1. **Define as OS. Explain the functions of OS. Differentiate between time sharing and real time OS.**

Ans:It is a collection of programs that acts as an interface between a user apps and computer hardware. It hides the complexities of hardware from the users and serves as a platform for runnig and serves as a platform for runnig the application softwares. Eg:windows, linux

Functions:

-Memory management: It refers to the managemnet fo primary memory . Main memory consists of large array of words/bytes where each word/byte has its own address.

-Process Management:In multiprogramming environment , OS decides which process get the processor anf for how mucht itme. The functio is called process scheduling.

-Device Management: OS manages device communictaion through respective drivers

->It decides which process gets the OS when and how much time

->keep track of all devices using I/O controllers.

-File Management: A file system is organizedo input directories for easy navigation and use.

|  |  |
| --- | --- |
| Time Sharing | Real Time |
| 1.Quick response is emphasized for a request. | 1.Computation tasks are emphasized before its nominative point. |
| 2.switching function is available | 2.while in this OS switching method is not available |
| 3.Any modification in the program can be possible. | 3.While in this modification does not take place |
| 4.computer resources are shared to the external | 1. But in this OS, computer resources are not shared to the external |
| 5.it deals with more that processes or applications simultaneously. | 5. it deals with only one process or application at a time |

1. **List the types of OS and explain all. What are the objectives of OS? Explain.**

**ans:Types of OS:**

**Batch Operating Systems**

These OS do not interact with the computer directly. Jobs with similar needs are batched together and executed by an operator. They are efficient in managing large work repeatedly and have less idle time. However, they can be complex to debug and may lead to delays if a job fails.

**Multi-Programming Operating Systems**

In multi-programming systems, multiple programs reside in the main memory, and the OS manages their execution to improve resource utilization. This leads to increased throughput and reduced response time, but lacks user interaction with system resources.

**Multi-Processing Operating Systems**

These systems utilize more than one CPU to execute processes, improving throughput and providing redundancy in case of CPU failure. However, they can be complex due to the presence of multiple CPUs.

**Multi-Tasking Operating Systems**

Multi-tasking OS can run multiple programs simultaneously using algorithms like Round-Robin Scheduling. They provide efficient memory management but may cause the system to heat up with heavy programs.

**Time-Sharing Operating Systems**

Time-sharing OS allocate CPU time to each task so that all tasks can run smoothly. They offer equal opportunity to each task and reduce CPU idle time. However, they come with higher overhead and increased complexity due to the need for scheduling and context switching.

**Distributed Operating Systems**

These systems consist of interconnected computers that communicate over a network. They allow remote access to files and software, providing fast computation and scalability. However, the failure of the main network can halt communication, and the underlying software can be complex.

**Network Operating Systems**

Network OS run on servers and manage data, users, security, and other networking functions. They offer centralized control but require regular maintenance and updates. Examples include Microsoft Windows Server and UNIX.

**Real-Time Operating Systems**

Real-time OS are designed for systems with strict timing constraints, like missile systems or robots. They are categorized into hard and soft real-time systems, with hard real-time systems having stricter timing constraints.

**Objectives of OS**

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

**3. OS acts as resource manager and as an extended machine. Explain in detail.**

Ans:

**Operating System as Extended Machine**

Let us investigate how the operating system functions as an Extended Machine.

-At the Machine level the structure of a computer’s system is complicated to program, mainly for input or output. Programmers do not deal with hardware. They will always mainly focus on implementing software. Therefore, a level of abstraction is supposed to be maintained.

-Operating systems provide a layer of abstraction for using disk such as files.

-This **[level of abstraction](https://www.tutorialspoint.com/what-is-data-abstraction-in-dbms" \t "https://www.tutorialspoint.com/_blank)** allows a program to create, write, and read files, without dealing with the details of how the hardware actually works.

-The level of abstraction is the key to managing the complexity.

-Good abstractions turn an impossible task into two manageable tasks.

-The first is to define and implement the abstractions.

-The second is to solve the problem at hand.

-Operating system provides abstractions to **[application programs](https://www.tutorialspoint.com/computer_fundamentals/computer_software.htm" \t "https://www.tutorialspoint.com/_blank)** in a top down view.

## Operating System as Resource Manager

-Now-a-days all modern computers consist of **[processors](https://www.tutorialspoint.com/embedded_systems/es_processors.htm" \t "https://www.tutorialspoint.com/_blank)**, **[memories](https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm" \t "https://www.tutorialspoint.com/_blank)**, **[timers](https://www.tutorialspoint.com/embedded_systems/es_timer_counter.htm" \t "https://www.tutorialspoint.com/_blank)**, network interfaces, printers, and so many other devices.

-The operating system provides for an orderly and controlled allocation of the processors, memories, and I/O devices among the various programs in the bottom-up view.

-Operating system allows multiple programs to be in memory and run at the same time.

**-[Resource management](https://www.tutorialspoint.com/operating-system-resource-management" \t "https://www.tutorialspoint.com/_blank)** includes multiplexing or sharing resources in two different ways: in time and in space.

-In time multiplexed, different programs take a chance of using **[CPU](https://www.tutorialspoint.com/computer_fundamentals/computer_cpu.htm" \t "https://www.tutorialspoint.com/_blank)**. First one tries to use the resource, then the next one that is ready in the queue and so on. For example: Sharing the printer one after another.

-In space multiplexing, Instead of the customers taking a chance, each one gets part of the resource. For example − **[Main memory](https://www.tutorialspoint.com/what-is-the-main-memory" \t "https://www.tutorialspoint.com/_blank)** is divided into several running programs, so each one can be resident at the same time.

**4 What are two modes of OS? Describe the structure of OS with suitable example.**

**Ans:** An error in one program can adversely affect many processes, it might modify data of another program or also can affect the operating system. For example, if a process stuck in the infinite loop then this infinite loop could affect the correct operation of other processes. So to ensure the proper execution of the operating system, there are two modes of operation:

**User mode –**   
When the computer system is run by user applications like creating a text document or using any application program, then the system is in user mode. When the user application requests for a service from the operating system or an interrupt occurs or [system call](https://www.geeksforgeeks.org/operating-system-introduction-system-call/), then there will be a transition from user to kernel mode to fulfill the requests.

**Kernel Mode:**When the system boots, the hardware starts in kernel mode and when the operating system is loaded, it starts user application in user mode. To provide protection to the hardware, we have privileged instructions which execute only in kernel mode. If the user attempts to run privileged instruction in user mode then it will treat instruction as illegal and traps to OS. Some of the privileged instructions are:

1. Handling Interrupts
2. To switch from user mode to kernel mode.
3. Input-Output management.

**Structure of OS**

\*Simple Structure: These type of operating system have simple system and rapidly expanded much further than their scope. Eg:MS DOS. It was designed simply for a small groups of people and company.

Application program

System program

MS DOS device Driver

Fig: MS - DOS Structure

ROM BIOS Device Drivers

\*Layered Structure:Organising OS as a hierarchy of layers each one constructed upon one below it. One way to achieve modularity is layered approach. Bottom layer is hardware and topmost layer is UI.

Layer Function:

1. Operator
2. Uses program
3. I/O management
4. Operator process
5. Memory management
6. Processor allocation and multi programming

\*Monotholic System: These systems are written as a collection of procedures, each of which can call any of the other ones whenever it needs to . When this technique is used, each procedure in the system has a well defined interface in terms of parameters and results, and each one is free to casll any other one.

This organization suggests a basic structure for the OS:

-A main program that invokes the requested service procedure.

-A set of service procedures that carry out system calls.

-A set of utility procedures that help the services procedures.

**5 Define kernel. Describe different types of kernels with suitable example.**

**Ans:Kernel**

A kernel is the core component of an operating system that manages system resources and facilitates communication between hardware and software. It provides essential services such as process management, memory management, device management, and system calls.

**Types of Kernels**

**a Monolithic Kernels**

In monolithic kernels, both user services and kernel services are implemented in the same memory space. This design increases the kernel's size and, consequently, the operating system's size. With no separation between user space and kernel space, process execution is faster.

Example:

- Linux: The Linux kernel is a monolithic kernel, where all essential services run in the same memory space for efficiency and speed.

**b Microkernel**

In microkernels, user services and kernel services are implemented in separate spaces, reducing the kernel's size and the overall operating system's size. Communication between applications and services is done through message passing, which can reduce execution speed.

Example:

- Minix: Minix is an example of a microkernel, where only essential functions are included in the kernel, and other services run in user space.

**c Hybrid Kernel**

A hybrid kernel combines the features of both monolithic and microkernels. It uses the speed of a monolithic kernel and the modularity of a microkernel. Some non-essential code runs in kernel space for speed, while other services run in user space to maintain modularity.

Example:

- Windows NT:The Windows NT kernel is a hybrid kernel, combining elements of both monolithic and microkernel architectures.

**d. Nanokernel**

A nanokernel is designed to have a very small code base, executing only the minimal code required in the privileged mode of the hardware. It supports a nanosecond clock resolution, emphasizing a minimalistic approach.

Example:

- Specific implementations of real-time operating systems (RTOS) use nanokernels to achieve high precision and minimal latency.

**e Exokernel**

Exokernels are designed to separate resource protection from management, allowing for application-specific customization. They do not enforce abstractions, which allows for more flexible and efficient use of hardware resources. However, the design of exokernels is complex.

Example:

- MIT Exokernel: Developed by MIT, this kernel allows applications to manage resources directly, providing high flexibility and performance.

**6. Define shell. How it is different from kernel? Explain.**

Ans:A shell is an environmnet in which we can run our command, program and scripts.

-It provieds and interface to UNIX/LINUX system.

-It gathers inputs from user and execute programs based on the input. When program finishes execution, it displays program output.

-Shell prompt is given by sign “$” which is called command prompt. It reads input after pressing “enter $date  
Fri 10 may 2024

-When any user logs in a shell is started up. It has the terminal as standard input and output.

**Shell:**

* **Environment:** The shell provides an environment for running commands and scripts. It allows users to execute programs and manage files.
* **Interface:** Acts as an interface between the user and the system, interpreting user commands and converting them into actions performed by the OS.
* **Command Prompt:** The command prompt, usually denoted by "$" (or other symbols, depending on the shell), is where users enter their commands. The shell reads the input, executes the command, and displays the output.

**Kernel:**

* **Core Component:** The kernel is the core part of the operating system, responsible for managing hardware resources and system operations.
* **Resource Management:** It handles CPU scheduling, memory management, device control, and system calls.
* **Isolation:** The kernel operates in a protected memory space (kernel space), separate from user applications (user space), ensuring stability and security.

1. **What is system call? Discuss process of handling system calls briefly.**

**ans:**The interface between a process and an OS is provided by system call. System calls are usually made when a process in user mode requires access to resources. Then it requests their kernel to provide the resource through a system call.

Handling system calls

Step 1: The process executed in user mode till time a system calls interrupts it.

Step 2: After that , the system call is executed I the kernel mode on a priority basis.

Step 3: Once system call execution is over control returns to the user mode.

Step 4: The execution of user processes resumed in Kernel mode.

1. **What are the main differences between OS for mainframe computer and personal computer?**

**Ans:**

| **Feature** | **Mainframe Computers** | **Personal Computers** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Purpose and Usage** | Designed for large-scale, high-volume transaction processing and complex computations. Utilized in enterprise environments like banks, insurance companies, and government agencies to support multiple users and applications simultaneously. | Designed for individual use, suitable for general computing tasks like word processing, browsing, and multimedia. Used by individuals in homes, offices, and small businesses for personal and professional tasks. |

|  |  |  |
| --- | --- | --- |
| **Scalability and Performance** | Highly scalable, capable of supporting thousands of simultaneous users and processes. Optimized for maximum throughput, reliability, and handling massive workloads efficiently. | Limited scalability, generally supporting a single user or a small number of users. Optimized for responsiveness and usability for individual tasks, with a focus on graphical interfaces and multimedia capabilities. |

|  |  |  |
| --- | --- | --- |
| **Resource Management** | Advanced resource management, including sophisticated memory management, I/O handling, and job scheduling. Extensive use of virtualization to create multiple virtual machines (VMs), each running its own OS and applications. | Simpler resource management designed for typical consumer applications and peripherals. Limited virtualization capabilities, often used for testing and development purposes. |

|  |  |  |
| --- | --- | --- |
| **Reliability and Availability** | High reliability with features such as redundancy, fault tolerance, and automated recovery. Designed for high availability, often achieving 99.999% uptime (five nines). | Standard reliability suitable for individual use, with regular updates and backups recommended. Availability depends on hardware quality and maintenance; not as critical as in mainframes. |

|  |  |  |
| --- | --- | --- |
| **Security** | Robust security features including advanced access controls, encryption, and auditing. Strong isolation between different users and applications to prevent unauthorized access and interference. | Basic security features including user authentication, firewalls, and antivirus software. Limited isolation, primarily focused on user accounts and permissions. |

|  |  |  |
| --- | --- | --- |
| **Operating Systems Examples** | IBM z/OS, IBM z/VM, Unisys OS 2200, specifically designed for the complex requirements of mainframe environments. | Microsoft Windows, macOS, various Linux distributions, designed for ease of use, graphical interfaces, and support for a wide range of consumer applications. |