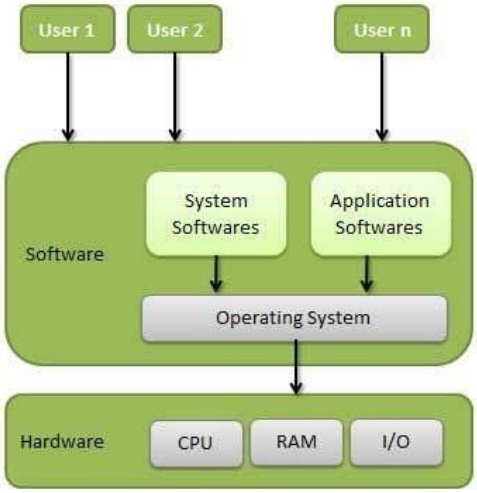
OPERATING SYSTEM NOTES

1. What is an Operating System?

Operating Systems: An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs. It is a layer of systems software that:

- directly has privileged access to the underlying hardware;
- hides the hardware complexity;
- Manages hardware on behalf of one or more application according to some predefined policies.
- In addition, it ensures that applications are isolated and protected from one another.



Types of Operating System (OS):

1. **Batch OS –** A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when the execution of the previous job completes.
2. **Multiprogramming OS –** The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it to the CPU. Whenever the executing process needs to wait for any other operation (like I/O), the OS selects another process from the job queue and assigns it to the CPU. This way, the CPU is never kept idle and the user gets the flavor of getting multiple tasks done at once.

3. **Multitasking OS –** Multitasking OS combines the benefits of Multiprogramming OS and CPU scheduling to perform quick switches between jobs. The switch is so quick that the user can interact with each program as it runs
4. **Time Sharing OS –** Time-sharing systems require interaction with the user to instruct the OS to perform various tasks. The OS responds with an output. The instructions are usually given through an input device like the keyboard.
5. **Real Time OS –** Real-Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines.

OS Elements

- **Abstractions** (corresponds to applications that OS executes)
    - process, thread, file, socket, memory page
- **Mechanisms** (on top of Abstractions)
    - create, schedule, open, write, allocate
- **Policies** (how mechanisms are used to manage underlying hardware)
    - Least Recently Used (LRU), Earliest Deadline First (EDF), etc.

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

2. What is the relationship between operating systems and computer hardware?

Answer: Operating system helps to make computer hardware available to the application programs. Without Operating System we cannot access computer hardware.

3. How Buffering can improve the performance of a Computer system?

Answer: If C.P.U and I/O devices are nearly same at speed, the buffering helps in making the C.P.U and the I/O devices work at full speed in such a way that C.P.U and the I/O devices never sit idle at any moment.

Normally the C.P.U is much faster than an input device. In this case the C.P.U always faces an empty input buffer and sits idle waiting for the input device which is to read a record into the buffer. For output, the C.P.U continues to work at full speed till the output buffer is full and then it starts waiting.

Thus buffering proves useful for those jobs that have a balance between computational work and I/O operations. In other cases, buffering scheme may not work well.

4. What are the primary differences between Network Operating System and Distributed Operating System?

S.No	Network Operating System	Distributed Operating System
1	A network operating system is made up of software and associated protocols that allow a set of computer network to be used together.	A distributed operating system is an ordinary centralized operating system but runs on multiple independent CPUs.
2	Environment users are aware of multiplicity of machines.	Environment users are not aware of multiplicity of machines.
3	Control over file placement is done manually by the user.	It can be done automatically by the system itself.
4	Performance is badly affected if certain part of the hardware starts malfunctioning.	It is more reliable or fault tolerant i.e distributed operating system performs even if certain part of the hardware starts malfunctioning.
5	Remote resources are accessed by either logging into the desired remote machine or transferring data from the remote machine to user's own machines.	Users access remote resources in the same manner as they access local resources.

5. What inconveniences that a user can face while interacting with a computer system, which is without an operating system?

Answer: Operating system is a required component of the computer system.
Without an operating system computer hardware is only an inactive electronic machine, which is inconvenient to user for execution of programs.
As the computer hardware or machine understands only the machine language. It is difficult to develop each and every program in machine language in order to execute it.
Thus without operating system execution of user program or to solve user problems is extremely difficult.

6. What are the differences between Batch processing system and Real Time Processing System?

Sr. No.	Batch Processing System	Real-time Processing System
1	Jobs with similar requirements are batched together and run through the computer as a group.	In this system, events mostly external to computer system are accepted and processed within certain deadlines.
2	This system is particularly suited for applications such as Payroll, Forecasting, Statistical analysis etc.	This processing system is particularly suited for applications such as scientific experiments, Flight control, few military applications, Industrial control etc.
3	It provides most economical and simplest processing method for business applications.	Complex and costly processing requires unique hardware and software to handle complex operating system programs.
4	In this system data is collected for defined period of time and is processed in batches.	Supports random data input at random time.
5	In this system sorting is performed before processing.	No sorting is required.

6	It is measurement oriented.	It is action or event oriented.
7	Transactions are batch processed and periodically.	Transactions are processed as and when they occur.
8	In this processing there is no time limit.	It has to handle a process within the specified time limit otherwise the system fails.

7. What are the differences between Real Time System and Timesharing System?

Sr. No.	**Real Time System**	**Timesharing System**
1	In this system, events mostly external to computer system are accepted and processed within certain deadlines.	In this system, many users are allowed to simultaneously share the computer resources.
2	Real time processing is mainly devoted to one application.	Time sharing processing deals with many different applications.
3	User can make inquiry only and cannot write or modify programs.	Users can write and modify programs.
4	User must get a response within the specified time limit; otherwise it may result in a disaster.	User should get a response within fractions of seconds but if not, the results are not disastrous.
5	No context switching takes place in this system.	The CPU switches from one process to another as a time slice expires or a process terminates.

8. What are the differences between multiprocessing and multiprogramming?

Sr. No.	Multiprocessing	Multiprogramming
1	Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.	Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2	It utilizes multiple CPUs.	It utilizes single CPU.
3	It permits parallel processing.	Context switching takes place.
4	Less time taken to process the jobs.	More Time taken to process the jobs.
5	It facilitates much efficient utilization of devices of the computer system.	Less efficient than multiprocessing.
6	Usually more expensive.	Such systems are less expensive.

9. What are the services of Operating system?

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system:

Program execution: Following are the major activities of an operating system with respect to program management:

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation: An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File system manipulation: Following are the major activities of an operating system with respect to file management:

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Error handling: Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling:

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management: In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management:

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Communication: Following are the major activities of an operating system with respect to communication:

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

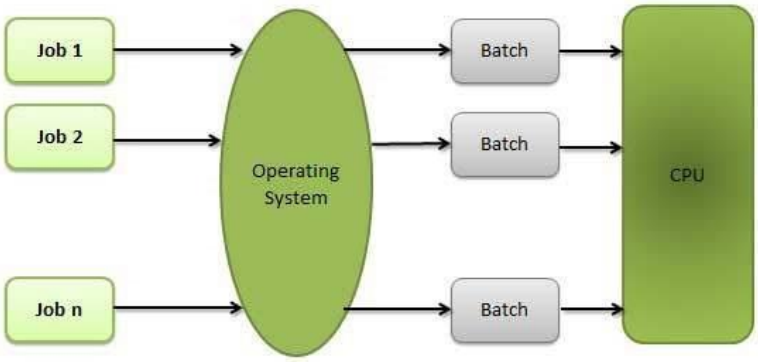
Protection: Following are the major activities of an operating system with respect to protection −

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

Batch processing

An operating system does the following activities related to batch processing −

- The OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- The OS keeps a number a jobs in memory and executes them without any manual information.
- Jobs are processed in the order of submission, i.e., first come first served fashion.
- When a job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.

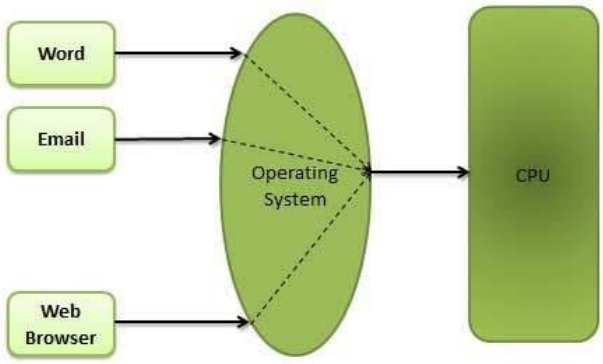


Multitasking

An OS does the following activities related to multitasking −

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- The OS handles multitasking in the way that it can handle multiple operations/executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.

- A time-shared operating system uses the concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- Each user has at least one separate program in memory.

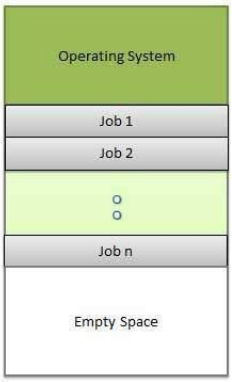


- A program that is loaded into memory and is executing is commonly referred to as a **process**.
- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.
- Since interactive I/O typically runs at slower speeds, it may take a long time to complete. During this time, a CPU can be utilized by another process.
- The operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Multiprogramming

An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensure that the CPU is never idle, unless there are no jobs to process



Interactivity

An Operating system does the following activities related to interactivity −

- Provides the user an interface to interact with the system.
- Manages input devices to take inputs from the user. For example, keyboard.
- Manages output devices to show outputs to the user. For example, Monitor.

Real Time System

Real-time systems are usually dedicated, embedded systems. An operating system does the following activities related to real-time system activity.

- In such systems, Operating Systems typically read from and react to sensor data.
- The Operating system must guarantee response to events within fixed periods of time to ensure correct performance.

Distributed Environment

An operating system does the following activities related to distributed environment −

- The OS distributes computation logics among several physical processors.
- The processors do not share memory or a clock. Instead, each processor has its own local memory.
- The OS manages the communications between the processors. They communicate with each other through various communication lines.

10. What are the advantages and disadvantages of batch processing?

Advantages

- Batch processing takes much of the work of the operator to the computer.
- Increased performance as a new job get started as soon as the previous job is finished, without any manual intervention.

Disadvantages

- Difficult to debug program.
- A job could enter an infinite loop.
- Due to lack of protection scheme, one batch job can affect pending jobs.

11. What are the advantages and disadvantages of multiprogramming?

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

12. What are the advantages of Multiprocessing or Parallel System?

Answer: Multiprocessing operating system or the parallel system support the use of more than one processor in close communication.

The advantages of the multiprocessing system are:

- **Increased Throughput** − by increasing the number of processors, more work can be completed in a unit time.
- **Cost Saving** − Parallel system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems. Also, if a number of programs are to operate on the same data, it is cheaper to store that data on one single disk and shared by all processors instead of using many copies of the same data.

- **Increased Reliability** − In this system, as the workload is distributed among several processors which results in increased reliability. If one processor fails then its failure may slightly slow down the speed of the system but system will work smoothly.

Threads

A thread is a lightweight process and forms the basic unit of CPU utilization. A process can perform more than one task at the same time by including multiple threads.

- A thread has its own program counter, register set, and stack
- A thread shares resources with other threads of the same process the code section, the data section, files and signals.

A new thread, or a child process of a given process, can be introduced by using the fork () system call. A process with n fork () system calls generates $2^n – 1$ child processes.

Advantages of Thread

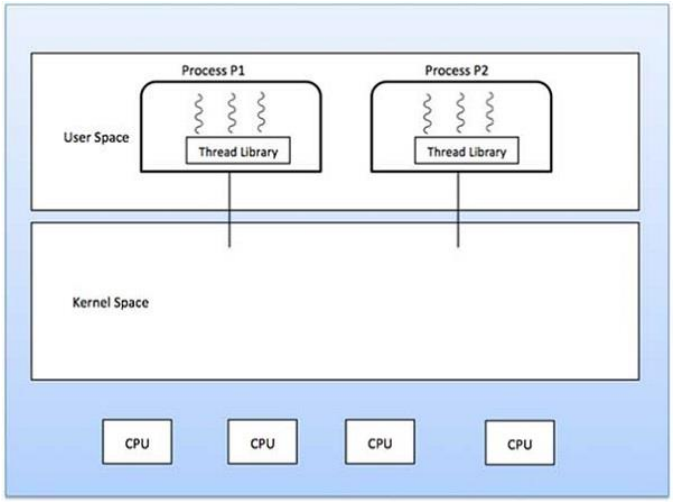
- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

There are two types of threads:

- **User Level Threads** − User managed threads.
- **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.

- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

Multithreading Models

Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

Many to many relationship.

Many to one relationship.

One to one relationship.

13. Differentiate between user thread and kernel thread.

User level thread	**Kernel level thread**
User threads are implemented by users.	Kernel threads are implemented by OS.
OS doesn't recognize user level threads.	Kernel threads are recognized by OS.
Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
Context switch time is less.	Context switch time is more.

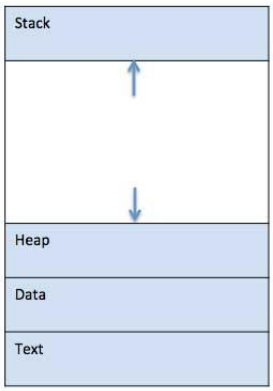
Context switch requires no hardware support.	Hardware support is needed.
If one user level thread performs blocking operation then entire process will be blocked.	If one kernel thread performs blocking operation then another thread can continue execution.

14. What is process and what are the main components of process?

Process

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections – stack, heap, text and data. The following image shows a simplified layout of a process inside main memory –



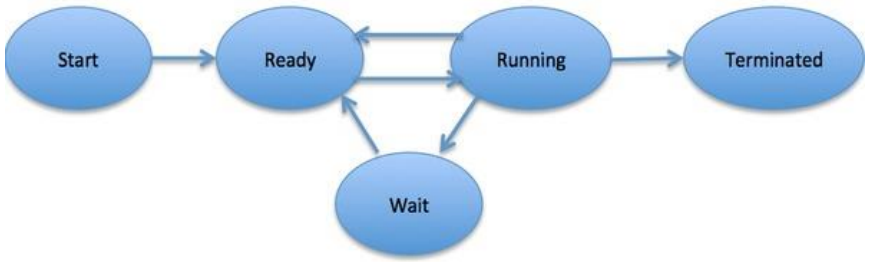
S.N.	Component & Description
1	**Stack** The process Stack contains the temporary data such as method/function parameters, return address and local variables.
2	**Heap** This is dynamically allocated memory to a process during its run time.

3	**Text** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
4	**Data** This section contains the global and static variables.

Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

In general, a process can have one of the following five states at a time.

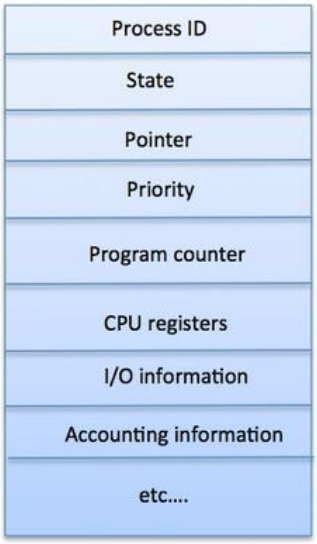


S.N.	State & Description
1	**Start** This is the initial state when a process is first started/created.
2	**Ready** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after **Start** state or while running it by but interrupted by the scheduler to assign CPU to some other process.
3	**Running** Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.

4	**Waiting** Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
5	**Terminated or Exit** Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process. Note: The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.



S.N.	Information & Description
1	**Process State** The current state of the process i.e., whether it is ready, running, waiting, or whatever.
2	**Process privileges** This is required to allow/disallow access to system resources.
3	**Process ID**

	Unique identification for each of the process in the operating system.
4	**Pointer** A pointer to parent process.
5	**Program Counter** Program Counter is a pointer to the address of the next instruction to be executed for this process.
6	**CPU registers** Various CPU registers where process need to be stored for execution for running state.
7	**CPU Scheduling Information** Process priority and other scheduling information which is required to schedule the process.
8	**Memory management information** This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
9	**Accounting information** This includes the amount of CPU used for process execution, time limits, execution ID etc.
10	**IO status information** This includes a list of I/O devices allocated to the process.

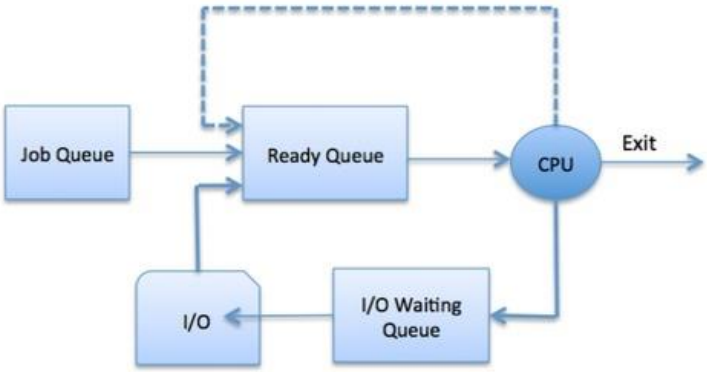
15. What is process scheduling?

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.

- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



Below are different times with respect to a process.

1. **Arrival Time –** Time at which the process arrives in the ready queue.
2. **Completion Time –** Time at which process completes its execution.
3. **Burst Time –** Time required by a process for CPU execution.
4. **Turn Around Time –** Time Difference between completion time and arrival time.

Turn Around Time = Completion Time - Arrival Time

5. **Waiting Time (WT) –** Time Difference between turnaround time and burst time.

Waiting Time = Turn Around Time - Burst Time

16. Why do we need scheduling?

A typical process involves both I/O time and CPU time. In a uni programming system like MS-DOS, time spent waiting for I/O is wasted and CPU is free during this time. In multiprogramming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

17. What are the objectives of Project Scheduling?

Objectives of Process Scheduling Algorithm:

- Max CPU utilization (Keep CPU as busy as possible)
- Fair allocation of CPU.
- Max throughput (Number of processes that complete their execution per time unit)

- Min turnaround time (Time taken by a process to finish execution)
- Min waiting time (Time for which a process waits in ready queue)
- Min response time (Time when a process produces first response)

18. What is a process scheduler? State the characteristics of a good process scheduler?

OR

What is scheduling? What criteria affect the scheduler's performance?

Answer: Scheduling can be defined as a set of policies and mechanisms which controls the order in which the work to be done is completed. The scheduling program which is a system software concerned with scheduling is called the scheduler and the algorithm it uses is called the scheduling algorithm.

Various criteria or characteristics that help in designing a good scheduling algorithm are:

CPU Utilization − A scheduling algorithm should be designed so that CPU remains busy as possible. It should make efficient use of CPU.

Throughput − Throughput is the amount of work completed in a unit of time. In other words throughput is the processes executed to number of jobs completed in a unit of time. The scheduling algorithm must look to maximize the number of jobs processed per time unit.

Response time − Response time is the time taken to start responding to the request. A scheduler must aim to minimize response time for interactive users.

Turnaround time − Turnaround time refers to the time between the moment of submission of a job/ process and the time of its completion. Thus how long it takes to execute a process is also an important factor.

Waiting time − It is the time a job waits for resource allocation when several jobs are competing in multiprogramming system. The aim is to minimize the waiting time.

Fairness − A good scheduler should make sure that each process gets its fair share of the CPU.

19. Differentiate between different types of schedulers.

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.

2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Scheduling algorithms

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

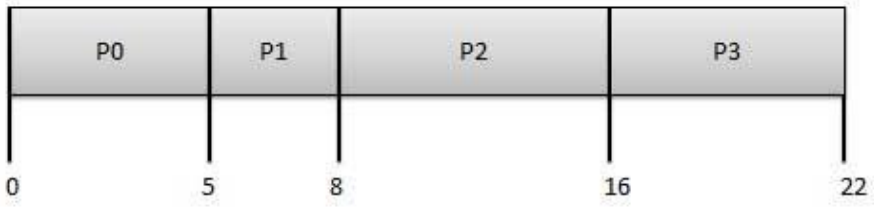
These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.

- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.

- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

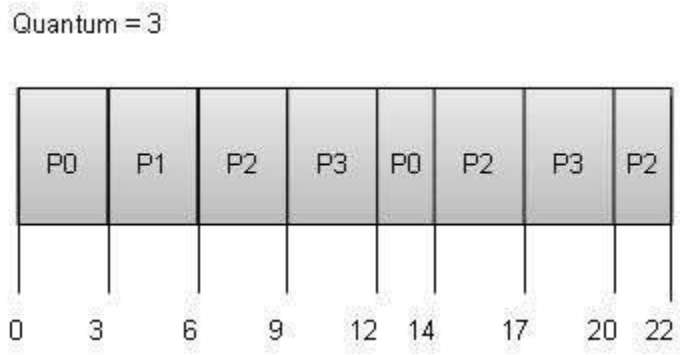
Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

20. Explain time slicing? How its duration affects the overall working of the system?

Answer: Time slicing is a scheduling mechanism/way used in time sharing systems. It is also termed as Round Robin scheduling. The aim of Round Robin scheduling or time slicing scheduling is to give all processes an equal opportunity to use CPU. In this type of scheduling, CPU time is divided into slices that are to be allocated to ready processes. Short processes may be executed within a single time quantum. Long processes may require several quanta.

The Duration of time slice or Quantum

The performance of time slicing policy is heavily dependent on the size/duration of the time quantum. When the time quantum is very large, the Round Robin policy becomes a FCFS policy. Too short quantum causes too many process/context switches and reduces CPU efficiency. So the choice of time quanta is a very important design decision. Switching from one process to another requires a certain amount of time to save and load registers, update various tables and lists etc.

Consider, as an example, process switch or context switch takes 5 m sec and time slice duration be 20 m sec. Thus CPU has to spend 5 m sec on process switching again and again wasting 20% of CPU time. Let the time slice size be set to say 500 m sec and 10 processes are in the ready queue. If P1 starts executing for first time slice then P2 will have to wait for 1/2 sec; and waiting time for other processes will increase. The unlucky last (P10) will have to wait for 5 sec, assuming that all others use their full time slices. To conclude setting the time slice.

- Too short will cause too many process switches and will lower CPU efficiency.
- Setting too long will cause poor response to short interactive processes.
- A quantum around 100 m sec is usually reasonable.

21. What are the different principles which must be considered while selection of a scheduling algorithm?

Answer: The objective/principle which should be kept in view while selecting a scheduling policy are the following −

1. **Fairness** − All processes should be treated the same. No process should suffer indefinite postponement.
2. **Maximize throughput** − Attain maximum throughput. The largest possible number of processes per unit time should be serviced.
3. **Predictability** − A given job should run in about the same predictable amount of time and at about the same cost irrespective of the load on the system.
4. **Maximum resource usage** − the system resources should be kept busy. Indefinite postponement should be avoided by enforcing priorities.
5. **Controlled Time** − There should be control over the different times −
    - Response time
    - Turnaround time
    - Waiting time

The objective should be to minimize above mentioned times.

Numerical

Question: Shown below is the workload for 5 jobs arriving at time zero in the order given below −

Job	Burst Time
1	10
2	29
3	3
4	7
4	12

Now find out which algorithm among FCFS, SJF and Round Robin with quantum 10, would give the minimum average time.

Answer: For FCFS, the jobs will be executed as:

Job	Waiting Time
1	0
2	10
3	39
4	42
5	49
	140

The average waiting time is 140/5=28.

For SJF (non-preemptive), the jobs will be executed as:

Job	Waiting Time
1	10
2	32
3	0
4	3
5	20
	65

The average waiting time is 65/5=13.

For Round Robin, the jobs will be executed as:

Job	Waiting Time
1	0
2	32
3	20
4	23
5	40
	115

The average waiting time is 115/5=23.

Thus SJF gives the minimum average waiting time.

Interrupt

Interrupts are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

1. **Hardware Interrupts** are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data (e.g., keystrokes on the keyboard or and data on the Ethernet card); or they have just completed a task which the operating system previous requested, such as transferring data between the hard drive and memory.
2. **Software Interrupts** are generated by programs when they want to request a system call to be performed by the operating system.
3. **Traps** are generated by the CPU itself to indicate that some error or condition occurred for which assistance from the operating system is needed.

Interrupts are important because they give the user better control over the computer. Without interrupts, a user may have to wait for a given application to have a higher priority over the CPU to be ran. This ensures that the CPU will deal with the process immediately.

22. How does CPU respond to Interrupts?

The CPU performs the following actions in response to an interrupt:

1. Using the pointer to the current process control block, the state and all register values for the process are saved for use when the process is later restarted.
2. The CPU mode bit is switched to *supervisory* mode.
3. Using the pointer to the interrupt handler table and the interrupt vector, the location of the kernel code to execute is determined. The interrupt vector is the IRQ for hardware interrupts (read from an interrupt controller register) and an argument to the interrupt assembly language instruction for software interrupts.
4. Processing is switched to the appropriate portion of the kernel.

23. What are the CPU execution mode?

There are two modes of execution, known as user mode and kernel or supervisor mode. User mode is restricted in that certain instructions cannot be executed, certain registers cannot be accessed, and I/O devices cannot be accessed. Kernel mode has none of these restrictions.
A system call will set the CPU to kernel mode, as will traps and interrupts. Application programs cannot do this.

24. What are the different types of Interrupt?

Supervisor calls or SVC interrupts

These interrupts occur when the program issues an SVC to request a particular system service. An SVC interrupts the program being executed and passes control to the supervisor so that it can perform the service. Programs request these services through macros such as OPEN (open a file), GETMAIN (obtain storage), or WTO (write a message to the system operator).

I/O interrupts

These interrupts occur when the channel subsystem signals a change of status, such as an input/output (I/O) operation completing, an error occurring, or an I/O device such as a printer has become ready for work.

External interrupts

These interrupts can indicate any of several events, such as a time interval expiring, the operator pressing the interrupt key on the console, or the processor receiving a signal from another processor.

Restart interrupts

These interrupts occur when the operator selects the restart function at the console or when a restart SIGP (signal processor) instruction is received from another processor.

Program interrupts

These interrupts are caused by program errors (for example, the program attempts to perform an invalid operation), page faults (the program references a page that is not in central storage), or requests to monitor an event.

Machine check interrupts

These interrupts are caused by machine malfunctions.

When an interrupt occurs, the hardware saves pertinent information about the program that was interrupted and, if possible, disables the processor for further interrupts of the same type. The hardware then routes control to the appropriate interrupt handler routine. The program status word or PSW is a key resource in this process.

25. What is meant by PSW?

The program status word (PSW) is a 128-bit data area in the processor that, along with a variety of other types of registers (control registers, timing registers, and prefix registers) provides details crucial to both the hardware and the software. The current PSW includes the address of the next program instruction and control information about the program that is running. Each processor has only one current PSW. Thus, only one task can execute on a processor at a time.

The PSW controls the order in which instructions are fed to the processor, and indicates the status of the system in relation to the currently running program. Although each processor has only one PSW, it is useful to think of three types of PSWs to understand interrupt processing:

- Current PSW
- New PSW
- Old PSW

The current PSW indicates the next instruction to be executed. It also indicates whether the processor is enabled or disabled for I/O interrupts, external interrupts, machine check interrupts, and certain program interrupts. When the processor is enabled, these interrupts can occur. When the processor is disabled, these interrupts are ignored or remain pending.

There is a new PSW and an old PSW associated with each of the six types of interrupts. The new PSW contains the address of the routine that can process its associated interrupt. If the processor is enabled for interrupts when an interrupt occurs, PSWs are switched using the following technique:

1. Storing the current PSW in the old PSW associated with the type of interrupt that occurred.
2. Loading the contents of the new PSW for the type of interrupt that occurred into the current PSW.

Other Registers

Mainframes provide other registers, as follows:

Access registers

These registers specify the address space in which data is found.

General registers

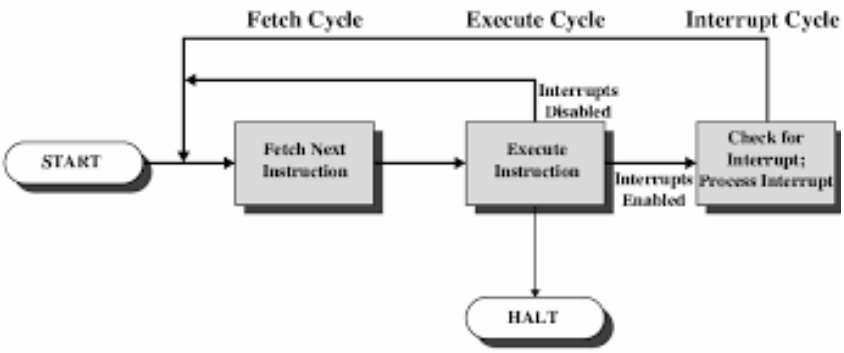
These registers address data in storage, and also hold user data.

Floating point registers

These registers hold numeric data in floating point form.

Control registers

These registers are used by the operating system itself, for example, as references to translation tables.



26. Differentiate between Software and Hardware Interrupt.

SR.NO.	Hardware Interrupt	Software Interrupt
1	Hardware interrupt is an interrupt generated from an external device or hardware.	Software interrupt is the interrupt that is generated by any internal system of the computer.
2	It do not increment the program counter.	It increment the program counter.
3	Hardware interrupt can be invoked with some external device such as request to start an I/O or occurrence of a hardware failure.	Software interrupt can be invoked with the help of INT instruction.
4	It has lowest priority than software interrupts	It has highest priority among all interrupts.
5	Hardware interrupt is triggered by external hardware and is considered one of the ways to communicate with the outside peripherals, hardware.	Software interrupt is triggered by software and considered one of the ways to communicate with kernel or to trigger system calls, especially during error or exception handling.
6	It is an asynchronous event.	It is synchronous event.

SR.NO.	Hardware Interrupt	Software Interrupt
7	Hardware interrupts can be classified into two types they are: 1. Maskable Interrupt. 2. Non Maskable Interrupt.	Software interrupts can be classified into two types they are: 1. Normal Interrupts. 2. Exception
8	Keystroke depressions and mouse movements are examples of hardware interrupt.	All system calls are examples of software interrupts

27. What is the difference between Job and Process?

Answer: A process refers to a program under execution. This program may be an application or system program.

Job means an application program and it is not a system program.

28. Differentiate between Process and Threads.

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.

5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Computer Architecture | Flynn's taxonomy

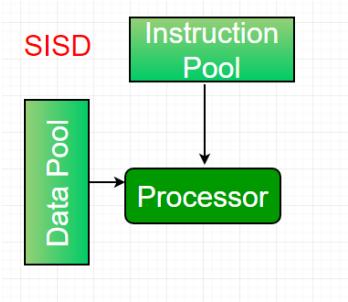
There are different system and memory architecture styles that need to be considered while designing the program or concurrent system. It is very necessary because one system & memory style may be suitable for one task but may be error prone to other task.

Michael Flynn in 1972 gave taxonomy for categorizing different styles of computer system architecture. This taxonomy defines four different styles as follows −

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data stream (SIMD)
- Multiple instruction stream, single data stream (MISD)
- Multiple instruction stream, multiple data stream (MIMD).

Single instruction stream, single data stream (SISD)

As the name suggests, such kind of systems would have one sequential incoming data stream and one single processing unit to execute the data stream. They are just like uniprocessor systems having parallel computing architecture. Following is the architecture of SISD



Advantages of SISD

The advantages of SISD architecture are as follows −

- It requires less power.

- There is no issue of complex communication protocol between multiple cores.

Disadvantages of SISD

The disadvantages of SISD architecture are as follows −

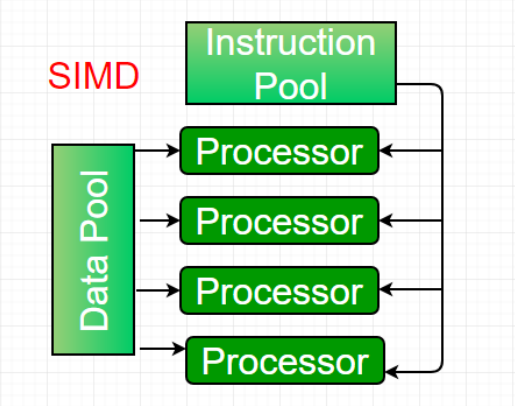
- The speed of SISD architecture is limited just like single-core processors.
- It is not suitable for larger applications.

Example

Dominant representative SISD systems are IBM PC, workstations.

Single instruction stream, multiple data stream (SIMD)

As the name suggests, such kind of systems would have multiple incoming data streams and number of processing units that can act on a single instruction at any given time. They are just like multiprocessor systems having parallel computing architecture. Following is the architecture of SIMD



Example

The best example for SIMD is the graphics cards. These cards have hundreds of individual processing units. If we talk about computational difference between SISD and SIMD then for the adding arrays **[5, 15, 20]** and **[15, 25, 10],** SISD architecture would have to perform three different add operations. On the other hand, with the SIMD architecture, we can add then in a single add operation.

Advantages of SIMD

The advantages of SIMD architecture are as follows −

- Same operation on multiple elements can be performed using one instruction only.
- Throughput of the system can be increased by increasing the number of cores of the processor.

- Processing speed is higher than SISD architecture.

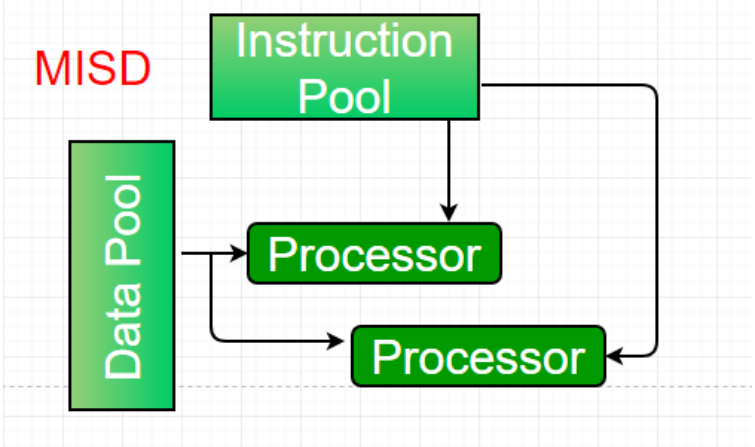
Disadvantages of SIMD

The disadvantages of SIMD architecture are as follows −

- There is complex communication between numbers of cores of processor.
- The cost is higher than SISD architecture.

Multiple Instruction Single Data (MISD) stream

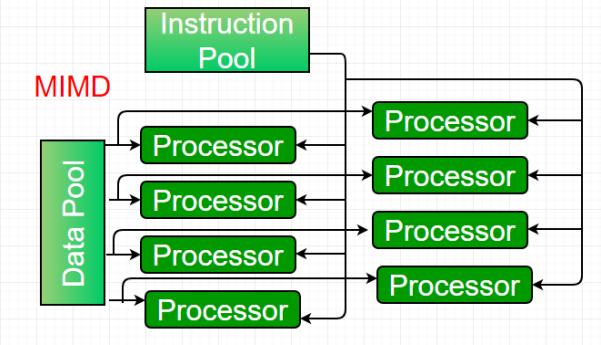
Systems with MISD stream have number of processing units performing different operations by executing different instructions on the same data set. Following is the architecture of MISD −



The representatives of MISD architecture do not yet exist commercially.

Multiple Instruction Multiple Data (MIMD) stream

An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously.



MIMD machines are broadly categorized into **shared-memory MIMD** and **distributed-memory MIMD** based on the way PEs are coupled to the main memory.

In the **shared memory MIMD** model (tightly coupled multiprocessor systems), all the PEs are connected to a single global memory and they all have access to it. The communication between PEs in this model takes place through the shared memory, modification of the data stored in the global memory by one PE is visible to all other PEs. Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

In **Distributed memory MIMD** machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter process communication channel, or IPC). The network connecting PEs can be configured to tree, mesh or in accordance with the requirement.
The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model. Failures in a shared-memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated. Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention. This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory. As a result of practical outcomes and user's requirement, distributed memory MIMD architecture is superior to the other existing models.

Process Switching

When a program runs then it may contain various threads run accordingly. In single threaded processes, the thread itself is the process. While in multithreaded processes we need to switch between different threads for the execution of our program.

1. Thread Switching **:**
Thread switching is a type of context switching from one thread to another thread in the same process. Thread switching is very efficient and much cheaper because it involves switching out only identities and resources such as the program counter, registers and stack pointers. The cost of thread-to-thread switching is about the same as the cost of entering and exiting the kernel.

2. Process Switching **:**
Process switching is a type of context switching where we switch one process with another process. It involves switching of all the process resources with those needed by a new process. This means switching the memory address space. This includes memory addresses, page tables, and kernel resources, caches in the processor.

29. Differentiate between Thread Context Switch and Process Context Switch :

No.	Thread Context Switch	Process Context Switch
1.	TCS occurs when the CPU saves the current state of the thread and switches to another thread of the same process.	PCS occurs when the operating system's scheduler saves the current state of the running Program (including the state of PCB) and switches to another program.
2.	TCS helps the CPU to handle multiple threads simultaneously.	PCS involves loading of the states of the new program for it's execution.
3.	TCS does not involves switching of memory address spaces.All the memory addresses that the processor accounts remain saved.	PCS involves switching of memory address spaces.All the memory addresses that the processor accounts gets flushed.
4.	Processor's cache and Translational Lookaside Buffer preserves their state.	Processor's cache and TLB gets flushed.
5.	Though TCS involves switching of registers and stack pointers, it does not afford the cost of changing the address space.Hence it is more efficient.	PCS involves the heavy cost of changing the address space.Hence it is less efficient.
6.	TCS is a bit faster and cheaper.	PCS is relatively slower and costlier.

30. What are the characteristics of Modern Operating System?

The **Characteristics of Modern OS** include:

1. Object-Oriented Design
2. Multi-threading
3. Systematic Multi-Processing
4. Distributed Operating System
5. Micro-Kernel Architecture