating system's kernel, it makes a system call. The system call uses an API to expose the operating system's services to user programs. It is the only method to access the kernel system. All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.
* Below are some examples of how a system call varies from a user function.

1. A system call function may create and use kernel processes to execute the asynchronous processing.
2. A system call has greater authority than a standard subroutine. A system call with kernel-mode privilege executes in the kernel protection domain.
3. System calls are not permitted to use shared libraries or any symbols that are not present in the kernel protection domain.
4. The code and data for system calls are stored in global kernel memory.
5. System calls control IPC and system resource allocation; user functions usually do not.
6. The system calls control signals, interrupts, and memory protection; user functions cannot modify these directly.
7. System calls execute with higher privileges and tighter security conditions; user functions are of lesser privilege.
8. System calls may directly use kernel-space resources; no direct effect on kernel operations is available for user functions.

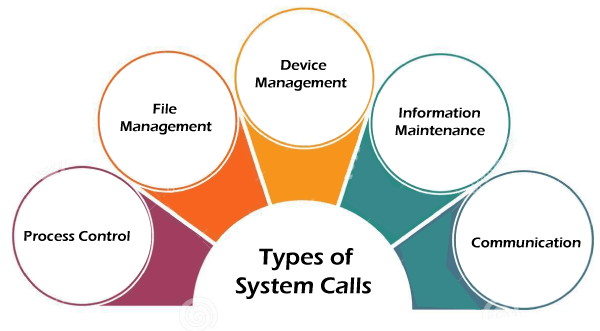
## Why do you need system calls in Operating System?

1. It is must require when a file system wants to create or delete a file.
2. Network connections require the system calls to sending and receiving data packets.
3. If you want to read or write a file, you need to system calls.
4. If you want to access hardware devices, including a printer, scanner, you need a system call.
5. System calls are used to create and manage new processes.
6. They allow system programs to exercise control over hardware devices-for example, to set parameters or to read status.
7. They enforce access controls and permissions to protect system resources.
8. Used for inter-process communication and coordination.
9. System calls provide mechanisms for accessing status information and configuration data about the system.

## How System Calls Work?

* The Applications run in an area of memory known as user space. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel. It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.
* If the request is permitted, the kernel performs the requested action, like creating or deleting a file. As input, the application receives the kernel's output. The application resumes the procedure after the input is received. When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.
* A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

## Types of System Calls



1. **Process Control**
2. **File Management**
3. **Device Management**
4. **Information Maintenance**
5. **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.
* Here's a closer look at how it all works out:
* **Process Creation:** Every time you start a new application, an operating system calls into life and assigns it a process. It then assigns it a unique process ID and gives it all the required resources, such as memory, that will enable it to begin its start-up.
* **Scheduling of Processes:** The operating system needs to decide which process gets its turn with the CPU. It achieves this with scheduling algorithms—kind of like making sure every person in line gets their turn at the checkout. This keeps things moving without waiting, whether it is a first-come-first-served approach or prioritizing specific tasks.
* **Process Execution:** After a process is born, it's time to run the program. OS swaps between processes, ensuring that your computer is multitasking. Imagine jugglers tossing balls: every time one goes to the floor, another snatched off the return in mid-air. That is how the CPU manages all sorts of processes!
* **Suspension and Resumption:** At other times, processes are suspended to free up resources or let other more important processes run. The system saves its state so it can pick up right where it left off when resumed, just like hitting the pause button in a video game.
* **Process Termination:** Every process eventually needs to terminate-it either finishes its work or an error occurs. In the case of an OS, it simply deallocates that resource from memory and gets rid of it, making everything neat and clean.
* **Inter-process Communication (IPC):** Processes sometimes need to communicate with other processes. To achieve this, the OS has facilities like message passing or shared memory. Think of two friends texting each other; that's the way processes communicate.
* **Synchronization:** Sometimes, processes desire to share resources - files or memory, for example. However, if they all wish to access a resource simultaneously, things may become somewhat unravelled. So, in order to prevent this kind of chaos, the OS uses tools such as semaphores and mutexes, which are in charge of orchestrating everything smoothly so that no such conflict might arise.
* **Deadlock Handling:** Deadlock is where the processes are deadlocked, waiting for each other's resources, people holding doors open for one another, and neither can get through. The OS has strategies to avoid or resolve these situations, ensuring that no process is deadlocked for an indefinite amount of time.

### File Management

* So, **File Management** basically refers to how your computer handles the organization and manipulation of files. It is rather like storing your physical documents in a drawer: each file has to be created, opened, edited, and perhaps sometimes even trashed. The OS takes care of everything behind the scenes.
* **Creating Files:** Each time you save a new document or program, your computer creates a file. It gives it a location on the hard drive or other storage - much the way you might put a paper file into a folder.
* **Opening Files:** To look at or edit a file, you open it. The operating system will retrieve it so that you can view it or work on it.
* **Reading Files:** Once the file is opened, one can read what is inside. Be it a text file, an image, or anything else, the OS will call from storage to be viewed and worked with.
* **Writing to Files:** Every time you make a change—either by typing into a document or touching up a picture—the operating system writes those changes to the file.
* **Closing Files:** When you are finished with it, closing the file saves everything and marks it as not in use. It is like putting the document back in its folder when you are finished reading or modifying it.
* **File Deletion:** You just delete files whenever you think they should not be in your system again. The operating system would then delete them from your storage, creating more space for other things.
* **Controlling Permissions:** You might sometimes need to control who can access and edit some of the files. The OS allows you to set permissions for each file so that you may choose whether someone else can run, alter, or open it.

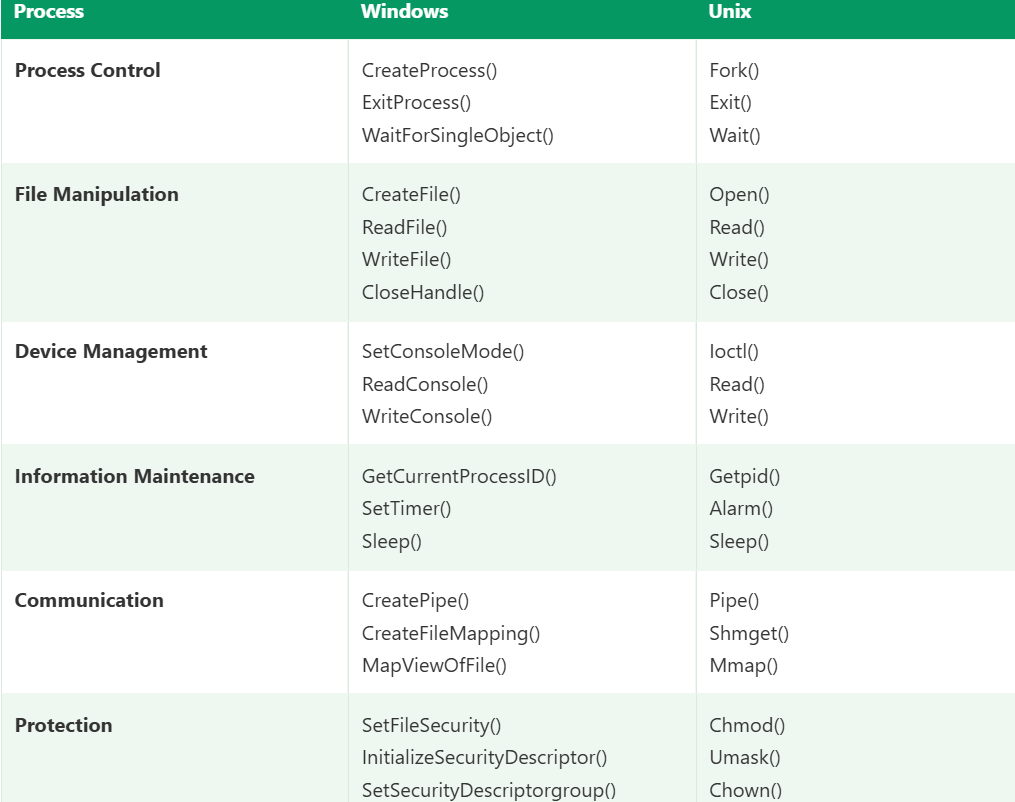
### Device Management

* 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.
* What device management does;
* **Reading from Devices:** Whenever the computer wants to read data from the device, be it reading a file off of your hard drive or input from the keyboard, the operating system handles that request. It makes sure the data is read right and then forwarded to the right place.
* **Writing to Devices:** whenever you need to export data to a device, such as saving the file of your report to the USB stick or just printing it, the OS takes care of it and makes sure that all the data gets written to the device in the right way.
* **Getting Device Attributes:** The OS can get such information about a device, say, its status or settings. For example, it can check what is available on disk or the level of the battery of a wireless mouse.
* **Allocation of Resources:** OS decides how many of the system resources like memory or processing power a device gets, so no device gets more than its share, and the whole thing runs without problems.
* **Release Devices:** The OS releases a device, that is, frees system resources allocated for that device once it has completed all its work. Essentially, it is as safe as pulling a USB drive out of the computer after you are done with it.
* **Handling Interrupts:** Devices often send signals to the CPU, termed as interrupts, claiming its attention. The OS handles these interruptions so that it will come to know whenever a device needs attention and deal with them in the right sequence.

### 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..
* The usual information maintenance in everyday life activities are;
* **Get System Data:** It fetches information about the system that it actually is, such as CPU or memory usage, network status, etc. It keeps records of all the resources in use.
* **Set System Data:** The OS can, whenever needed, change some of the configurations associated with the systems, such as changing system behaviour, updating configurations, setting user preferences, etc.
* **Retrievability of Time or Date:** OS can retrieve the time and date of the system, which is used in many applications such as file timestamp, logging and scheduling.
* **Setting Time or Date:** The system may update the current time and date when necessary; thus, accurate times and dates can be appended to files or processes or synchronized with an external time clock.
* **System logs and audits:** The operating system maintains an event log and activity record of errors, system performance, or user actions. It is primarily used for debugging and monitoring the system.

### 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.
* Key activities involved in communication are as follows:
* **Setting Up Communication Links:** OS connects two processes or devices through a communication channel. The connection may be performed using sockets, pipes, or shared memory; thus, it facilitates easy communication between these two processes or devices.
* **Closing Communication Links:** OS dismantles or deletes a link if its communication process is complete; resources are released, and the system is not affected.
* The OS ensures that a message or data sent by one process or system reaches the target recipient, either locally or on a network.
* **Receiving Messages:** The receiving process gets the data sent by another process. The OS manages the incoming messages and makes sure that it is delivered to the correct target.
* **Manages the protocols of communication:** The OS manages rules as well as communications protocols, which enable data to be both sent and received in appropriate formats and sequences.
* 

### open()

* An open() call allows a process to start accessing the file stored in the file system. As such, it can allocate resources on demand and return a file descriptor, which becomes a handle used to access or read from the file.
* A number of processes could open the same file, or access to a file might be exclusive to only one process again, depending upon the design of the file system and permissions. A file open() system call needs to be performed before any reads or writes are performed on a file.

### read()

* It is used to obtain data from a file on the file system.
* It accepts three arguments in general:
* A file descriptor.
* A buffer to store read data.
* The number of bytes to read from the file.
* The file descriptor of the file to be read could be used to identify it and open it using **open()** before reading.

### wait()

* In some systems, a process may have to wait for another process to complete its execution before proceeding. When a parent process makes a child process, the parent process execution is suspended until the child process is finished.
* The **wait()** system call is used to suspend the parent process. Once the child process has completed its execution, control is returned to the parent process.

### write()

* It is used to write data from a user buffer to a device like a file. This system call is one way for a program to generate data.
* It takes three arguments in general:
* A file descriptor.
* A pointer to the buffer in which data is saved.
* The number of bytes to be written from the buffer.

### fork()

* Processes generate clones of themselves using the **fork()** system call. It is one of the most common ways to create processes in operating systems. When a parent process spawns a child process, execution of the parent process is interrupted until the child process completes. Once the child process has completed its execution, control is returned to the parent process.

### close()

* It is used to end file system access. When this system call is invoked, it signifies that the program no longer requires the file, and the buffers are flushed, the file information is altered, and the file resources are de-allocated as a result.

### exec()

* When an executable file replaces an earlier executable file in an already executing process, this system function is invoked. As a new process is not built, the old process identification stays, but the new process replaces data, stack, data, head, etc.

### exit()

* The **exit()** is a system call that is used to end program execution. This call indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the **exit()** system function.

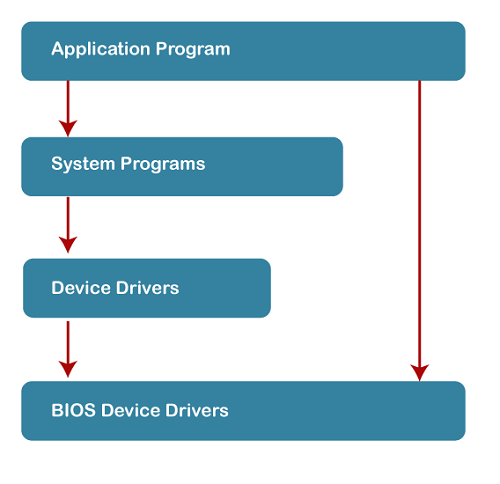
1. Structure of operating system:

* Structures are-

1. Simple Structure
2. Monolithic Structure
3. Layered Approach Structure
4. Micro-Kernel Structure
5. Exo-Kernel Structure
6. Virtual Machines

## Simple Structure

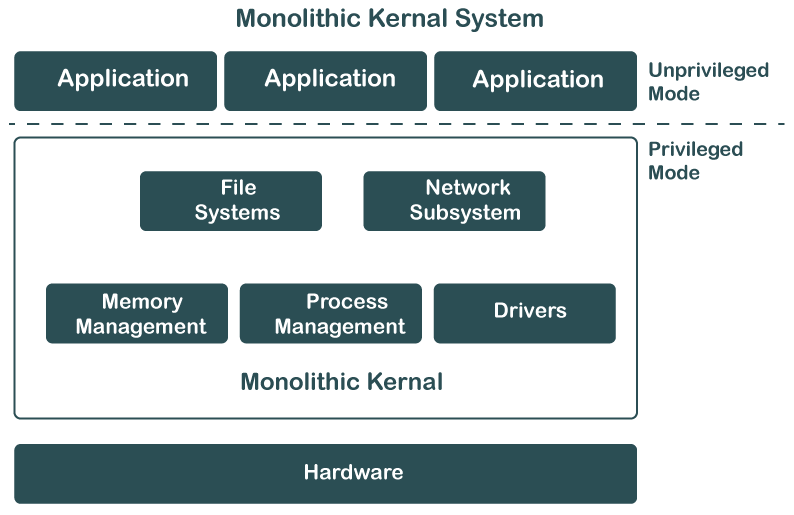
* It is the most straightforward operating system structure, but it lacks definition and is only appropriate for usage with tiny and restricted systems.
* Since the interfaces and degrees of functionality in this structure are clearly defined, programs are able to access I/O routines, which may result in unauthorized access to I/O procedures.
* **This organizational structure is used by the MS-DOS operating system:**
* There are four layers that make up the MS-DOS operating system, and each has its own set of features.
* These layers include ROM BIOS device drivers, MS-DOS device drivers, application programs, and system programs.
* The MS-DOS operating system benefits from layering because each level can be defined independently and, when necessary, can interact with one another.
* If the system is built in layers, it will be simpler to design, manage, and update. Because of this, simple structures can be used to build constrained systems that are less complex.
* When a user program fails, the operating system as whole crashes.
* Because MS-DOS systems have a low level of abstraction, programs and I/O procedures are visible to end users, giving them the potential for unwanted access.



* **Advantages of Simple Structure:**
* It is easy to develop as it contains only a few interfaces and levels.
* It executes faster because there are fewer layers between the hardware and applications.
* Direct access to hardware provides faster operations.
* Minimal consumption of system resources as the structure is very simple.
* Light in weight hence suitable for small or even embedded systems.
* **Disadvantages of Simple Structure:**
* As these components are not distinctly separated, making any change or upgrade in the system is considerably harder to achieve.
* All things are connected, and therefore, debugging is quite hectic; finding the error is very hard.
* Due to the lack of data abstraction, a security vulnerability is a strong possibility.
* It does not scale well to larger, more complex systems. Adding new features is particularly challenging in this architecture.
* One failure in one part might bring down the whole system.

### MONOLITHIC STRUCTURE

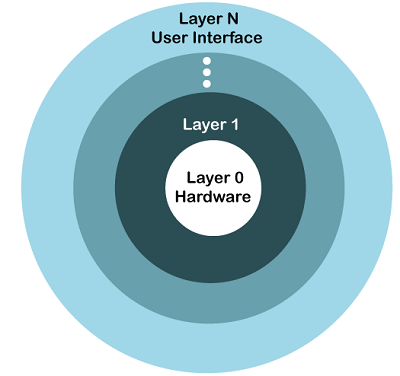
* The monolithic operating system controls all aspects of the operating system's operation, including file management, memory management, device management, and operational operations.
* The core of an operating system for computers is called the kernel (OS). All other System components are provided with fundamental services by the kernel.
* The operating system and the hardware use it as their main interface. When an operating system is built into a single piece of hardware, such as a keyboard or mouse, the kernel can directly access all of its resources.
* The monolithic operating system is often referred to as the monolithic kernel. Multiple programming techniques such as batch processing and time-sharing increase a processor's usability. Working on top of the operating system and under complete command of all hardware, the monolithic kernel performs the role of a virtual computer.
* This is an old operating system that was used in banks to carry out simple tasks like batch processing and time-sharing, which allows numerous users at different terminals to access the Operating System.



* **Advantages of Monolithic Structure:**
* Because everything is packed into one big kernel, there is no need for multi-layer complexities. Hence, the design and bringing into practice of such a system is relatively easy.
* As long as all the components run in the same memory space, there is no need to have inter-process communication or to perform context switching between layers; thus, execution is faster.
* Direct access to hardware and little additional-abstraction overhead result in high performance.
* Development can thus be easier as developers have to deal with one codebase only, and no interactions among various layers or modules need to be considered.
* **Disadvantages of Monolithic Structure:**
* Fault in any single component has the potential to bring down the whole system since there is no isolation between different parts.
* This can be a hard and unsafe task, as changes in one part of the system may accidentally filter out into another part.
* Since the components are more interdependent, debugging is also more complex and always requires heavy testing and testing-related activities.
* This is an issue where scaling up or adjusting to the latest trends in a system involves significant changes in the kernel.
* Consequently, it will be cumbersome to update or upgrade the system because it involves a large codebase in a monolithic form where changes may have widespread effects.

### LAYERED STRUCTURE

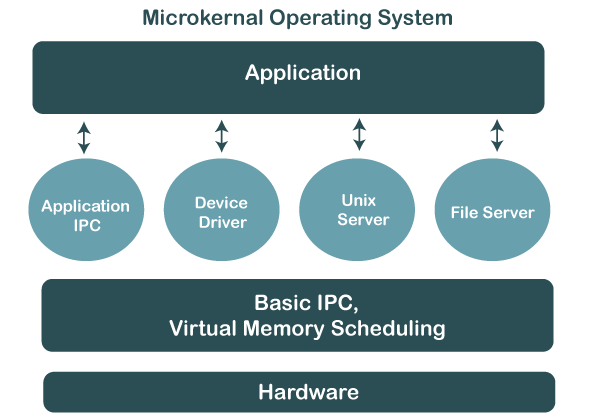
* The OS is separated into layers or levels in this kind of arrangement. Layer 0 (the lowest layer) contains the hardware, and layer 1 (the highest layer) contains the user interface (layer N).
* The model, in this respect, uses the functions' availability through the layer below it. For example, memory management may have to rely on process scheduling, while higher layers use it here with respect to user interaction.
* This approach also introduces abstraction between layers, whereby the system is also easier to manage and debug. In case any issue arises, testing and debugging can be performed at lower layers before going onto higher layers
* The functionalities of each layer are separated in this method, and abstraction is also an option. Because layered structures are hierarchical, debugging is simpler, therefore all lower-level layers are debugged before the upper layer is examined. As a result, the present layer alone has to be reviewed since all the lower layers have already been examined.



* **Advantages of Layered Structure:**
* Development, maintenance, and understanding are much easier, as every layer is assigned a specific role.
* It can isolate problems in individual layers because testing and debugging would be less demanding.
* Sensitive operations can be confined to specific layers; thus, unauthorized access can be minimized to such layers because the concerns have been clearly separated.
* Modifying or updating one layer will not affect other independent layers directly; thus, upgrading or fixing could be done more easily.
* With layers, basic operations can be entrusted to lower layers, and hence, the layers above need not be concerned with the minute details, thereby reducing the complexity of the overall management.
* **Disadvantages of Layered Structure:**
* The layered approach adds delays due to the fact that data and commands must pass through several layers, thus slowing down the system in general.
* Designing a properly working layered structure requires forethought and iron discipline for proper interaction between different layers.
* Layers are generally dependent on lower layers; thus skipping layers and directly accessing the lower-level functionalities becomes harder, hence reducing flexibility in many systems.
* System performance optimization becomes challenging due to the extra abstractions introduced by the layers, which harden the tuning of individual parts of the system to make them run more effectively.
* Unless the lower-layer design is lousy or inefficient, the whole system will suffer since the higher layer depends on them for functionality.

### MICRO-KERNEL STRUCTURE

* The operating system is created using a micro-kernel framework that strips the kernel of any unnecessary parts. Systems and user applications are used to implement these optional kernel components. So, Micro-Kernels is the name given to these systems that have been developed.
* The **Micro-Kernel Structure** minimizes the functions handled by the kernel, delegating most services to user-level applications. The micro-kernel itself only handles core functions, such as memory management, process scheduling, and basic communication between processes.
* By stripping away unnecessary services from the kernel, the system becomes more secure and reliable. If a service crashes, it doesn’t affect the entire system.
* Micro-kernels are common in modern OS architectures like macOS and QNX, as they allow for easier updates and better system resilience.



* **Advantages of Micro-Kernel Structure:**
* The just necessary services running in the kernel ensure that user service failures don't crash the whole system.
* It isolates the services from the kernel, thus minimizing the possibility of compromising the system since every service operates independently of the other.
* Smaller kernels are easier to operate and alter. Hence, updating and bug fixes are less challenging.
* Services can be added or removed without affecting the kernel. So, this gives more flexibility while designing the system.
* Due to the small size of the micro-kernel, it can be ported on different hardware platforms with minimal modification.
* **Disadvantages of Micro-Kernel Structure:**
* The inter-process communication between kernel and user services introduces additional overhead and hence is slower as compared to the monolithic approach.
* In a micro-kernel, the separation of services must be done very carefully, which may complicate the initial development process.
* Context switches are more frequent between user space and kernel space in this approach; hence, performance would be slower in highly loaded conditions.
* Due to the complexity and possibly extra performance trade-offs, few operating systems have been built using the micro-kernel architecture.
* Modular requires more planning and testing in detail; hence, the development cycle may be longer.

### EXOKERNEL

* An operating system called Exokernel was created at MIT with the goal of offering application-level management of hardware resources. The exokernel architecture's goal is to enable application-specific customization by separating resource management from protection. Exokernel size tends to be minimal due to its limited operability.
* Because the OS sits between the programs and the actual hardware, it will always have an effect on the functionality, performance, and breadth of the apps that are developed on it.
* By rejecting the idea that an operating system must offer abstractions upon which to base applications, the exokernel operating system makes an effort to solve this issue. The goal is to give developers as few restriction on the use of abstractions as possible while yet allowing them the freedom to do so when necessary. Because of the way the exokernel architecture is designed, a single tiny kernel is responsible for moving all hardware abstractions into unreliable libraries known as library operating systems. Exokernels differ from micro- and monolithic kernels in that their primary objective is to prevent forced abstraction.
* **Exokernel operating systems have a number of features, including:**
* Enhanced application control support.
* Splits management and security apart.
* A secure transfer of abstractions is made to an unreliable library operating system.
* Brings up a low-level interface.
* Operating systems for libraries provide compatibility and portability.
* **Advantages of Exokernel Structure:**
* Applications directly interface with hardware to realize high power.
* It provides fine-grained resource allocation and revocation for efficient usage.
* Testing and development of new operating systems with low-level control become easy.
* Each application can have its memory management; thus, optimization would be tailored.
* It removes extra kernel features; thus, it makes the system lean and faster.
* Because of its minimalistic approach, it is easily adaptable to different hardware platforms.
* **Disadvantages of Exokernel Structure:**
* Developing and maintaining such a complicated interface would be challenging.
* The abstractions are not enforced, which might make the different applications inconsistent with each other.
* It makes the developers responsible for dealing with resource management, which enhances the difficulty level of it.
* Lack of strong abstractions with direct access to hardware may result in security vulnerabilities.
* Debugging is more difficult since most of the system is on a low level of abstraction.
* This means that older applications, which rely on lower-level abstractions, will not function well since the latter abstract higher-level operating system abstractions.

### VIRTUAL MACHINES (VMs)

* The hardware of our personal computer, including the CPU, disc drives, RAM, and NIC (Network Interface Card), is abstracted by a virtual machine into a variety of various execution contexts based on our needs, giving us the impression that each execution environment is a separate computer. A virtual box is an example of it.
* Using CPU scheduling and virtual memory techniques, an operating system allows us to execute multiple processes simultaneously while giving the impression that each one is using a separate processor and virtual memory. System calls and a file system are examples of extra functionalities that a process can have that the hardware is unable to give. Instead of offering these extra features, the virtual machine method just offers an interface that is similar to that of the most fundamental hardware. A virtual duplicate of the computer system underneath is made available to each process.
* We can develop a virtual machine for a variety of reasons, all of which are fundamentally connected to the capacity to share the same underlying hardware while concurrently supporting various execution environments, i.e., various operating systems.
* Disk systems are the fundamental problem with the virtual machine technique. If the actual machine only has three-disc drives but needs to host seven virtual machines, let's imagine that. It is obvious that it is impossible to assign a disc drive to every virtual machine because the program that creates virtual machines would require a sizable amount of disc space in order to offer virtual memory and spooling. The provision of virtual discs is the solution.
* The result is that users get their own virtual machines. They can then use any of the operating systems or software programs that are installed on the machine below. Virtual machine software is concerned with programming numerous virtual machines simultaneously into a physical machine; it is not required to take into account any user-support software. With this configuration, it may be possible to break the challenge of building an interactive system for several users into two manageable chunks.
* **Advantages of Virtual Machines:**
* Each VM is in itself independent, which means any failure or security consequences that take place will not affect the other virtual machines.
* One can have a number of operating systems running on one hardware platform.
* Cloning, backups, and restoration of virtual machines are pretty easy; hence, options galore as far as data recovery is concerned.
* Different operating systems, such as Windows, Linux, etc., can be used with one machine itself.
* They are ideal for testing software updates or any OS in an isolated environment without incurring changes anywhere in the host system.
* They can be easily moved between different physical machines with very little effort.
* **Disadvantages of Virtual Machines:**
* Running more than one virtual machine on the same host may decrease its performance due to the sharing of the resources.
* VMs are CPU, memory-intensive, and storage-intensive applications; this will put a high load on host systems.
* Managing many virtual machines requires further advanced tools and knowledge.
* It cannot utilize the hardware as effectively, as would a physical machine, and hence it affects performance-related applications.
* Uncontrolled creation can lead to disorganization, inefficient use of resources, and security risks.
* Vulnerabilities in either the hypervisor or VM management software may have the potential to expose all VMs to security threats.