**Assignment-1**

1. What is Operating System? Discuss role/functions of OS as a resource manager.

An operating system is the interface between the user and the machine which controls and coordinates the use of the hardware among the various application programs for the various users.

Operating System as Resource Manager

Let us understand how the operating system works as a Resource Manager.

* Now-a-days all modern computers consist of processors, memories, timers, 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 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. 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 is divided into several running programs, so each one can be resident at the same time.

1. Write different operating system services and describe it with suitable example.

Services of Operating System

1. Program Execution
2. Input Output Operations
3. File Management
4. Error Handling
5. Resource Management
6. Communication between Processes

Program Execution:

It is the Operating System that manages how a program is going to be executed. It loads the program into the memory after which it is executed. The order in which they are executed depends on the CPU Scheduling Algorithms. A few are FCFS, SJF, etc. When the program is in execution, the Operating System also handles deadlock i.e. no two processes come for execution at the same time. The Operating System is responsible for the smooth execution of both user and system programs. The Operating System utilizes various resources available for the efficient running of all types of functionalities.

Input Output Operations:

Operating System manages the input-output operations and establishes communication between the user and device drivers. Device drivers are software that is associated with hardware that is being managed by the OS so that the sync between the devices works properly. It also provides access to input-output devices to a program when needed.

File Management:

The operating system helps in managing files also. If a program needs access to a file, it is the operating system that grants access. These permissions include read-only, read-write, etc. It also provides a platform for the user to create, and delete files. The Operating System is responsible for making decisions regarding the storage of all types of data or files, i.e, floppy disk/hard disk/pen drive, etc. The Operating System decides how the data should be manipulated and stored.

Error Handling:

The Operating System also handles the error occurring in the CPU, in Input-Output devices, etc. It also ensures that an error does not occur frequently and fixes the errors. It also prevents the process from coming to a deadlock. It also looks for any type of error or bugs that can occur while any task. The well-secured OS sometimes also acts as a countermeasure for preventing any sort of breach of the Computer System from any external source and probably handling them.

Resource Management:

System resources are shared between various processes. It is the Operating system that manages resource sharing. It also manages the CPU time among processes using CPU Scheduling Algorithms. It also helps in the memory management of the system. It also controls input-output devices. The OS also ensures the proper use of all the resources available by deciding which resource to be used by whom.

Communication between Processes:

The Operating system manages the communication between processes. Communication between processes includes data transfer among them. If the processes are not on the same computer but connected through a computer network, then also their communication is managed by the Operating System itself.

1. Enlist various generations of operating system and explain them with advantages and disadvantages in detail.

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

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

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

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

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

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

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

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

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

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

## The Fourth Generation ( 1980 - Present ): Personal Computers

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

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

1. Describe RTOS its types and DOS with suitable examples.

**Real-time operating system (RTOS)** is an operating system intended to serve real time application that process data as it comes in, mostly without buffer delay. The full form of RTOS is Real time operating system.

In a RTOS, Processing time requirement are calculated in tenths of seconds increments of time. It is time-bound system that can be defined as fixed time constraints. In this type of system, processing must be done inside the specified constraints. Otherwise, the system will fail.

## Types of RTOS

Three types of RTOS systems are:

### Hard Real Time :

In Hard RTOS, the deadline is handled very strictly which means that given task must start executing on specified scheduled time, and must be completed within the assigned time duration.

Example: Medical critical care system, Aircraft systems, etc.

### Firm Real time:

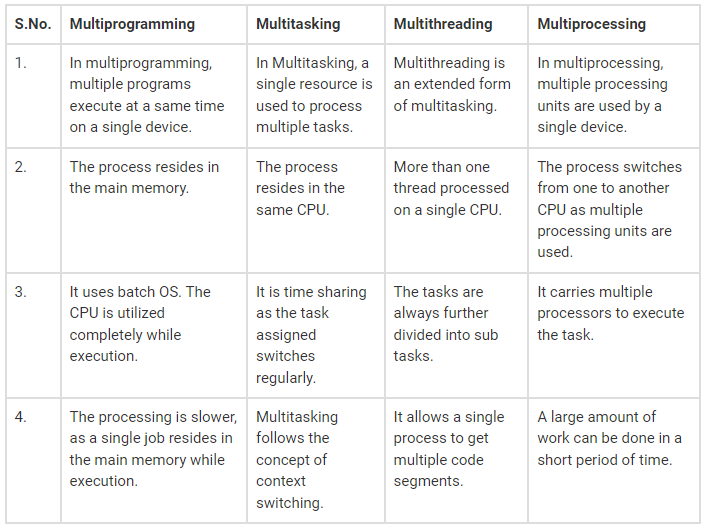
These type of RTOS also need to follow the deadlines. However, missing a deadline may not have big impact but could cause undesired affects, like a huge reduction in quality of a product.

Example: Various types of Multimedia applications.

### Soft Real Time:

Soft Real time RTOS, accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS.

1. Identify the differences between Multi-Programming, Multi-tasking, and Multiprocessing System.



1. What are System Calls? Explain various types of system calls with appropriate examples.

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers.

System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.



1. What is an interrupt and context switching? How it is handle by operating system.

Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.

A diagram that demonstrates context switching is as follows −

In the above diagram, initially Process 1 is running. Process 1 is switched out and Process 2 is switched in because of an interrupt or a system call. Context switching involves saving the state of Process 1 into PCB1 and loading the state of process 2 from PCB2. After some time again a context switch occurs and Process 2 is switched out and Process 1 is switched in again. This involves saving the state of Process 2 into PCB2 and loading the state of process 1 from PCB1.

## Context Switching Triggers

There are three major triggers for context switching. These are given as follows −

* **Multitasking:** In a multitasking environment, a process is switched out of the CPU so another process can be run. The state of the old process is saved and the state of the new process is loaded. On a pre-emptive system, processes may be switched out by the scheduler.
* **Interrupt Handling:** The hardware switches a part of the context when an interrupt occurs. This happens automatically. Only some of the context is changed to minimize the time required to handle the interrupt.
* **User and Kernel Mode Switching:** A context switch may take place when a transition between the user mode and kernel mode is required in the operating system.

## Context Switching Steps

The steps involved in context switching are as follows −

* Save the context of the process that is currently running on the CPU. Update the process control block and other important fields.
* Move the process control block of the above process into the relevant queue such as the ready queue, I/O queue etc.
* Select a new process for execution.
* Update the process control block of the selected process. This includes updating the process state to running.
* Update the memory management data structures as required.
* Restore the context of the process that was previously running when it is loaded again on the processor. This is done by loading the previous values of the process control block and registers.

## Context Switching Cost

Context Switching leads to an overhead cost because of TLB flushes, sharing the cache between multiple tasks, running the task scheduler etc. Context switching between two threads of the same process is faster than between two different processes as threads have the same virtual memory maps. Because of this TLB flushing is not required.

1. Explain Monolithic, Micro Kernel and Layered system of Operating System Structure.

|  |  |  |
| --- | --- | --- |
| **Terms** | **Monolithic Kernel** | **microkernel** |
| Definition | A monolithic kernel is a type of kernel in operating systems where the entire operating system works in the kernel space. | A microkernel is a kernel type that provides low-level address space management, thread management, and interprocess communication to implement an operating system. |
| Address space | In a monolithic kernel, both user services and kernel services are kept in the same address space. | In microkernel user services and kernel, services are kept in separate address spaces. |
| Size | The monolithic kernel is larger than the microkernel. | The microkernel is smaller in size. |
| Execution | It has fast execution. | It has slow execution. |
| OS services | In a monolithic kernel system, the kernel contains the OS services. | In a microkernel-based system, the OS services and kernel are separated. |
| Extendible | The monolithic kernel is quite complicated to extend. | The microkernel is easily extendible. |
| Security | If a service crashes, then the whole system crashes in a monolithic kernel. | If a service crashed, it does not affect the working of the microkernel. |
| Customization | It is difficult to add new functionalities to the monolithic kernel. Therefore, it is not customizable. | It is easier to add new functionalities to the microkernel. Therefore, it is more customizable. |
| Code | Less coding is required to write a monolithic kernel. | A microkernel is required more coding. |
| Example | Linux, FreeBSD, OpenBSD, NetBSD, Microsoft Windows (95, 98, Me), Solaries, HP-UX, DOS, OpenVMS, XTS-400, etc. | QNX, Symbian, L4L.inux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotos. |