Case study and working of different Process Scheduling Algorithms.

We are going to provide detailed information on three types of scheduling algorithms:

1] FCFS ALGORITHM
2] SJF ALGORITHM
3] RR ALGORITHM

Here are certain full forms which are important.

FULL FORMS: -

AT: ARRIVAL TIME
CT: COMPLETION TIME
FT: FINISH TIME
TAT: TURN AROUND TIME
WT: WAITING TIME

Author: Kush Amit Shah

1] FCFS ALGORITHM

DEFINATION: -

→ FCFS **(First Come First Serve)** is a scheduling algorithm used in computer operating systems to manage the execution of processes.

The basic idea behind FCFS algorithm is quite simple: -

→ The first process that arrives is the first one to be executed, and subsequent processes are executed in the order of their arrival.

Working of FCFS algorithm: -
The First-Come, First-Served (FCFS) scheduling algorithm is a simple process scheduling algorithm that assigns CPU to processes based on their arrival time.

Here's a detailed description of how FCFS algorithm works:

→ **Arrival of Processes**: As processes arrive in the system, they are added to the ready queue. The ready queue is a data structure that holds processes waiting to be executed.

→ **Order of Execution:** The process that arrives first is the one that gets executed first. The CPU is assigned to the process at the front of the ready queue.

→ **Execution:** The selected process runs until it completes its execution or is preempted by a higher-priority process. Once a process starts executing, it continues until it finishes without any interruption.

→ **Completion:** Once a process completes its execution, the next process in the ready queue is selected for execution. This process continues until all processes in the system have been executed.

→ **Waiting Time:** The waiting time for a process is the total time the process spends waiting in the ready queue before getting CPU time.

→ **Turnaround Time:** Turnaround time is the total time taken by a process to complete its execution, including both waiting time and execution time.

→ **Visualization:** You can visualize the execution of processes on a Gantt chart, showing the timeline of when each process starts and finishes.

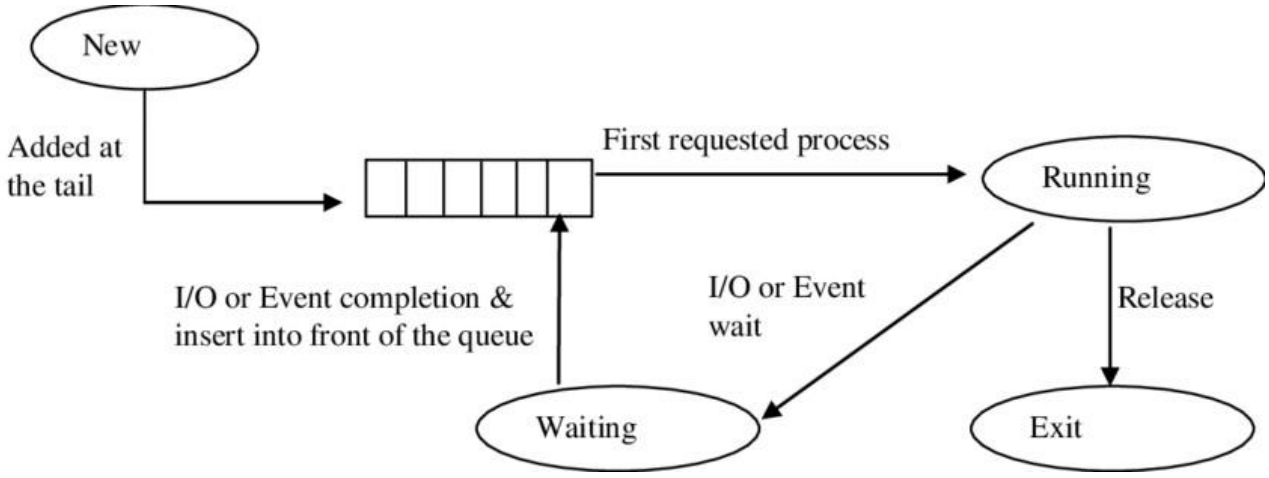
→ **Pros:**

1. The simplest form of a CPU scheduling algorithm
2. Easy to program 3. First come first served

→ **Cons:**

1. It is a Non-Preemptive CPU scheduling algorithm, so after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.
2. The Average Waiting Time is high.
3. Short processes that are at the back of the queue have to wait for the long process at the front to finish.
4. Not an ideal technique for time-sharing systems.
5. Because of its simplicity, FCFS is not very efficient.

PICTORIAL REPRESENTATION OF WORKING OF FCFS ALGORITHM: -



EXAMPLE: -

Process	Arrival Time	Burst Time
P1	0	4
P2	2	2
P3	5	3

SOLUTION: -

→ Gantt Chart:

P1	P2	P3

→ **Process Execution:** P1 arrives at time 0 and gets the CPU first because it's the only process. P1 executes for 4 units of time and completes its execution.

→ **Next Process:** P2 which arrived at time 2, is the next in line. It starts executing after P1. P2 executes for 2 units of time and completes its execution.

→ **Final Process:** P3 which arrived at time 5, is the next in line. It starts executing after P2. P3 executes for 3 units of time and completes its execution.

→ **Calculating Completion Time:** Completion time is the time at which a process completes its execution.

1. Completion time of P1 = 4
2. Completion time of P2 = 6
3. Completion time of P3 = 9

→ **Calculating Turnaround Time:** Turnaround time is the total time taken by a process to complete its execution.

1. Turnaround time of P1 = Completion time - Arrival time = 4 - 0 = 4 2. Turnaround time of P2 = Completion time - Arrival time = 6 - 2 = 4 3. Turnaround time of P3 = Completion time - Arrival time = 9 - 5 = 4

→ **Calculating Waiting Time:** Waiting time is the time a process spends waiting in the ready queue.

1. Waiting time of P1 = Turnaround time - Burst time = 4 - 4 = 0 2. Waiting time of P2 = Turnaround time - Burst time = 4 - 2 = 2
3. Waiting time of P3 = Turnaround time - Burst time = 4 - 3 = 1

→ **Calculating Average Waiting Time and Turnaround Time:**

1. Average Waiting Time = (0 + 2 + 1) / 3 = 1
2. Average Turnaround Time = (4 + 4 + 4) / 3 = 4

2] SJF ALGORITHM

DEFINATION: -

→ SJF (Shortest Job First) is a scheduling algorithm used in computer operating systems to manage the execution of processes.

→ The Shortest Job First (SJF) scheduling algorithm is a non-preemptive scheduling algorithm that selects the process with the smallest total remaining processing time.

The key idea behind SJF algorithm is: -

→ To prioritize the execution of the process with the shortest total burst time first. The burst time is the total time a process takes to execute on the CPU.

Here's a detailed description of how SJF algorithm works:

→ **Arrival of Processes:** Processes arrive at the ready queue. Each process has an associated burst time, which is the total time required to execute the process.

→ **Initial Queue:** The processes are initially in the ready queue. The scheduler selects the process with the shortest burst time from the ready queue.

→ **Execution:** The selected process is then dispatched for execution. The CPU is allocated to this process until it completes its execution.

→ **Completion:** Once the process completes its execution, it leaves the system.

→ **Selection of Next Process:** The scheduler then selects the process with the shortest burst time from the remaining processes in the ready queue.

→ **Waiting Time:** The waiting time for a process is the total time it spends in the ready queue before getting CPU time for execution.

→ **Turnaround Time:** Turnaround time is the total time taken by a process to complete its execution, including waiting time and execution time.

→ **Calculation of Burst Time:** Burst time can be provided by the user or estimated based on the previous behavior of the process.

→ **Pros: -**

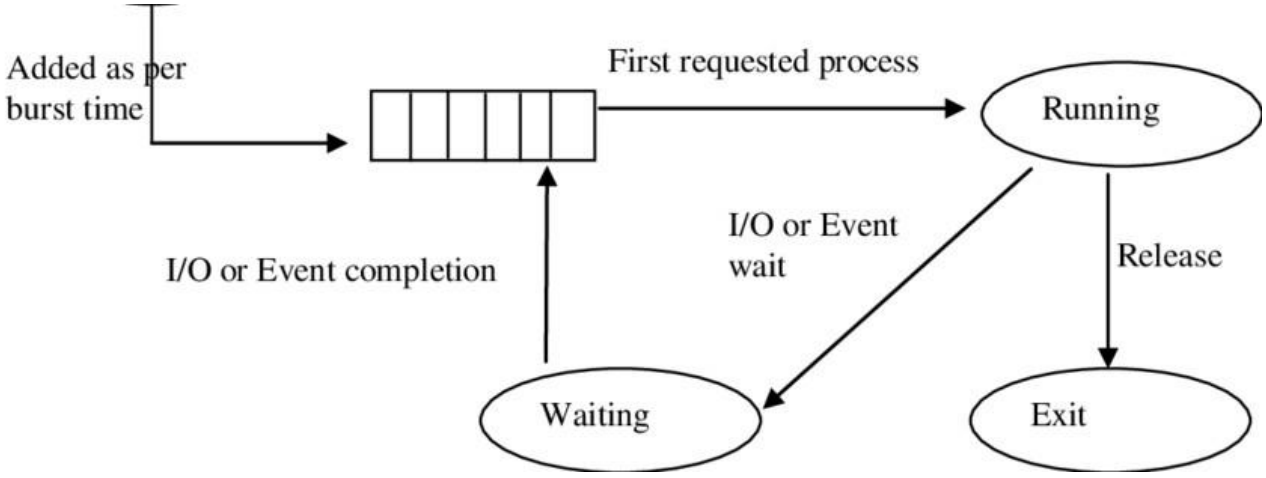
1. **Optimal for Minimizing Waiting Time:** SJF scheduling minimizes the waiting time of processes, as it selects the process with the shortest burst time first, leading to quicker turnaround times.
2. **Efficient Utilization of CPU:** The algorithm tends to maximize CPU utilization by selecting the process that will execute in the shortest time, thus allowing more processes to be executed in a given time frame.
3. **Predictable Execution Time:** SJF provides a more predictable execution time, as the process with the shortest burst time is scheduled next, allowing for better planning and resource allocation.

→ **Cons: -**

1. **Difficulty in Predicting Burst Times:** One major drawback is the difficulty in accurately predicting the burst time of a process. This information is typically not known in advance and can change dynamically.

2. **Starvation of Longer Processes:** Longer processes may suffer from starvation, especially if shorter processes continue to arrive. If a process with a long burst time is repeatedly preempted by shorter processes, it may wait a long time before getting CPU time.
3. **Indefinite Blocking:** In a scenario where new processes with shorter burst times keep arriving, longer processes may never get a chance to execute, leading to indefinite blocking.

PICTORIAL REPRESENTATION OF WORKING OF SJF ALGORITHM: -



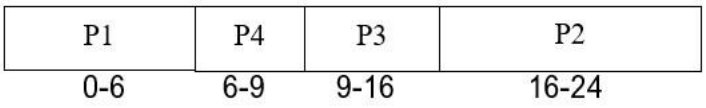
EXAMPLE: -

Process	Arrival Time	Burst Time
P1	0	6
P2	2	8
P3	4	7
P4	6	3

SOLUTION: -

→ **Sorting based on Arrival Time:** The processes are already sorted by arrival time in this example.

→ **Gantt Chart: -**



→ **Calculating Completion Time:** Select the process with the shortest burst time first.

1. Completion Time for P1 (6 units): P1 is the only process available initially.
2. Completion Time for P4 (9 units): P4 has the next shortest burst time.

→ **Calculating Waiting Time:**

1. Waiting Time for P1 (0 units): P1 started at time 0.
2. Waiting Time for P4 (3 units): P4 started at time 6, so it waited for 3 units.

→ **Calculating Turnaround Time:**

1. Turnaround Time for P1 (6 units): Time from arrival to completion for P1. 2. Turnaround Time for P4 (9 units): Time from arrival to completion for P4.

→ **Repeat Steps 2-4 for the remaining processes:**

1. Completion Time for P3 (16 units): P3 has the next shortest burst time.
2. Completion Time for P2 (24 units): P2 has the next shortest burst time.

→ **Final Calculations [Remaining Turnaround and Waiting Time]:**

1. Waiting Time for P2 (14 units): P2 started at time 10, so it waited for 14 units.
2. Waiting Time for P3 (12 units): P3 started at time 9, so it waited for 12 units.
3. Turnaround Time for P2 (22 units): Time from arrival to completion for P2.
4. Turnaround Time for P3 (19 units): Time from arrival to completion for P3.

3] RR ALGORITHM

DEFINATION: –

→ RR is a scheduling algorithm commonly used in computer operating systems to manage the execution of processes.

→ RR algorithm is designed to provide fair and equal access to the CPU for all processes in the system.

→ The Round Robin scheduling algorithm is a widely used approach in operating systems to allocate CPU time to multiple processes. It is a preemptive algorithm, meaning that the operating system can interrupt the currently executing process and allocate the CPU to another process.

Here's a detailed description of how RR algorithm works:

→ **Initialization:** Assign a fixed time unit or time quantum to each process. This time quantum is the maximum amount of time a process is allowed to run in one turn before being interrupted.

→ **Queue:** Maintain a ready queue to hold the processes that are ready to execute. Initially, all processes are placed in this queue.

→ **Execution:** The operating system selects a process from the front of the ready queue and allocates the CPU to it for the defined time quantum.

→ **Time Quantum Expiry:** If the process completes its execution within the time quantum, it is moved to the back of the ready queue. If the process does not finish within the time quantum, it is moved to the back of the ready queue, and the next process in the queue is given a turn.

→ **Cycle Continuation:** The process continues in a cyclic manner. The operating system keeps selecting processes from the front of the ready queue and allocates CPU time until all processes are completed.

→ **Completion:** The process repeats until all processes are executed.

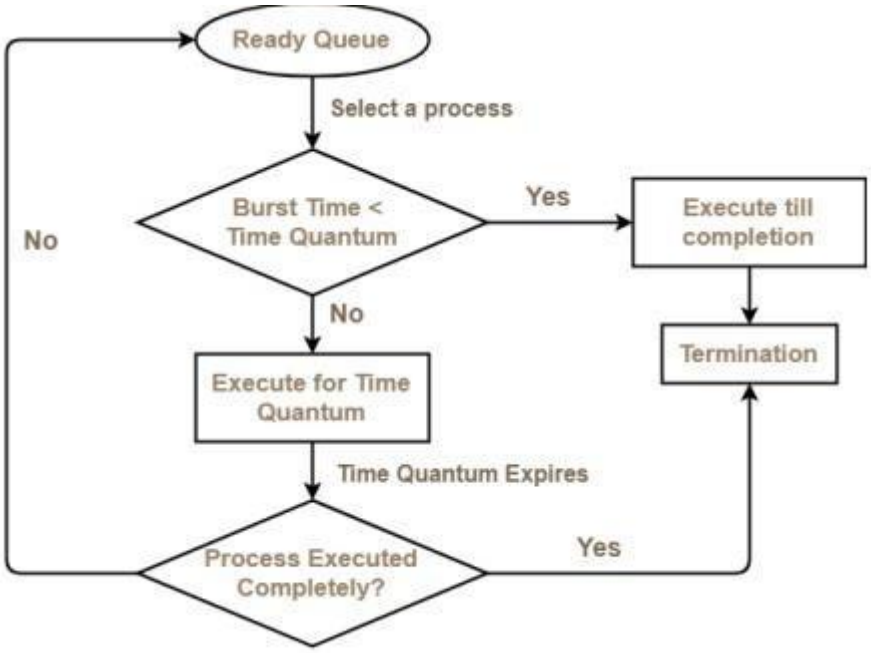
→ **Pros: -**

1. **Fairness:** Provides fairness to all processes in terms of CPU time.
2. **Simple Implementation:** The algorithm is simple to implement and understand.
3. **No Starvation:** No process starves for resources since each process gets a turn.

→ **Cons:**

1. **Poor Performance for Certain Workloads:** Round Robin may not perform optimally for processes with highly variable execution times.
2. **Overhead:** The frequent context switching can introduce overhead.

PICTORIAL REPRESENTATION OF WORKING OF RR ALGORITHM: -



EXAMPLE: -

Process	Arrival Time	Burst Time
P1	0	20
P2	0	15
P3	0	25

SOLUTION: -

→ **Initial State:** All processes are in the ready queue: [P1, P2, P3]. The scheduler starts with the first process in the queue.

→ **First Round (Time Units 0-10):** P1 starts executing and runs for the first 10 units. At the end of its time quantum, P1 is moved to the back of the queue: [P2, P3, P1].

→ **Second Round (Time Units 10-20):** P2 starts executing and runs for the entire time quantum (15 units). P2 is moved to the back of the queue: [P3, P1, P2].

→ **Third Round (Time Units 20-30):** P3 starts executing and runs for the first 10 units. At the end of its time quantum, P3 is moved to the back of the queue: [P1, P2, P3].

→ **Fourth Round (Time Units 30-40):** P1 resumes execution and runs for the next 10 units. P1 is moved to the back of the queue: [P2, P3, P1].

→ **Fifth Round (Time Units 40-50):** P2 resumes execution and runs for the remaining 5 units of its burst time. P2 is moved to the back of the queue: [P3, P1, P2].

→ **Sixth Round (Time Units 50-60): P3** resumes execution and runs for the remaining 15 units of its burst time. P3 completes its execution and is removed from the queue: [P1, P2].

→ **Seventh Round (Time Units 60-70):** P1 resumes execution and completes its remaining 10 units of burst time. P1 is removed from the queue: [P2].

→ **Eighth Round (Time Units 70-80):** P2 resumes execution and completes its remaining 5 units of burst time. P2 is removed from the queue.

→ **Ninth Round (Time Units 80-90):** The queue is now empty, and the system is idle.

ROUNDS REPRESENTAED IN TABULAR FORM: -

Time Units	Ready Queue	Execution (Process)	Remaining Burst Time
0-10	[P1, P2, P3]	P1	P1: 10
10-20	[P2, P3, P1]	P2	P2: 15
20-30	[P3, P1, P2]	P3	P3: 15
30-40	[P1, P2, P3]	P1	P1: 10
40-50	[P2, P3, P1]	P2	P2: 5
50-60	[P3, P1, P2]	P3	P3: 0 (completed)
60-70	[P1, P2]	P1	P1: 0 (completed)
70-80	[P2]	P2	P2: 0 (completed)
80-90	[]	-	-

→ Gantt Chart: -

0	10	20	30	40	50	60	70	80	90
|P1|	|P1|	|P2|	|P3|	|P1|	|P2|	|P3|	|P1|	|P2|	||