OPERATING SYSTEM

ASSIGNMENT

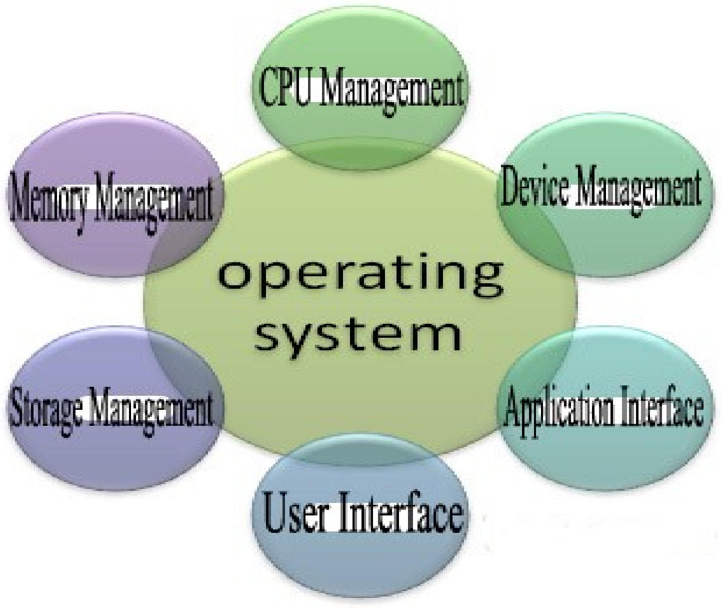
Submitted To-

Submitted By-

ER. CHAITALI MALIK

Q1. What are the basic function of an Operating System ?

Basic Function of Operating System -



Basic Function of Operating System

1) Device Management
2) Memory Management
3) CPU Management
4) Storage Management
5) Application Interface
6) User Interface

Device Management -

Device management controls peripheral devices by sending them commands in their own proprietary language.

Memory Management -

The memory management function keeps track of the status of each memory location, either allocated or free. It determines how memory is allocated among competing processes.

CPU Management -

The OS makes sure that as many processor cycles are used for work as possible. The OS schedules the processor's attention among the demands of different processes.

Storage Management -

OS keep track that each process must have enough memory in which to execute, and it can neither run into the memory space of another process nor be run into by another process. The different types of memory in the system must be used properly so that each process can run most effectively.

Application Interface -

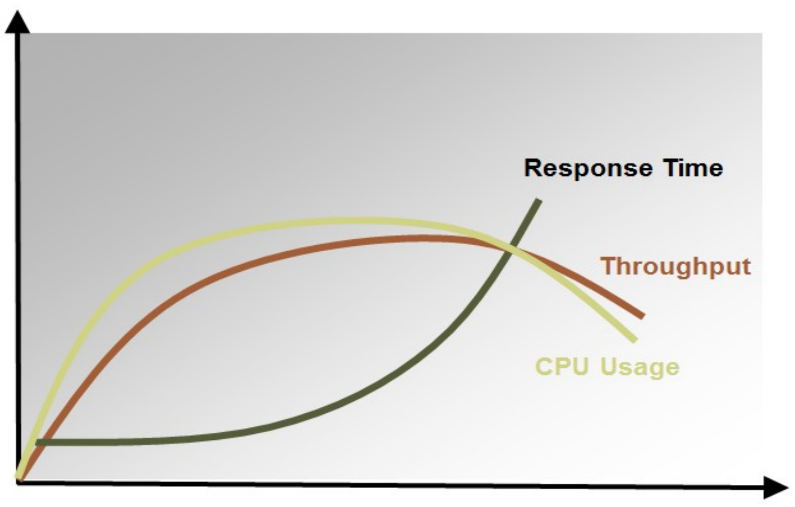
It allow application programmers to use functions of the computer and operating system without having the directly record of all the details in the CPU's operation.

User Interface -

The UI brings structure to the interaction between a user and the computer.

Q.2 Define the given below term :

1) Throughput 2) Turnaround Time 3) Waiting Time 4) Response Time



Throughput -

It is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour & for short transactions, throughput might be 10 processes per second.

Turnaround Time -

It is the interval from the time of submission to the time of completion of a process. it is the sum of the periods spent waiting to get in to memory , waiting in the ready queue, executing on the CPU & doing I/O.

Waiting Time -

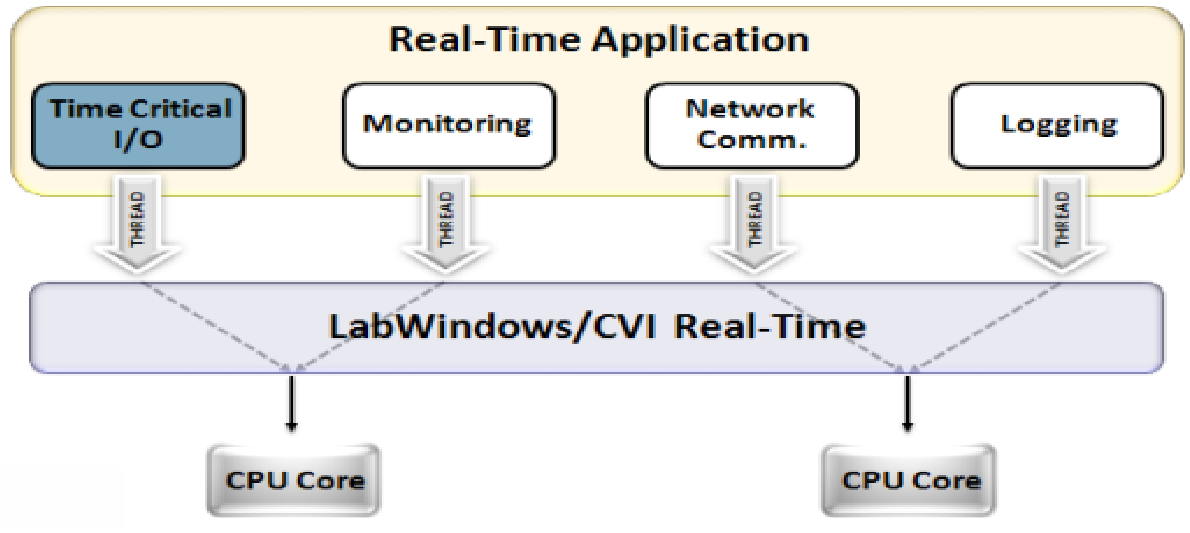
This is the amount of time for which a process has been waiting in the ready queue.

Response Time -

It is the amount of time taken from which a request was submitted until the first response is produced, not the output (for time-sharing environment).

Q3 .What are the differences between hard and soft real time system ?

Real time system -



Real time system

It is an operating system that guarantees a certain capability within a specified time constraint. In this Response Time is already fixed. Means time to Display the Results after Possessing has fixed by the Processor or CPU. Real Time System is used at those Places in which we Requires higher and Timely Response. These Types of Systems are used in Bomb and Reservation. So When we Specify the Request , the CPU will Perform at that Time.

Real-time operating systems are of two types :

1) Soft real-time systems.

2) Hard real-time systems.

Examples -The example of the real-time systems are as follows :

Food processing.

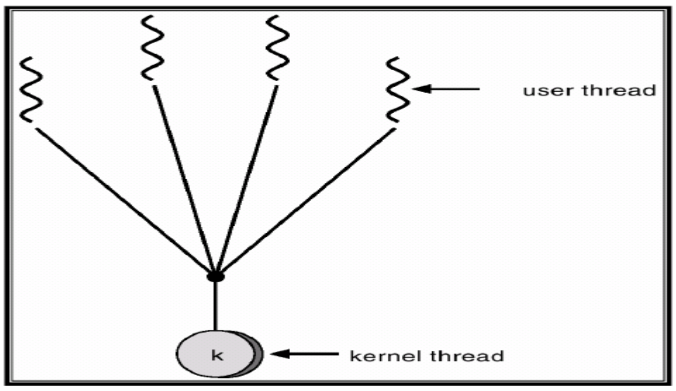
Engine Controls.

Anti-lock breaking systems.

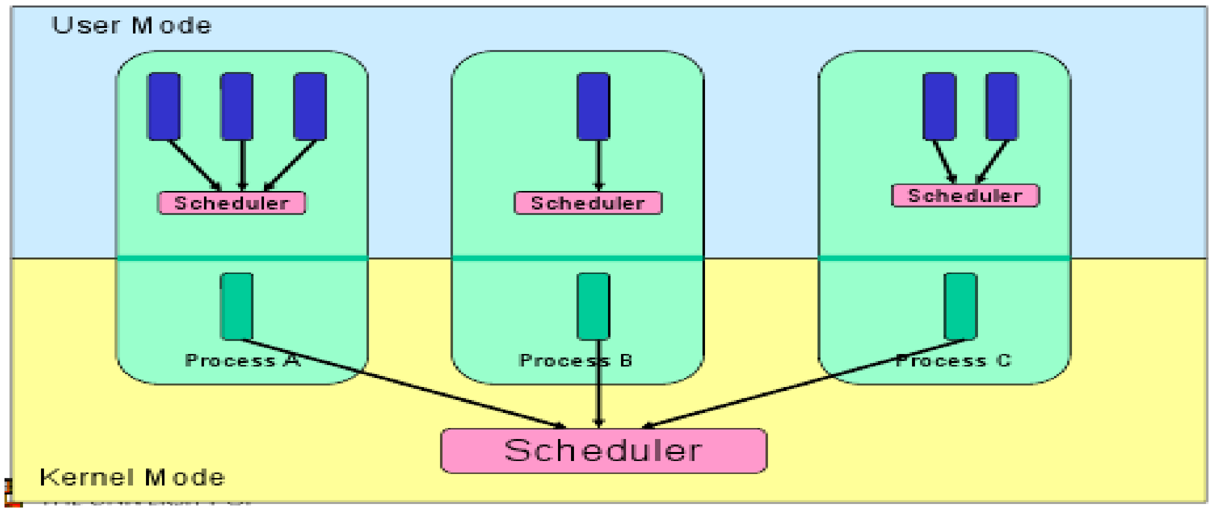
Hard real time system	**Soft real time system**
1) Hard Real Time Systems are the systems in which if the process is not completed in the given time the process is terminated and next process is started. 2) In a hard real-time operating system however, not only must tasks be completed within the specified timeframe, but they must also be completed correctly. 3) Critical task completion on time is guaranteed by a hard real time system. The tasks needed for delays in the system are to be bounded by retrieving the stored data at the time which takes the operating system to complete any request. 4) In this system, if the calculation could not be performed for making the object available at the designated time, the operating system would terminate with a failure. 5) The example of the hard real-time system is Pacemakers, Airplane control systems etc.	1) Soft Real Time Systems are the systems in which if the process is not completed in the given time then also the process continues to execute until it complete its operations. 2) In a soft real-time system, tasks are completed as fast as possible without having to be completed within a specified timeframe. 3) A critical task obtains a priority over other tasks and maintaining that priority until the completion of the task. This is performed by a soft real time system. The system kernel delays need to be bounded as in the case of hard real time system. 4) An operating system might be designed to ensure that a certain object was available for a robot on an assembly line. In a soft real-time operating system, the assembly line would continue to function but the production output might be lower as objects failed to appear at their designated time, causing the robot to be temporarily unproductive. 5) The example of Soft real-time system is Live Video Streaming.

Q4. What are the differences between User level thread and Kernel level thread ?

Thread -



A thread is a single sequence stream within a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes. In a process, threads allow multiple executions of streams. In many respect, threads are popular way to improve application through parallelism. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel. A thread has or consists of a program counter (PC), a register set, and a stack space.



User-level Threads	**Kernel-level Threads**
1) User-level threads implement in user-level libraries, rather than via systems calls, so thread switching does not need to call operating system and to cause interrupt to the kernel. 2) User-level threads does not require modification to operating systems. 3) Each thread is represented simply by a PC, registers, stack and a small control block, all stored in the user process address space.	1) A kernel thread, sometimes called a LWP (Lightweight Process) is created and scheduled by the kernel. Kernel threads are often more expensive to create than user threads and the system calls to directly create kernel threads are very platform specific. 2) The kernel knows about and manages the threads. No runtime system is needed in this case. Instead of thread table in each process, the kernel has a

4) Creating a thread, switching between threads and synchronization between threads can all be done without intervention of the kernel. 5) Thread switching is not much more expensive than a procedure call. 6) when using multiprocessor systems, user threads completely managed by the threading library can't be ran in parallel on the different CPUs, although this means they will run fine on uniprocessor systems.	thread table that keeps track of all threads in the system. 3) Scheduler may decide to give more time to a process having large number of threads than process having small number of threads. 4) Kernel-level threads are especially good for applications that frequently block. 5) All thread operations are implemented in the kernel and the OS schedules all threads in the system. 6) Kernel threads use the kernel scheduler, different kernel threads can run on different CPUs.

Q5. What are the various type of Scheduling Algorithms.

1) Priority Queue.

2) Shortest Job first.

3) Round Robbin.