Students must refer GALVIN OS Book.

(2) (Copied From websites)

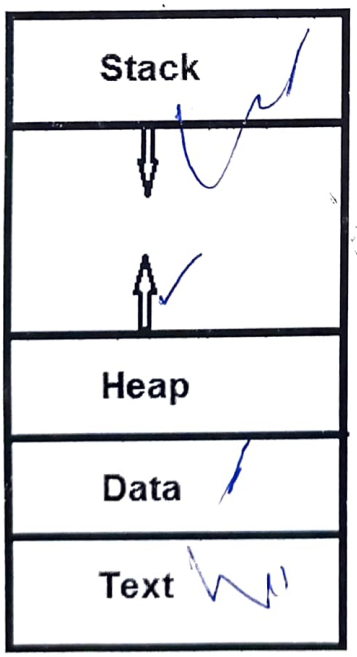
Introduction of Process Management

Program vs Process: A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.
A process is an 'active' entity instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

What does a process look like in memory?

Program vs Process: A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A process is an 'active' entity instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created)

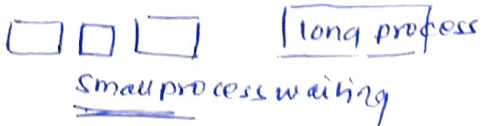


Homework

① what is kernel
② what is monolithic & micro kernel Architecture
③ what are differences among schedular.

engage

→ The Convoy Effect is a phenomenon in which entire operating system slow down owing to a few slower proces in the system. when CPU time is allotted to a process the FCFS algorithm assumes that other processes only get CPU time when the current one is finishe



Text Section: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the **Program Counter**.
Stack: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
Data Section: Contains the global variable.
Heap Section: Dynamically allocated memory to process during its run time.

Attributes of a Process: A process has the following attributes.

1. **Process Id**: A unique identifier assigned by the operating system

2. **Process State**: Can be ready, running, etc.

3. **CPU registers**: Like the Program Counter (CPU registers must be saved and restored when a process is swapped in and out of CPU)

4. **Accounts information**: Amount of CPU used for process execution, time limits, execution ID etc.

5. **I/O status information**: For example, devices allocated to the process, open files, etc.

6. **CPU scheduling information**: For example, Priority (Different processes may have different priorities, for example a shorter process assigned high priority in the shortest job first scheduling)

All of the above attributes of a process are also known as the ***context of the process***. Every process has its own process control block(PCB), i.e each process will have a unique PCB. All of the above attributes are part of the PCB.

States of Process: A process is in one of the following states:

1. **New**: Newly Created Process (or) being-created process.

2. **Ready**: After creation process moves to Ready state, i.e. the process is ready for execution.

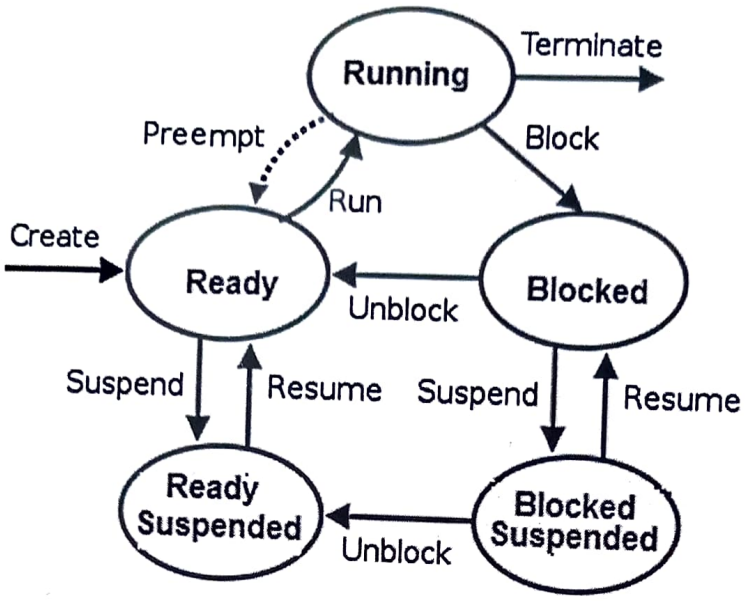
3. **Run**: Currently running process in CPU (only one process at a time can be under execution in a single processor).

4. **Wait** (or Block): When a process requests I/O access.

5. **Complete** (or terminated): The process completed its execution.

6. **Suspended Ready**: When the ready queue becomes full, some processes are moved to suspended ready state

7. **Suspended Block**: When waiting queue becomes full.

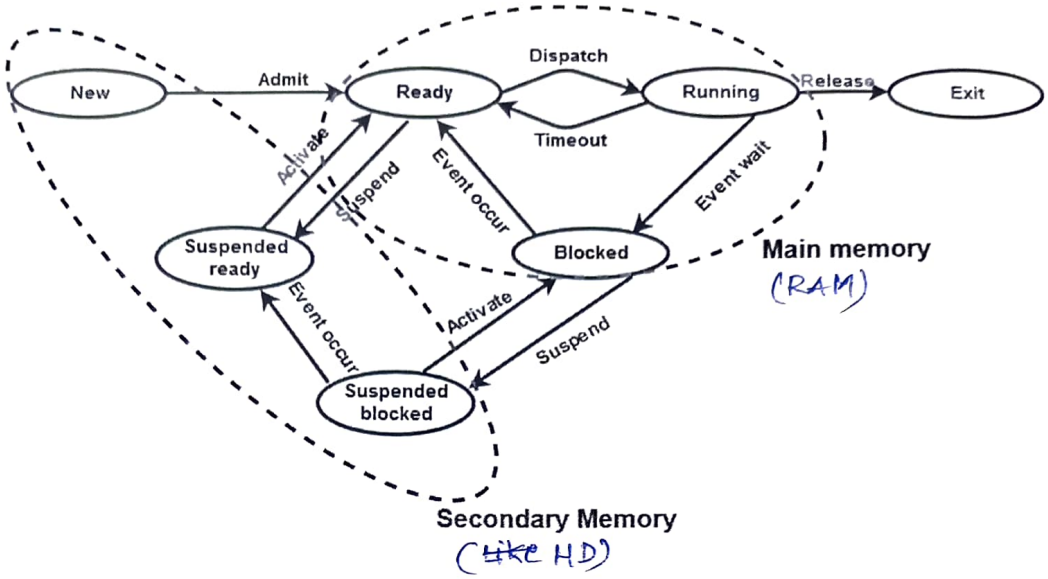


Context Switching: The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from the running state to the ready state.

When does context switching happen?

1. When a high-priority process comes to a ready state (i.e. with higher priority than the running process)
2. An Interrupt occurs ← ① software ② Hardware
3. User and kernel-mode switch (It is not necessary though)
4. Pre-emptive CPU scheduling used.

States of a Process in Operating Systems



- **New (Create)** – In this step, the process is about to be created but not yet created, it is the program which is present in secondary memory that will be picked up by OS to create the process.
- **Ready** – New -> Ready to run. After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.
- **Run** – The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- **Blocked or wait** – Whenever the process requests access to I/O or needs input from the user or needs access to a critical region(the lock for which is already acquired) it enters the blocked or wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
- **Terminated or completed** – Process is killed as well as PCB is deleted.
- **Suspend ready** – Process that was initially in the ready state but was swapped out of main memory(refer Virtual Memory topic) and placed onto external storage by scheduler is said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.
- **Suspend wait or suspend blocked** – Similar to suspend ready but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished it may go to suspend ready.

CPU and I/O Bound Processes: If the process is intensive in terms of CPU operations then it is called CPU bound process. Similarly, If the process is intensive in terms of I/O operations then it is called I/O bound process.

Types of schedulers: → Also known as Job schedular

1. **Long term – performance** – Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming. Once a decision is taken it lasts for a long time hence called long term scheduler.
2. **Short term – Context switching time** – Short term scheduler will decide which process to be executed next and then it will call dispatcher. A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.
3. **Medium term – Swapping time** – Suspension decision is taken by medium term scheduler. Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

Multiprogramming – We have many processes ready to run. There are two types of multiprogramming:

1. **Pre-emption** – Process is forcefully removed from CPU. Pre-emption is also called as time sharing or multitasking.
2. **Non pre-emption** – Processes are not removed until they complete the execution.

→ decided by Long term S.

Degree of multiprogramming – The number of processes that can reside in the ready state at maximum decides the degree of multiprogramming, e.g., if the degree of programming = 100, this means 100 processes can reside in the ready state at maximum.

Process Schedulers in Operating System

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.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

There are three types of process scheduler.

1. **Long Term or job scheduler :**
   It brings the new process to the 'Ready State'. It controls ***Degree of Multi-programming***, i.e., number of process present in ready state at any point of time. It is important that the long-term scheduler make a careful selection of both I/O and CPU-bound processes. I/O bound tasks are which use much of their time in input and output operations while CPU bound processes are which spend their time on CPU. The job scheduler increases efficiency by maintaining a balance between the two.

2. **Short term or CPU scheduler :**
   It is responsible for selecting one process from ready state for scheduling it on the running state. Note: Short-term scheduler only selects the process to schedule it doesn't load the process on running. Here is when all the scheduling algorithms are used. The CPU scheduler is responsible for ensuring there is no starvation owing to high burst time processes. ***Dispatcher*** is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:
   1. Switching context.
   2. Switching to user mode.
   3. Jumping to the proper location in the newly loaded program.
3. **Medium-term scheduler :**
   It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. It is helpful in maintaining a perfect balance between the I/O bound and the CPU bound. It reduces the degree of multiprogramming.

Process Table and Process Control Block (PCB)

While creating a process the operating system performs several operations. To identify the processes, it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process's execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc. All these information is required and must be saved when the process is switched from one state to another. When the process makes a transition from one state to another, the operating system must update information in the process's PCB.

A process control block (PCB) contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCB's, that

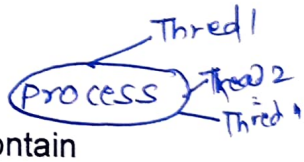
means logically contains a PCB for all of the current processes in the system.

- **Pointer** – It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.
- **Process state** – It stores the respective state of the process.
- **Process number** – Every process is assigned with a unique id known as process ID or PID which stores the process identifier.
- **Program counter** – It stores the counter which contains the address of the next instruction that is to be executed for the process.
- **Register** – These are the CPU registers which includes: accumulator, base, registers and general purpose registers.
- **Memory limits** – This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.
- **Open files list** – This information includes the list of files opened for a process.

Interrupts

The interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices one of the bus control lines is dedicated for this purpose and is called the *Interrupt Service Routine (ISR)*.

Thread in Operating System



What is a Thread?

A thread is a path of execution within a process. A process can contain multiple threads.

Why Multithreading?

A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces. Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Difference between Process and Thread

Process: Processes are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution. PCB(Process Control Block) holds the concept of process. A process can create other processes which are known as Child Processes. The process takes more time to terminate and it is isolated means it does not share the memory with any other process.
The process can have the following states new, ready, running, waiting, terminated, and suspended.
Thread: Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process. A thread has three states: Running, Ready, and Blocked.

Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.

S.NO	Process	Thread
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	The process is isolated.	Threads share memory.
8.	The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
9.	Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
10.	If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.
11.	The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
12.	Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behaviour of the other threads of the process.
13.	A system call is involved in it.	No system call is involved, it is created using APIs.

S.NO	Process	Thread
14.	The process does not share data with each other.	Threads share data with each other.

CPU Scheduling in Operating Systems

Scheduling of processes/work is done to finish the work on time. **CPU Scheduling** is a process that allows one process to use the CPU while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU. The purpose of CPU Scheduling is to make the system more efficient, faster, and fairer.

Why do we need to schedule processes?

- **Scheduling** is important in many different computer environments. One of the most important areas is scheduling which programs will work on the CPU. This task is handled by the Operating System (OS) of the computer and there are many different ways in which we can choose to configure programs.
- **Process Scheduling** allows the OS to allocate CPU time for each process. Another important reason to use a process scheduling system is that it keeps the CPU busy at all times. This allows you to get less response time for programs.
- Considering that there may be hundreds of programs that need to work, the OS must launch the program, stop it, switch to another program, etc. The way the OS configures the system to run another in the CPU is called "context switching". If the OS keeps context-switching programs in and out of the provided CPUs, it can give the user a tricky idea that he or she can run any programs he or she wants to run, all at once.
- So now that we know we can run 1 program at a given CPU, and we know we can change the operating system and remove another one using the context switch, how do we choose which programs we need. run, and with what program?
- That's where **scheduling** comes in! First, you determine the metrics, saying something like "the amount of time until the end". We will define this metric as "the time interval between which a function enters the system until it is completed". Second, you decide on a metrics that reduces metrics. We want our tasks to end as soon as possible.

What is the need for CPU scheduling algorithm?

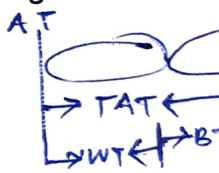
CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

In Multiprogramming, if the long-term scheduler selects multiple I / O binding processes then most of the time, the CPU remains an idle. The function of an effective program is to improve resource utilization.

Objectives of Process Scheduling Algorithm:

- Utilization of CPU at maximum level. **Keep CPU as busy as possible**.
- **Allocation of CPU should be fair**.
- **Throughput should be Maximum**. i.e. Number of processes that complete their execution per time unit should be maximized.
- **Minimum turnaround time**, i.e. time taken by a process to finish execution should be the least.
- There should be a **minimum waiting time** and the process should not starve in the ready queue.
- **Minimum response time.** It means that the time when a process produces the first response should be as less as possible.

What are the different terminologies to take care of in any CPU Scheduling algorithm?



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

Turn Around Time = Completion Time – Arrival Time { TAT = CT − AT)

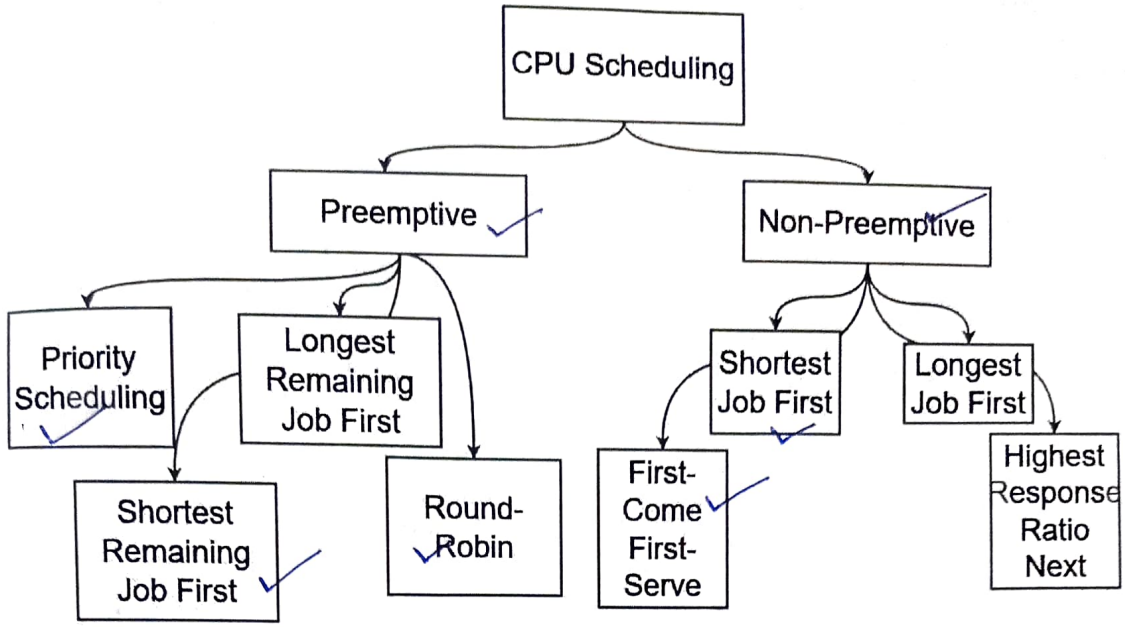
- **Waiting Time(W.T):** Time Difference between turn around time and burst time.

Waiting Time = Turn Around Time – Burst Time { TAT = WT + BT

* If arrival time is zero TAT equals to CT for the process

What are the different types of CPU Scheduling Algorithms?
There are mainly two types of scheduling methods:

- Preemptive Scheduling: Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
- Non-Preemptive Scheduling: Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.



Let us now learn about these CPU scheduling algorithms in operating systems one by one:

1. First Come First Serve:

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- FCFS supports non-pre-emptive and pre-emptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement
- First come, first serve method

Disadvantages of FCFS:

- FCFS suffers from **Convoy effect**.
- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not much efficient.

2. Shortest Job First(SJF):

Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling

method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

Characteristics of SJF:

- Shortest Job first has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of Shortest Job first:

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- SJF is generally used for long term scheduling

Disadvantages of SJF:

- One of the demerit SJF has is starvation.
- Many times it becomes complicated to predict the length of the upcoming CPU request

3. Longest Job First(LJF):

Longest Job First(LJF) scheduling process is just opposite of shortest job first (SJF), as the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. Longest Job First is non-preemptive in nature.

Characteristics of LJF:

- Among all the processes waiting in a waiting queue, CPU is always assigned to the process having largest burst time.
- If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
- LJF CPU Scheduling can be of both preemptive and non-preemptive types.

Advantages of LJF:

- No other task can schedule until the longest job or process executes completely.
- All the jobs or processes finish at the same time approximately.

Disadvantages of LJF:

- Generally, the LJF algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
- This may lead to convoy effect.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the Longest job first scheduling.

4. Priority Scheduling:

Preemptive Priority CPU Scheduling Algorithm is a pre-emptive method of CPU scheduling algorithm that works **based on the priority** of a process. In this algorithm, the editor sets the functions to be as important, meaning that the most important process must be done first. In the case of any conflict, that is, where there are more than one processor with equal value, then the most important CPU planning algorithm works on the basis of the FCFS (First Come First Serve) algorithm.

Characteristics of Priority Scheduling:

- Schedules tasks based on priority.
- When the higher priority work arrives while a task with less priority is executed, the higher priority work takes the place of the less priority one and
- The latter is suspended until the execution is complete.
- Lower is the number assigned, higher is the priority level of a process.

Advantages of Priority Scheduling:

- The average waiting time is less than FCFS
- Less complex

Disadvantages of Priority Scheduling:

- One of the most common demerits of the Preemptive priority CPU scheduling algorithm is the Starvation Problem. This is the problem in which a process has to wait for a longer amount of time to get scheduled into the CPU. This condition is called the starvation problem.

5. Round robin:

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Characteristics of Round robin:

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round robin:

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the Round robin Scheduling algorithm.

6. Shortest Remaining Time First:

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

- SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given it's overhead charges are not counted.
- The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.

Advantages of SRTF:

- In SRTF the short processes are handled very fast.
- The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

Disadvantages of SRTF:

- Like the shortest job first, it also has the potential for process starvation.
- Long processes may be held off indefinitely if short processes are continually added.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the shortest remaining time first.

7. Longest Remaining Time First:

The longest remaining time first is a preemptive version of the longest job first scheduling algorithm. This scheduling algorithm is used by the operating system to program incoming processes for use in a systematic way. This algorithm schedules those processes first which have the longest processing time remaining for completion.

Characteristics of longest remaining time first:

- Among all the processes waiting in a waiting queue, the CPU is always assigned to the process having the largest burst time.
- If two processes have the same burst time then the tie is broken using FCFS i.e. the process that arrived first is processed first.
- LJF CPU Scheduling can be of both preemptive and non-preemptive types.

Advantages of LRTF:

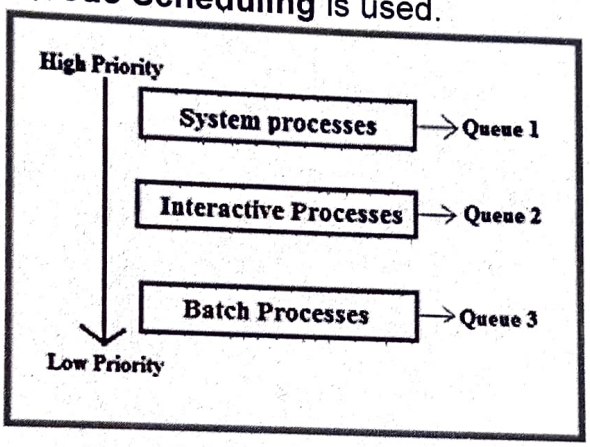
- No other process can execute until the longest task executes completely.
- All the jobs or processes finish at the same time approximately.

Disadvantages of LRTF:

- This algorithm gives a very high average waiting time and average turn-around time for a given set of processes.
- This may lead to a convoy effect.

9. Multiple Queue Scheduling:

Processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation **Multilevel Queue Scheduling** is used.



The description of the processes in the above diagram is as follows:

- **System Processes:** The CPU itself has its process to run, generally termed as System Process.
- **Interactive Processes:** An Interactive Process is a type of process in which there should be the same type of interaction.
- **Batch Processes:** Batch processing is generally a technique in the Operating system that collects the programs and data together in the form of a **batch** before the **processing** starts.

Advantages of multilevel queue scheduling:

- The main merit of the multilevel queue is that it has a low scheduling overhead.

Disadvantages of multilevel queue scheduling:

- Starvation problem
- It is inflexible in nature

0. Multilevel Feedback Queue Scheduling::

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like **Multilevel Queue Scheduling** but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel feedback queue scheduling:

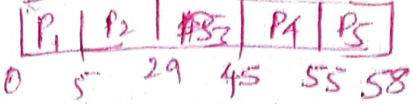
- It is more flexible
- It allows different processes to move between different queues

Disadvantages of Multilevel feedback queue scheduling:

- It also produces CPU overheads
- It is the most complex algorithm.

① FCFS

	AT	BT	CT	TAT	WT
P1	0	5	5	5	0
P2	2	40	29	29	5
P3	0	16	45	45	29
P4	0	10	55	55	45
P5	0	3	58	58	55



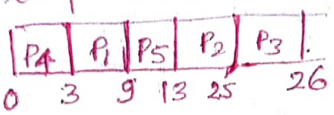
Avg(WT) = ~~39.4ms~~ 26.8ms
Avg(TAT) = 39.4ms

pre-emptive
↳ P_1 Jump P_2
context switching
↳ PCB is context

Sudekula

② Priority

	BT	Priority	CT	TAT	WT
P1	6	2	9	9	3
P2	12	4	25	25	13
P3	1	5	26	26	25
P4	3	1	3	3	0
P5	4	3	13	13	9



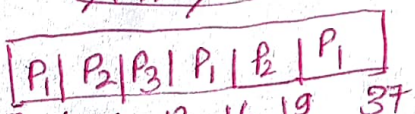
Avg(TAT) = 15.2ms
Avg(WT) = 10ms

③ RR

Process	BT	CT	TAT	WT
P1	26	37	37	11
P2	7	19	19	12
P3	4	12	12	8

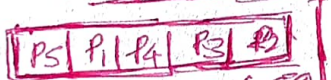
QT = 4

Avg(WT) = 11+12+8/3 = 31/3 ≈ 10.33



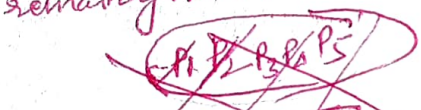
④ Shortest Job First (SJF) | Preemptive version shortest remaining time first (SRTF)

	BT	CT	TAT
P1	3	8	8
P2	24	58	58
P3	16	34	34
P4	10	18	18
P5	3	3	3

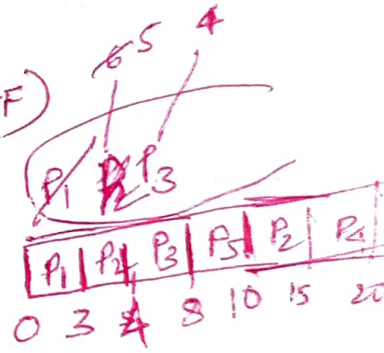


Avg(TAT) = 24.2

	BT	AT
P1	3	0
P2	6	2
P3	4	4
P4	5	6
P5	2	8



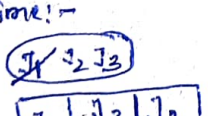
Avg(TAT) = 7.2



Avg(WT) = 3.2

⑤ Priority with arrival time:-

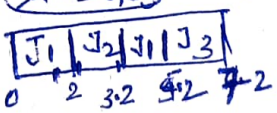
	AT	BT	Pri
J1	0	4	3
J2	2	1.2	5
J3	3	2.0	3



Avg(TAT) = 4.066ms
Avg(WT) = 1.667ms

Pre-emptive SJF

Ready Queue



Avg(TAT) = 3.533ms
Avg(WT) = 1.133ms

60 = 24 + 4 + 4 + 8 + 20