Unit - 1

What is OS

Operating System lies in the category of system software. It basically manages all the resources of the computer. An operating system acts as an interface between the software and different parts of the computer or the computer hardware. The operating system is designed in such a way that it can manage the overall resources and operations of the computer.

Functions of the Operating System - **Read in Detail - Click Here**

- Resource Management
- Process Management
- Memory Management
- Security
- File Management
- Device Management
- Networking
- User Interface
- Backup and Recovery
- Virtualization
- Performance Monitoring
- Time-Sharing
- System Calls

Types of Operating Systems

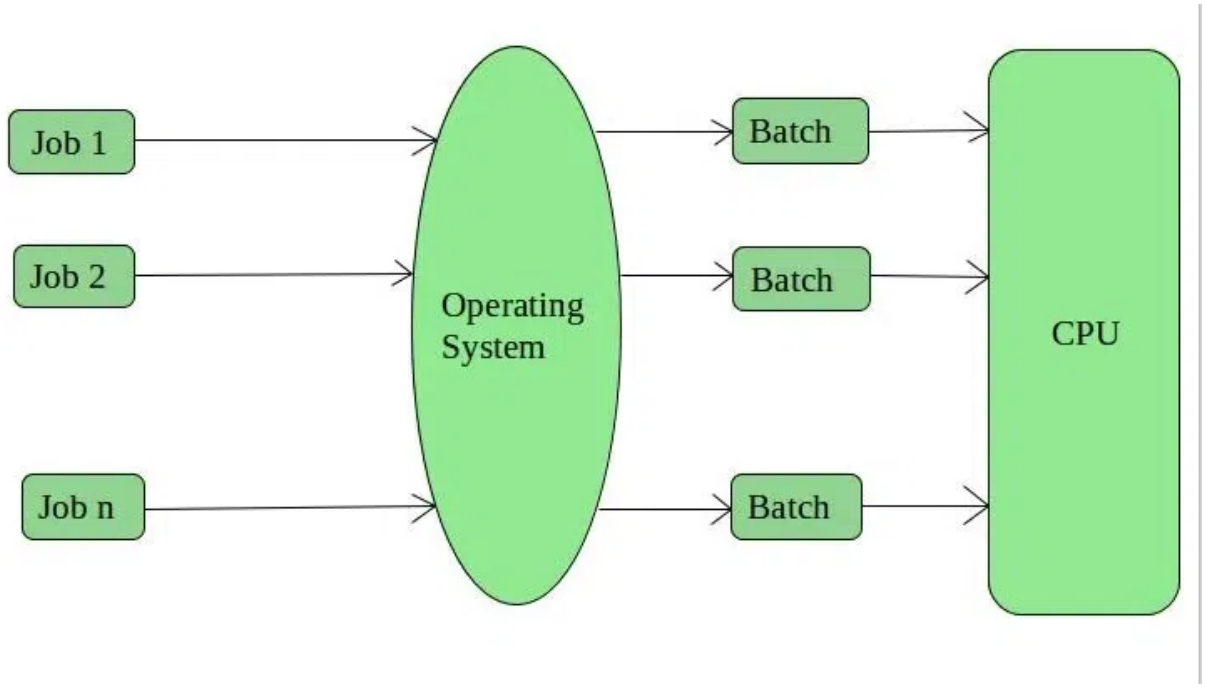
- **Batch Operating System:** A **Batch Operating System** processes jobs without direct user interaction. An operator groups similar jobs with the same requirements into batches for processing.
- **Time-sharing Operating System:** A **Time-sharing Operating System** allows multiple users to share computer resources simultaneously, maximizing resource utilization.
- **Distributed Operating System:** A **Distributed Operating System** manages a network of computers as if they were a single system. It enables multiple users to access shared resources and communicate over the network. Examples include Microsoft Windows Server and various Linux distributions designed for servers.
- **Network Operating System:** A **Network Operating System** runs on a server to manage data, users, groups, security, applications, and other networking functions.
- **Real-time Operating System:** A **Real-time Operating System** responds to inputs with minimal delay. It's designed for applications requiring quick and predictable responses, such as embedded systems, industrial control systems, and robotics.
- **Multiprocessing Operating System:** **Multiprocessor Operating Systems** enhance performance by utilizing multiple CPUs within a single computer system. They divide and execute jobs across linked CPUs for faster processing.
- **Single-User Operating Systems:** **Single-User Operating Systems** support one user at a time. Examples include Microsoft Windows for personal computers and Apple macOS.
- **Multi-User Operating Systems:** **Multi-User Operating Systems** support multiple users simultaneously. Examples include Linux and Unix.
- **Embedded Operating Systems:** **Embedded Operating Systems** run on devices with limited resources, such as smartphones, wearable devices, and household appliances. Examples include Google's Android and Apple's iOS.
- **Cluster Operating Systems:** Cluster Operating Systems manage a group of computers working together as a single system. They're used for high-

performance computing and applications requiring high availability and reliability. Examples include Rocks Cluster Distribution and OpenMPI.

Explanation of Types

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 requirements and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs. Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.



Advantages of Batch Operating System

- 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

- 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.
- In batch operating system the processing time for jobs is commonly difficult to accurately predict while they are in the queue.
- It is difficult to accurately predict the exact time required for a job to complete while it is in the queue.

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

2. Interactive Operating System

Interactive operating systems are computers that accept human inputs. Users give commands or some data to the computers by typing or by gestures. Some examples of interactive systems include MS Word and Spreadsheets, etc. They facilitate interactive behavior. `Mac and Windows OS` are some examples of interactive operating systems.

What is an Interactive Operating System?

- An interactive operative system is an operating system that allows the execution of interactive programs. All PC operating systems are interactive operating systems only.
- An interactive operating system gives permission to the user to interact directly with the computer. In an Interactive operating system, the user enters some command into the system and the work of the system is to execute it.
- Programs that allow users to enter some data or commands are known as Interactive Operating Systems. Some commonly used examples of Interactive operating systems include Word Processors and Spreadsheet Applications.
- A non-interactive program can be defined as one that once started continues without the need for human interaction. A compiler can be an example of a non-interactive program.

Properties of Interactive Operating System:

1. **Batch Processing:** It is defined as the process of gathering programs and data together in a batch before performing them. The job of the operating system is to define the jobs as a single unit by using some already defined sequence of commands or data, etc.Before they are performed or carried out, these are stored in the memory of the system and their processing depends on a FIFO basis. The operating system releases the memory and then copies the output into an output spool for later printing when the job is finished. Its use is that it basically improves the system performance because a new job begins only when the old one is completed without any interference from the user. One disadvantage is that there is a small chance that the jobs will enter an infinite loop. Debugging is also somewhat difficult with batch processing.
2. **Multitasking:** The CPU can execute many tasks simultaneously by switching between them. This is known as Time- Sharing System and also it has a very fast response time. They switch so Fastly that the users can very easily interact with each running program.
3. **Multiprogramming:** Multiprogramming happens when the memory of the system stores way too many processes. The job of the operating system here is that runs these processes in parallel on the same processor. Multiple processes share the CPU, thus increasing CPU utilization. Now, the CPU only performs one job at a particular time while the rest wait for the processor to be assigned to them. The operating system takes care of the fact that the CPU is never idle by using its memory management programs so that it can monitor the state of all system resources and active programs. One advantage of this is that it gives the user the feeling that the CPU is working on multiple programs simultaneously.
4. **Distributive Environment:** A distributive environment consists of many independent processors. The job of the operating system here is that distribute computation logic among the physical processors and also at the same time manage communication between them. Each processor has its own local memory, so they do not share a memory.
5. **Interactivity:** Interactivity is defined as the power of a user to interact with the system. The main job of the operating system here is that it basically provides

an interface for interacting with the system, manages I/O devices, and also ensures a fast response time.

6. **Real-Time System:** Dedicated embedded systems are real-time systems. The main job of the operating system here is that reads and react to sensor data and then provide a response in a fixed time period, therefore, ensuring good performance.
7. **Spooling:** Spooling is defined as the process of pushing the data from different I/O jobs into a buffer or somewhere in the memory so that any device can access the data when it is ready. The operating system here handles the I/O device data spooling because the devices have different data access rates in order to maintain the spooling buffer. Now the job of the buffer here is that acts like a waiting station for the data to rest while the devices which are slower can catch up.

Example of an Interactive Operating System:

Some examples of Interactive operating systems are as follows:

1. Unix Operating System
2. Disk Operating System

Advantages of Interactive Operating System:

The advantages of an Interactive Operating System are as follows:

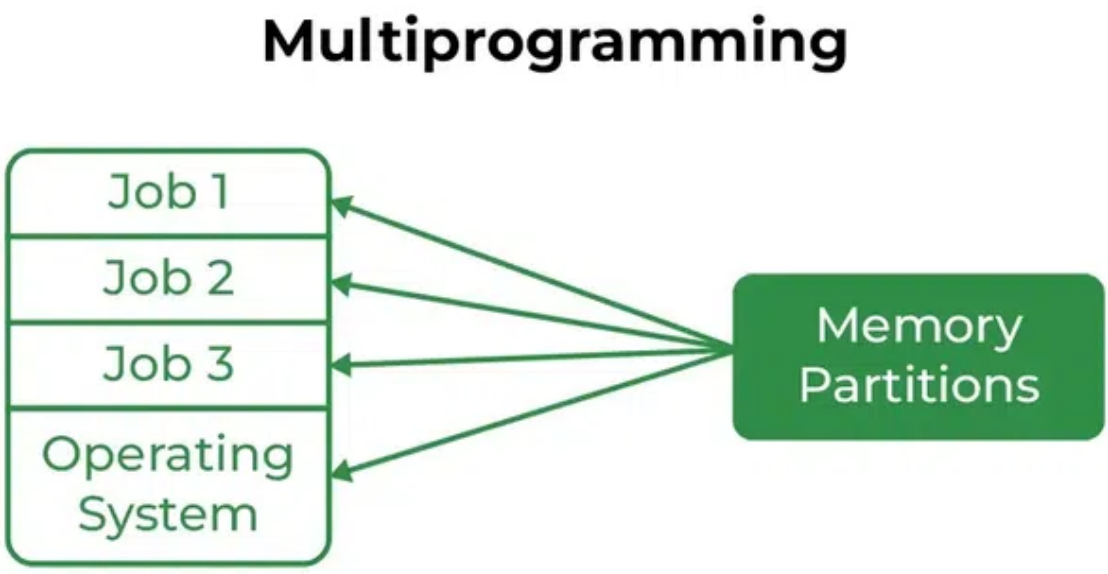
1. **Usability:** An operating system is designed to perform something and the interactiveness allows the user to manage the tasks more or less in real-time.
2. **Security:** Simple security policy enhancement. In non-interactive systems, the user virtually always knows what their programs will do during their lifetime, thus allowing us to forecast and correct the bugs.

Disadvantages of Interactive Operating System:

1. **Tough to design:** Depending on the target device, interactivity might be proved challenging to design because the user must be prepared for every input. What about having many inputs? The state of a program can alternate at any particular time, all the programs should be handled in some way, and also it doesn't always work out properly.

3. 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 utilization of resources.



Advantages of Multi-Programming Operating System

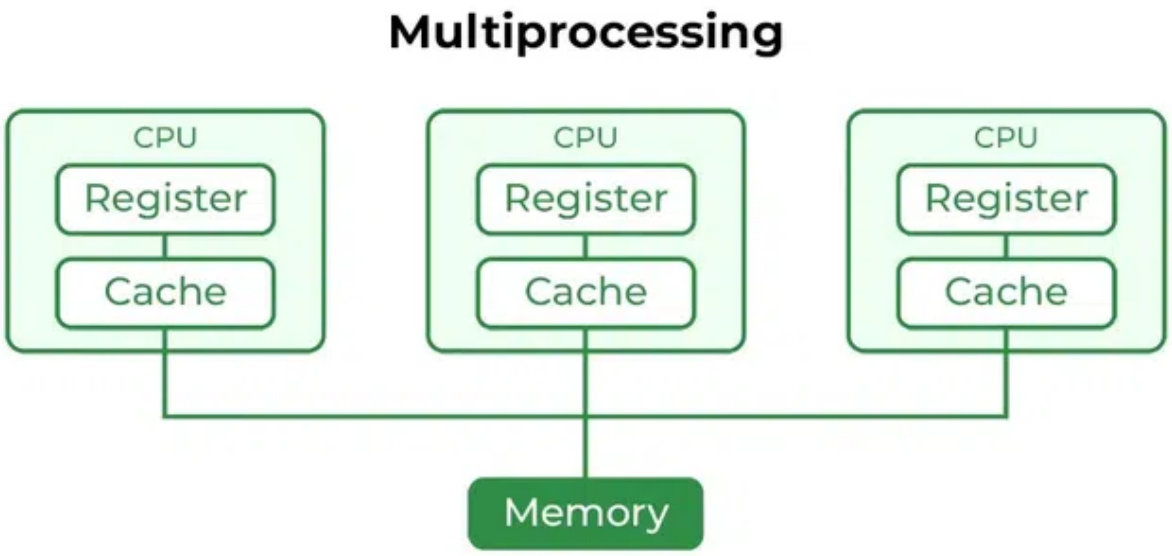
- Multi Programming increases the Throughput of the System.
- It helps in reducing the response time.

Disadvantages of Multi-Programming Operating System

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

4. 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

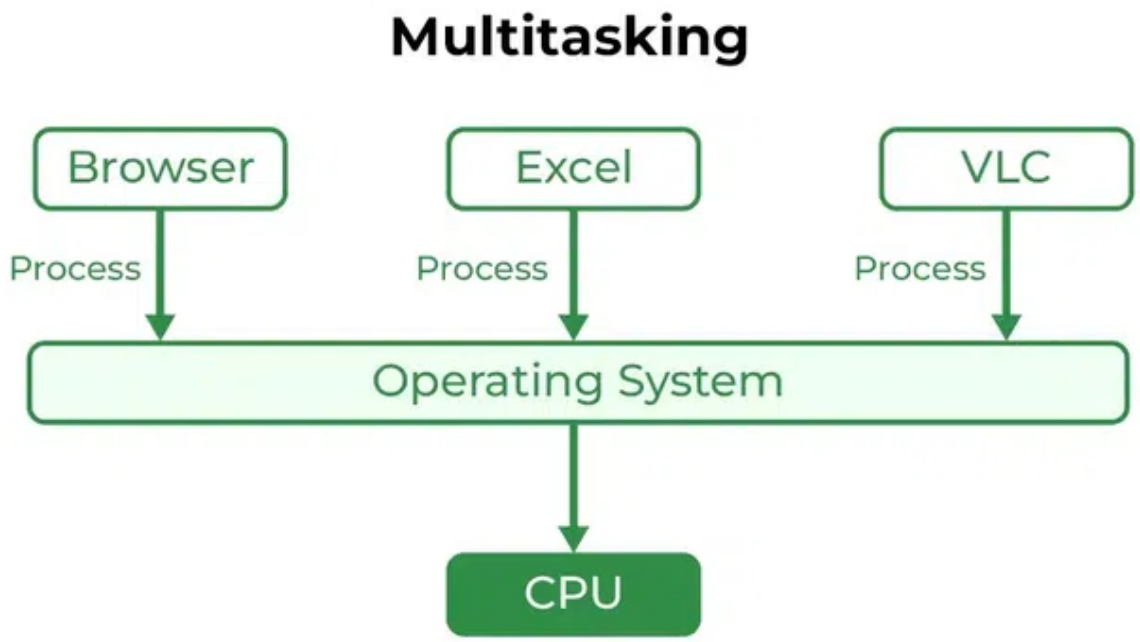
- Due to the multiple CPU, it can be more complex and somehow difficult to understand.

5. 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.

- **Preemptive Multi-Tasking**
- **Cooperative Multi-Tasking**



Multitasking Operating System

Advantages of Multi-Tasking Operating System

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

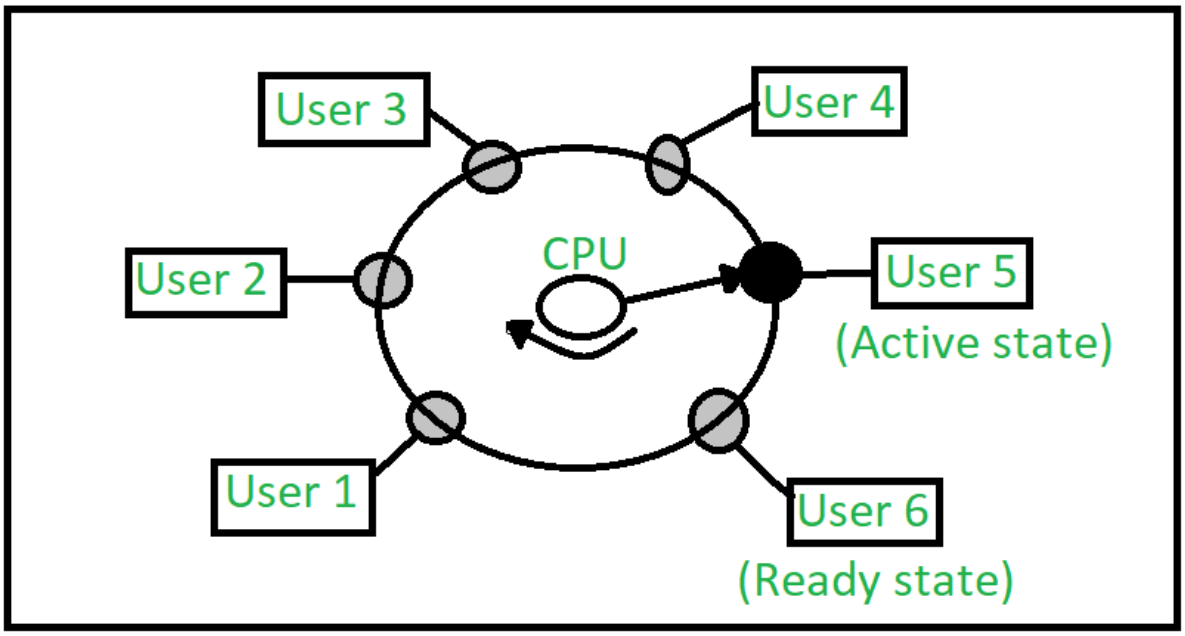
Disadvantages of Multi-Tasking Operating System

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

6. 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.



Time-Sharing OS

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.

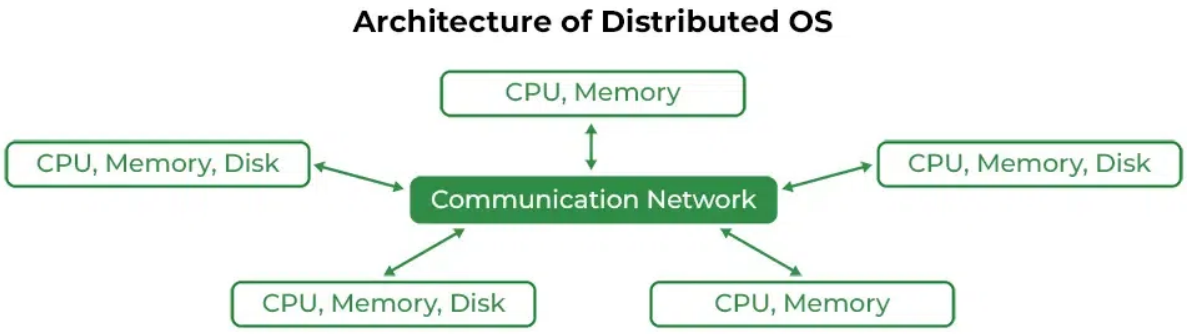
Examples of Time-Sharing OS with explanation

- **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.

7. Distributed Operating System

A Distributed Operating System refers to a model in which applications run on multiple interconnected computers, offering enhanced communication and integration capabilities compared to a network operating system.

In a Distributed Operating System, multiple CPUs are utilized, but for end-users, it appears as a typical centralized operating system. It enables the sharing of various resources such as CPUs, disks, network interfaces, nodes, and computers across different sites, thereby expanding the available data within the entire system.



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.

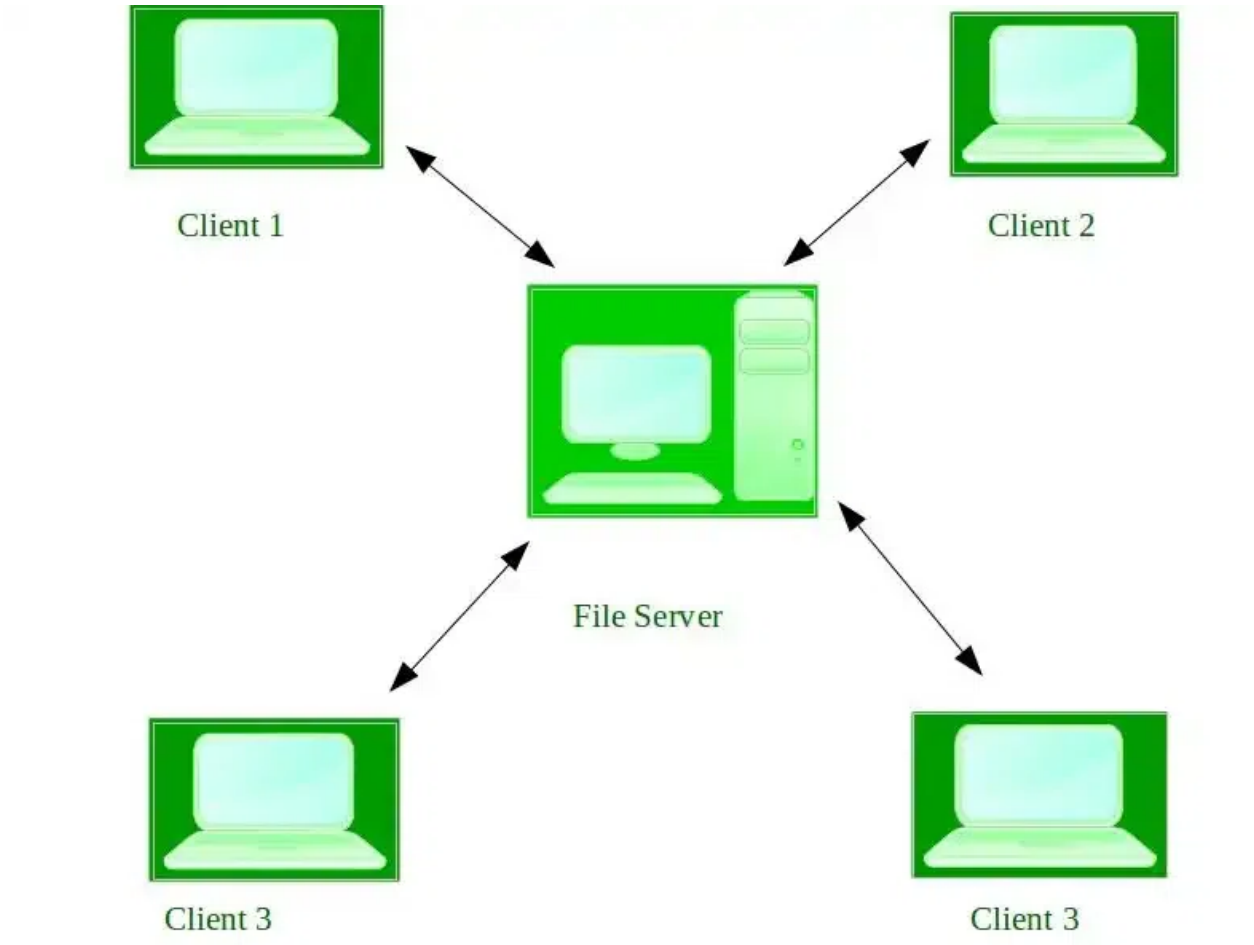
Examples of Distributed Operating Systems are LOCUS, etc.

Issues With Distributed Operating Systems

- Networking causes delays in the transfer of data between nodes of a distributed system. Such delays may lead to an inconsistent view of data located in different nodes, and make it difficult to know the chronological order in which events occurred in the system.
- Control functions like scheduling, resource allocation, and deadlock detection have to be performed in several nodes to achieve computation speedup and provide reliable operation when computers or networking components fail.
- Messages exchanged by processes present in different nodes may travel over public networks and pass through computer systems that are not controlled by the distributed operating system. An intruder may exploit this feature to tamper with messages, or create fake messages to fool the authentication procedure and masquerade as a user of the system.

8. 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** .



Network Operating System

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.

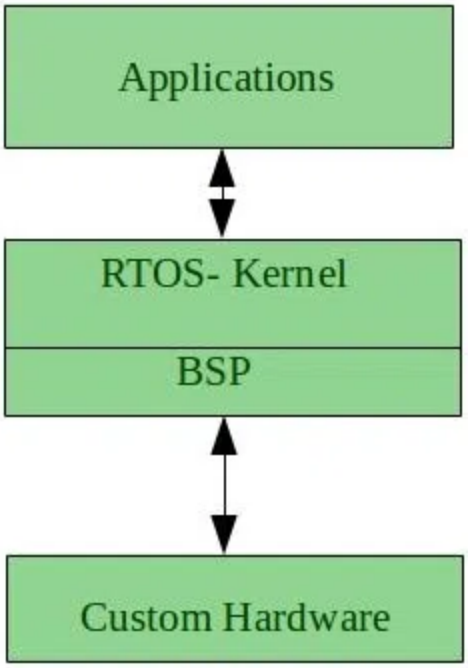
Examples of Network Operating Systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, BSD, etc.

9. 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 Systems

- **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

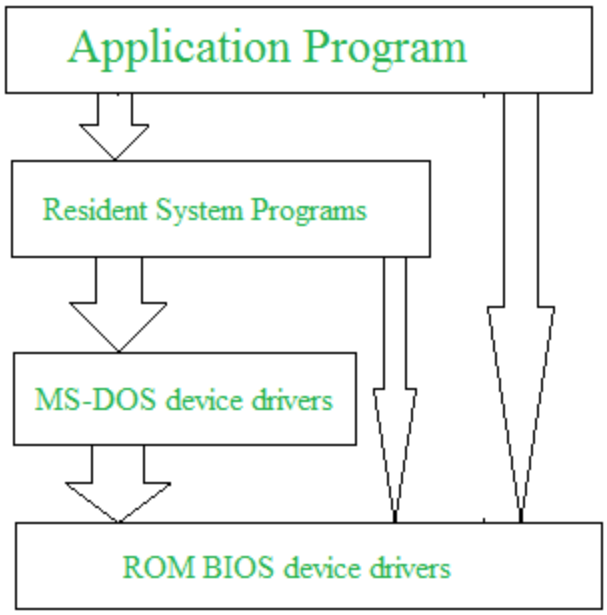
- **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on a few applications to avoid errors.
- **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms:** The algorithms are very complex 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.

OS Structure

Simple/Monolithic structure

Such operating systems do not have well-defined structures and are small, simple, and limited. The interfaces and levels of functionality are not well separated. **MS-DOS** is an example of such an operating system. In MS-DOS, application programs are able to access the basic I/O routines. These types of operating systems cause the entire system to crash if one of the user programs fails.



Advantages of Simple/Monolithic Structure

- It delivers better application performance because of the few interfaces between the application program and the hardware.
- It is easy for kernel developers to develop such an operating system.

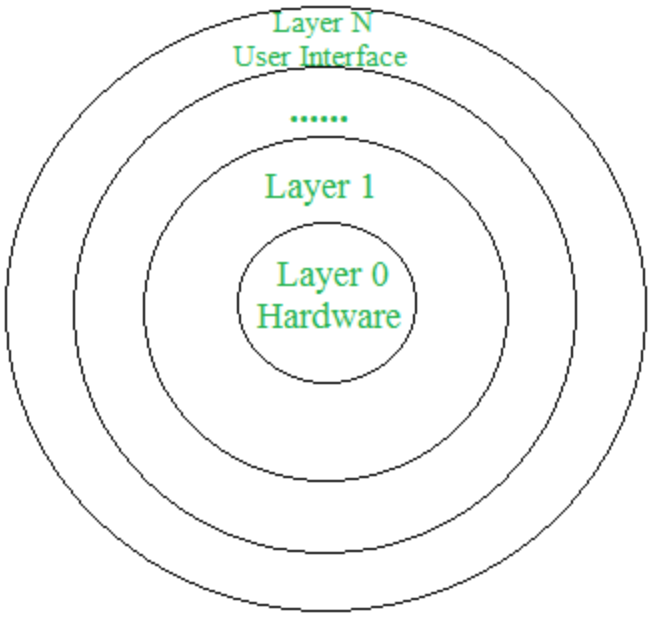
Disadvantages of Simple/Monolithic Structure

- The structure is very complicated, as no clear boundaries exist between modules.
- It does not enforce data hiding in the operating system.

Layered Structure

An OS can be broken into pieces and retain much more control over the system. In this structure, the OS is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware, and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower-level layers. This simplifies the debugging process, if lower-level layers are debugged and an error occurs during debugging, then the error must be on that layer only, as the lower-level layers have already been debugged.

The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover, careful planning of the layers is necessary, as a layer can use only lower-level layers. **UNIX** is an example of this structure.



Advantages of Layered Structure

- Layering makes it easier to enhance the operating system, as the implementation of a layer can be changed easily without affecting the other layers.
- It is very easy to perform debugging and system verification.

Disadvantages of Layered Structure

- In this structure, the application's performance is degraded as compared to simple structure.
- It requires careful planning for designing the layers, as the higher layers use the functionalities of only the lower layers.

What is a System Call?

A system call is a mechanism used by programs to request services from the operating system (OS). In simpler terms, it is a way for a program to interact with the underlying system, such as accessing hardware resources or performing privileged operations.

A system call is initiated by the program executing a specific instruction, which triggers a switch to kernel mode, allowing the program to request a service from the OS. The OS then handles the request, performs the necessary operations, and returns the result back to the program.

Services Provided by System Calls

- Process Creation and Management
- Main **Memory Management**
- File Access, Directory, and **File System Management**
- Device Handling(I/O)
- Protection
- Networking, etc.
    - **Process Control:** end, abort, create, terminate, allocate, and free memory.
    - **File Management:** create, open, close, delete, read files, etc.
    - **Device Management**
    - **Information Maintenance**
    - **Communication**

Advantages of System Calls

- **Access to Hardware Resources:** System calls allow programs to access hardware resources such as **disk drives**, **printers**, and network devices.
- **Memory Management:** System calls provide a way for programs to allocate and deallocate memory, as well as access memory-mapped hardware

devices.

- **Process Management:** System calls allow programs to create and terminate processes, as well as manage inter-process communication.
- **Security:** System calls provide a way for programs to access privileged resources, such as the ability to modify system settings or perform operations that require administrative permissions.
- **Standardization:** System calls provide a standardized interface for programs to interact with the operating system, ensuring consistency and compatibility across different hardware platforms and operating system versions.

Disadvantages of System Call

- **Performance Overhead**: System calls involve switching between user mode and kernel mode, which can slow down program execution.
- **Security Risks**: Improper use or vulnerabilities in system calls can lead to security breaches or unauthorized access to system resources.
- **Error Handling Complexity**: Handling errors in system calls, such as resource allocation failures or timeouts, can be complex and require careful **programming**.
- **Compatibility Challenges**: System calls may vary between different **operating systems**, requiring developers to write code that works across multiple platforms.
- **Resource Consumption**: System calls can consume significant system resources, especially in environments with many concurrent processes making frequent calls.

Examples of a System Call in Windows and Unix

System calls for Windows and Unix come in many different forms. These are listed in the table below as follows:

Process	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	Fork() Exit() Wait()
File manipulation	CreateFile() ReadFile() WriteFile()	Open() Read() Write() Close()
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	Ioctl() Read() Write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	Getpid() Alarm() Sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	Pipe() Shmget() Mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorgroup()	Chmod() Umask() Chown()

What is Kernel?

A kernel is a central component of an operating system that manages the operations of computers and hardware. It basically manages operations of memory and CPU time. It is a core component of an operating system. Kernel acts as a bridge between applications and data processing performed at the hardware level using inter-process communication and system calls.

A kernel is the core part of an operating system. It acts as a bridge between software applications and the hardware of a computer. The kernel manages system resources, such as the CPU, memory, and devices, ensuring everything works together smoothly and efficiently. It handles tasks like running programs, accessing files, and connecting to devices like printers and keyboards.

Types of Kernel

The kernel manages the system's resources and facilitates communication between hardware and software components. These kernels are of different types let's discuss each type along with its advantages and disadvantages:

1. Monolithic Kernel

It is one of the types of kernel where all operating system services operate in kernel space. It has dependencies between systems components. It has huge lines of code which is complex.

Example:

```
Unix, Linux, Open VMS, XTS-400 etc.
```

Advantages

- **Efficiency:** Monolithic kernels are generally faster than other types of kernels because they don't have to switch between user and kernel modes for every system call, which can cause overhead.
- **Tight Integration:** Since all the operating system services are running in kernel space, they can communicate more efficiently with each other, making it easier to implement complex functionalities and optimizations.
- **Simplicity:** Monolithic kernels are simpler to design, implement, and debug than other types of kernels because they have a unified structure that makes it easier to manage the code.
- **Lower latency:** Monolithic kernels have lower latency than other types of kernels because system calls and **interrupts** can be handled directly by the kernel.

Disadvantages

- **Stability Issues:** Monolithic kernels can be less stable than other types of kernels because any bug or security vulnerability in a kernel service can affect the entire system.

- **Security Vulnerabilities:** Since all the operating system services are running in kernel space, any security vulnerability in one of the services can compromise the entire system.
- **Maintenance Difficulties:** Monolithic kernels can be more difficult to maintain than other types of kernels because any change in one of the services can affect the entire system.
- **Limited Modularity:** Monolithic kernels are less modular than other types of kernels because all the operating system services are tightly integrated into the kernel space. This makes it harder to add or remove functionality without affecting the entire system.

2. Micro Kernel

It is kernel types which has minimalist approach. It has virtual memory and thread scheduling. It is more stable with less services in kernel space. It puts rest in user space. It is use in small os.

Example :

```
Mach, L4, AmigaOS, Minix, K42 etc.
```

Advantages

- **Reliability:** Microkernel architecture is designed to be more reliable than monolithic kernels. Since most of the operating system services run outside the kernel space, any bug or security vulnerability in a service won't affect the entire system.
- **Flexibility**: Microkernel architecture is more flexible than monolithic kernels because it allows different operating system services to be added or removed without affecting the entire system.
- **Modularity:** Microkernel architecture is more modular than monolithic kernels because each operating system service runs independently of the others. This makes it easier to maintain and debug the system.

- **Portability:** Microkernel architecture is more portable than monolithic kernels because most of the operating system services run outside the kernel space. This makes it easier to port the operating system to different hardware architectures.

Disadvantages

- **Performance:** Microkernel architecture can be slower than monolithic kernels because it requires more context switches between user space and kernel space.
- **Complexity:** Microkernel architecture can be more complex than monolithic kernels because it requires more communication and **synchronization mechanisms** between the different operating system services.
- **Development Difficulty:** Developing operating systems based on microkernel architecture can be more difficult than developing monolithic kernels because it requires more attention to detail in designing the communication and synchronization mechanisms between the different services.
- **Higher Resource Usage:** Microkernel architecture can use more system resources, such as memory and CPU, than monolithic kernels because it requires more communication and synchronization mechanisms between the different operating system services.

3. Hybrid Kernel

It is the combination of both monolithic kernel and microkernel. It has speed and design of monolithic kernel and modularity and stability of microkernel.

Example :

```
Windows NT, Netware, BeOS etc.
```

Advantages

- **Performance:** Hybrid kernels can offer better performance than microkernels because they reduce the number of context switches required between user space and kernel space.

- **Reliability:** Hybrid kernels can offer better reliability than monolithic kernels because they isolate **drivers** and other kernel components in separate protection domains.
- **Flexibility:** Hybrid kernels can offer better flexibility than monolithic kernels because they allow different operating system services to be added or removed without affecting the entire system.
- **Compatibility:** Hybrid kernels can be more compatible than microkernels because they can support a wider range of device drivers.

Disadvantages

- **Complexity:** Hybrid kernels can be more complex than monolithic kernels because they include both monolithic and microkernel components, which can make the design and implementation more difficult.
- **Security:** Hybrid kernels can be less secure than microkernels because they have a larger attack surface due to the inclusion of monolithic components.
- **Maintenance:** Hybrid kernels can be more difficult to maintain than microkernels because they have a more complex design and implementation.
- **Resource Usage:** Hybrid kernels can use more system resources than microkernels because they include both monolithic and microkernel components.

4. Exo Kernel

It is the type of kernel which follows end-to-end principle. It has fewest hardware abstractions as possible. It allocates physical resources to applications.

Example :

```
Nemesis, ExOS etc.
```

Advantages

- **Flexibility:** Exokernels offer the highest level of flexibility, allowing developers to customize and optimize the operating system for their specific application

needs.

- **Performance:** Exokernels are designed to provide better performance than traditional kernels because they eliminate unnecessary abstractions and allow applications to directly access hardware resources.
- **Security:** Exokernels provide better security than traditional kernels because they allow for fine-grained control over the allocation of system resources, such as memory and **CPU** time.
- **Modularity:** Exokernels are highly modular, allowing for the easy addition or removal of operating system services.

Disadvantages

- **Complexity:** Exokernels can be more complex to develop than traditional kernels because they require greater attention to detail and careful consideration of system resource allocation.
- **Development Difficulty:** Developing applications for exokernels can be more difficult than for traditional kernels because applications must be written to directly access hardware resources.
- **Limited Support:** Exokernels are still an emerging technology and may not have the same level of support and resources as traditional kernels.
- **Debugging Difficulty:** Debugging applications and operating system services on exokernels can be more difficult than on traditional kernels because of the direct access to hardware resources.

5. Nano Kernel

It is the type of kernel that offers hardware abstraction but without system services. Micro Kernel also does not have system services therefore the Micro Kernel and Nano Kernel have become analogous.

Example :

```
EROS etc.
```

Advantages

- **Small Size:** Nanokernels are designed to be extremely small, providing only the most essential functions needed to run the system. This can make them more efficient and faster than other kernel types.
- **High Modularity:** Nanokernels are highly modular, allowing for the easy addition or removal of operating system services, making them more flexible and customizable than traditional monolithic kernels.
- **Security:** Nanokernels provide better security than traditional kernels because they have a smaller attack surface and a reduced risk of errors or bugs in the code.
- **Portability:** Nanokernels are designed to be highly portable, allowing them to run on a wide range of hardware architectures.

Disadvantages

- **Limited Functionality:** Nanokernels provide only the most essential functions, making them unsuitable for more complex applications that require a broader range of services.
- **Complexity:** Because nanokernels provide only essential functionality, they can be more complex to develop and maintain than other kernel types.
- **Performance:** While nanokernels are designed for efficiency, their minimalist approach may not be able to provide the same level of performance as other kernel types in certain situations.
- **Compatibility:** Because of their minimalist design, nanokernels may not be compatible with all hardware and software configurations, limiting their practical use in certain contexts.

Functions of Kernel

The kernel is responsible for various critical functions that ensure the smooth operation of the computer system. These functions include:

1. Process Management

- Scheduling and execution of processes.
- Context switching between processes.
- Process creation and termination.

2. Memory Management

- Allocation and deallocation of memory space.
- Managing virtual memory.
- Handling memory protection and sharing.

3. Device Management

- Managing input/output devices.
- Providing a unified interface for hardware devices.
- Handling device driver communication.

4. File System Management

- Managing file operations and storage.
- Handling file system mounting and unmounting.
- Providing a file system interface to applications.

5. Resource Management

- Managing system resources (CPU time, disk space, network bandwidth)
- Allocating and deallocating resources as needed
- Monitoring resource usage and enforcing resource limits

6. Security and Access Control

- Enforcing access control policies.
- Managing user permissions and authentication.
- Ensuring system security and integrity.

7. Inter-Process Communication

- Facilitating communication between processes.
- Providing mechanisms like message passing and shared memory.

Working of Kernel

- A kernel loads first into memory when an operating system is loaded and remains in memory until the operating system is shut down again. It is responsible for various tasks such as **disk management**, task management, and **memory management**.
- The kernel has a process table that keeps track of all active processes
- The process table contains a per-process region table whose entry points to entries in the region table.
- The kernel loads an executable file into memory during the 'exec' system call'.
- It decides which process should be allocated to the processor to execute and which process should be kept in the main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of the kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

Objectives of Kernel

- To establish communication between user-level applications and hardware.
- To decide the state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.

1. Android OS

Android is an open-source mobile operating system developed by Google, primarily for touch devices like smartphones and tablets.

Key Design Features:

- **Linux Kernel**: Android is based on the Linux kernel, providing low-level memory management, process control, networking, and security.
- **Java/Kotlin Application Framework**: The apps are primarily built using Java or Kotlin, running on a virtual machine (Android Runtime - ART).
- **Modular Architecture**: Android uses a modular system, where the hardware abstraction layer (HAL) allows easy integration with various hardware platforms.
- **Material Design**: Android follows Material Design guidelines for UI and UX consistency. It emphasizes bold colors, responsive animations, and padding for touch-friendly navigation.
- **Customizability**: It is open-source and allows device manufacturers and users to customize the OS heavily.

System Components:

- **Activity Manager**: Manages the lifecycle of applications.
- **Content Providers**: Allows data sharing between apps.
- **Broadcast Receivers**: Enables apps to listen to system-wide announcements like battery status or network changes.
- **Sensors and Inputs**: Supports a variety of sensors for location, orientation, etc.

2. iOS

iOS is Apple's mobile operating system for its iPhone and iPad product lines.

Key Design Features:

- **UNIX-based Kernel**: Built on the Darwin operating system (BSD Unix-based), it provides low-level system functionalities.
- **Cocoa Touch Framework**: Provides key UI controls and is designed around event-driven programming and multitouch gestures.

- **App Ecosystem**: iOS apps are primarily developed using Swift or Objective-C, with strict guidelines for performance, security, and privacy.
- **Human Interface Guidelines (HIG)**: Apple follows specific HIG for the design of apps, focusing on simplicity, consistency, and user experience. The design emphasizes minimalism, clarity, and intuitiveness.
- **Closed Ecosystem**: iOS is more tightly controlled, with less customizability, ensuring consistency and security across devices.

System Components:

- **Core OS**: Manages system functions such as memory, file system, and networking.
- **Core Services**: Provides APIs for data, networking, and security.
- **Core Animation**: Handles advanced visual animations and transitions.
- **UIKit**: Manages the user interface and handles touch interactions.

3. Virtual OS

A virtual operating system is an environment that simulates the functionality of an OS, running on top of another host operating system.

Key Design Features:

- **Hypervisor**: A key component, it creates and runs virtual machines (VMs). A hypervisor can be type 1 (bare-metal) or type 2 (running on a host OS).
- **Resource Management**: Virtual OS environments share resources like CPU, memory, and storage from the host system.
- **Isolation**: VMs provide isolation, meaning different OSes can run independently on the same physical hardware without interfering.
- **Snapshot Functionality**: Virtual OSes allow taking snapshots of the current state, enabling easy backups and rollbacks.

Common Use Cases:

- **Development and Testing**: Developers use Virtual OS for testing different OS environments without needing dedicated hardware.
- **Cloud Infrastructure**: Many cloud services are powered by virtualized environments to provide scalable services.

4. Cloud OS

Cloud OS (or Cloud Operating Systems) refers to systems designed to manage cloud computing environments, including virtualized resources, storage, networking, and computing power.

Key Design Features:

- **Abstraction Layer**: A Cloud OS abstracts and virtualizes underlying hardware resources, allowing for the pooling of resources and scalability.
- **Distributed Systems**: It is designed for distributed computing where resources are spread across different data centers and connected via the internet.
- **On-Demand Scalability**: Cloud OS supports dynamic resource allocation depending on the needs of applications.
- **Fault Tolerance**: Built-in mechanisms to handle hardware and software failures without service interruption.
- **Multi-Tenancy**: Supports multiple users (tenants) running applications in a shared environment with isolated resources.

Examples:

- **OpenStack**: An open-source cloud OS managing large pools of compute, storage, and networking resources.
- **Microsoft Azure**: Provides cloud services, including hosting and running applications on distributed servers.
- **Google Cloud OS**: Google's distributed OS manages its cloud infrastructure.

Each of these operating systems has unique design elements tailored to their use cases—mobile interaction, secure closed ecosystems, virtualization, or cloud infrastructure management.