ASSIGNMENT No: 3

Title : SJF & Round Robin CPU Scheduling

AIM: Implement C program for CPU scheduling algorithms: Shortest Job First (SJF) and Round Robin with different arrival time.

THEORY:

Scheduling

- 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.
- Whenever the CPU becomes idle, the operating system must select one of the processes in the line ready for launch.
- The selection process is done by a temporary (CPU) scheduler.
- The Scheduler selects between memory processes ready to launch and assigns the CPU to one of them.

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?

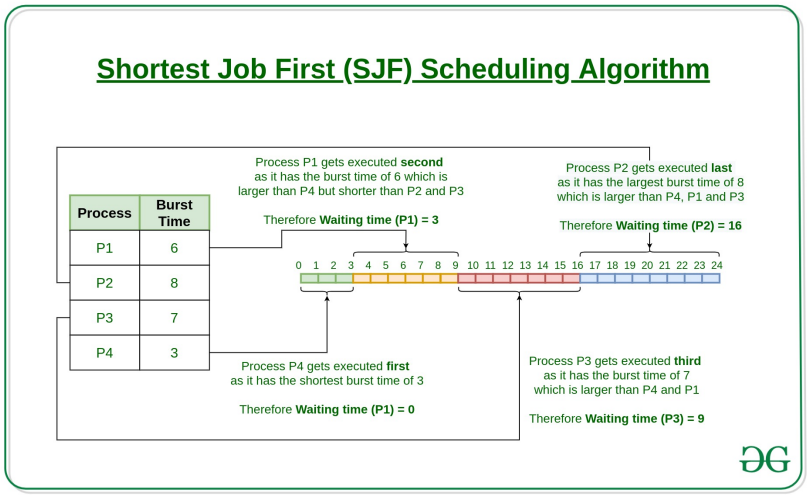
- **Arrival Time:** Time at which the process arrives in the ready queue.
- **Completion Time:** Time at which process completes its execution.
- **Burst Time:** Time required by a process for CPU execution.
- **Turn Around Time:** Time Difference between completion time and arrival time.

  Turn Around Time = Completion Time – Arrival Time
- **Waiting Time(W.T):** Time Difference between turn around time and burst time.

  Waiting Time = Turn Around Time – Burst Time

1.Shortest Job First scheduling

The CPU scheduling algorithm Shortest Job First (**SJF**), allocates the CPU to the processes according to the process with smallest execution time.



Advantages of SJF:

- It has the minimum waiting time among all the scheduling algorithms.
- A process having larger burst time may get into starvation but the problem can be solved using concept of Ageing.
- It is a greedy algorithm and provides optimal scheduling.

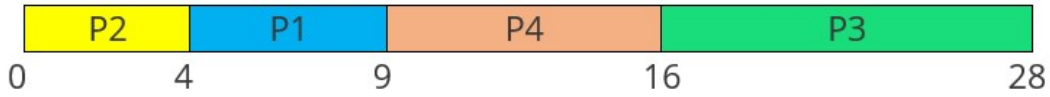
SJF Scheduling Algorithm:

1. Enter number of processes.
2. Enter the **burst time** of all the processes.
3. Sort all the processes according to their **burst time**.
4. Find waiting time, **WT** of all the processes.
5. For the smallest process, **WT = 0**.
6. For all the next processes **i**, find waiting time by adding burst time of all the previously completed process.
7. Calculate **Turnaround time = WT + BT** for all the processes.
8. Calculate **average waiting time = total waiting time / no. of processes**.
9. Calculate **average turnaround time= total turnaround time / no. of processes**.

SJF Example:

Process	Arrival Time	Burst Time
P1	0	5
P2	0	4
P3	0	12
P4	0	7

Gantt Chart:



Waiting Time: Time Difference between turnaround time and burst time.
Waiting Time = Turnaround Time – Burst Time
P1 waiting time: 4
P2 waiting time: 0

P3 waiting time: 16
P4 waiting time: 9

Average Waiting Time = (4 + 0 + 16 + 9)/4 = 29/4 = 7.25
Turnaround Time: Difference between completion time and arrival time.
Turnaround Time = Completion Time – Arrival Time
P1 turnaround time: 9-0 = 9
P2 turnaround time: 4-0 = 4
P3 turnaround time: 28-0 = 28
P4 turnaround time: 16-0 = 16

Average Turnaround Time = (9 + 4 + 28 + 16)/4 = 14.25

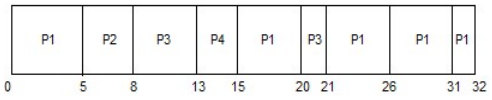
2. Round Robin(RR) scheduling algorithm

Round Robin Scheduling is a CPU scheduling algorithm in which each process is executed for a fixed time slot. Since the resources are snatched after the time slot, round robin is preemptive.

This algorithm is similar to FCFS scheduling, but in Round Robin(RR) scheduling, preemption is added which enables the system to switch between processes.

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for round robin scheduling will be,



The average waiting time will be, 11 ms.

- A fixed time is allotted to each process, called a **quantum**, for execution.
- Once a process is executed for the given time period that process is preempted and another process executes for the given time period.
- Context switching is used to save states of preempted processes.
- This algorithm is simple and easy to implement and the most important is thing is this algorithm is starvation-free as all processes get a fair share of CPU.
- It is important to note here that the length of time quantum is generally from 10 to 100 milliseconds in length.

Round Robin Scheduling Algorithm:
Step 1: Start the Program.
Step 2: Input the number of processes.
Step 3: Input the burst time and arrival time of each process and the limit of the time slot.
Step 4: Push all processes into the ready queue according to their arrival time. Then execute each process upto time slot and push left over process in queue again for execution.
Step 5: After a process is completely executed, print its turn around time and waiting time.

Example:
Following is the example of round robin scheduling.

Process Id	Arrival Time	Burst Time
P1	0	10
P2	1	8
P3	2	7

Time Slot is 5 Sec.
First **P1** is executed for 5 seconds, left burst time is 5 sec
Then **P2** is executed for 5 seconds, left burst time is 3 sec
Then **P3** is executed for 5 seconds, left burst time is 2 sec
Then **P1** is executed for 5 seconds, execution of **P1** is completed.
Then **P2** is executed for 3 seconds, execution of **P2** is completed.
Then **P1** is executed for 2 sec, execution **P3** is completed.
Execution of all processes completed

Process Id	Burst Time	Wait Time	Turn Around Time
P1	10	20	10
P2	8	22	14
P3	7	23	16

Average Waiting Time = (20 + 22 + 23)/3 = 65/3 = 21.666666
Average Turnaround Time = (10 + 14 + 16)/3 = 40/3 = 13.333333

Advantages:

- There is no starvation of resources as each process gets equal share.
- Every Process gets equal allocation of CPU.
- Increases Performance time in terms of response time.

Disadvantages:

- More time is wasted on context switching.
- Frequent context switching increases CPU overhead.
- Average Waiting time increases for each process.