Module: Operating System Concepts
Session: CPU Scheduling

By

Donkada Mohana Vamsi

C-DAC Hyderabad

First-Come, First-Serve

- In First Come First Serve (FCFS) Scheduling Algorithm, the process which requests the CPU first will be first assigned access to the CPU
- It is the easiest algorithm to implement
- The average waiting time for FCFS scheduling algorithm is high

First-Come,First-Serve

Criteria to form queue : Arrival time

Mode : Non-preemptive

As the CPU is allocated to the process, the process keeps the CPU until it releases the CPU voluntarily either by terminating or by I/O request.

First-Come,First-Serve

P.no	Burst time	Arrival time
P1	24 ns	0
P2	3 ns	0
P3	3 ns	0

Applying First Come First Serve for above data

First-Come,First-Serve

Ready queue - In this example, it is assumed that all the processes arrive in the **ready queue** at the same time

P.no	Burst time	Arrival time
P1	24 ns	0
P2	3 ns	0
P3	3 ns	0

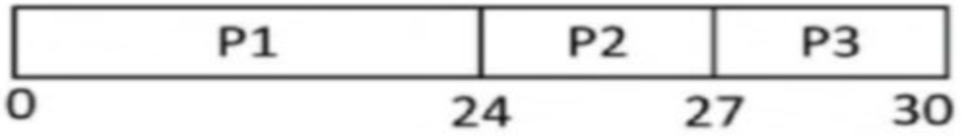
| P3 | P2 | P1 | ← Queue formation

Gantt chart

- The Gantt chart was given by an American social scientist Henry Gantt in 1971. It is used for graphical illustration of a schedule.
- Which helps to plan,coordinate, and track specific tasks in a project

First-Come,First-Serve

- Gantt chart representation of execution of the processes:



- Process P1 enters the system first, since we are following the FCFS scheduling algorithm, therefore P1 utilize the CPU first for the burst time of 24ns. Then at 24ns , the CPU is allocated to P2(3ns). Now at 27ns, P3 starts its execution for 3ns

First-Come,First-Serve

P.no	Burst time	Arrival time
P1	24 ns	0
P2	3 ns	0
P3	3 ns	0

Calculating Turnaround time & Waiting time for above example

First-Come,First-Serve

Turnaround time : Turnaround time is the interval between the submission of the process: Arrival Time(AT) and its Completion Time(CT)

TAT(P1) = **t1**= CT(P1)-AT(P1) = 24-0= **24**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 27-0= **27**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 30-0= **30**

First-Come,First-Serve

Average turnaround time = Sum of individual process turnaround time / No of processes

Avg = t1 + t2 + t3 / 3

24+27+30/3 = 81/3 = **27 ns**

Avg = 27 ns (Average turnaround time)

First-Come,First-Serve

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 24-24=0

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 27-3=24

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 30-3=27

First-Come,First-Serve

Average waiting time = Sum of individual process waiting time / No of processes

Avg = w1 + w2 + w3 / 3

0+24+27/3 = 51/3 = **17 ns**

Avg = 27 ns (Average waiting time)

First-Come,First-Serve - Example 2

First-Come,First-Serve - Example 2

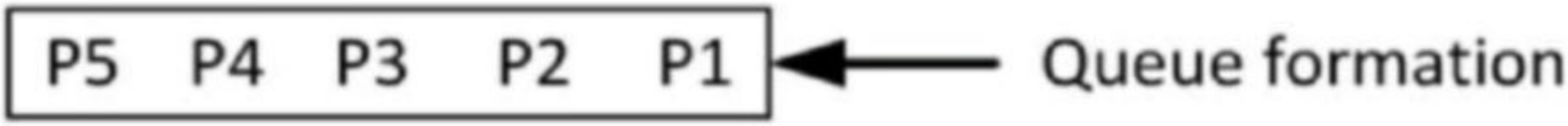
P.no	Burst time	Arrival time
P1	4 ns	0
P2	3 ns	1
P3	1 ns	2
P4	2 ns	3
P5	5 ns	4

Calculating Turnaround time & Waiting time for above example

First-Come,First-Serve - Example 2

P.no	Burst time	Arrival time
P1	4 ns	0
P2	3 ns	1
P3	1 ns	2
P4	2 ns	3
P5	5 ns	4

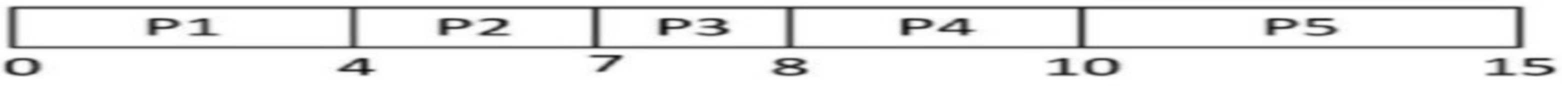
Queue formation



First-Come,First-Serve - Example 2

P.no	Burst time	Arrival time
P1	4 ns	0
P2	3 ns	1
P3	1 ns	2
P4	2 ns	3
P5	5 ns	4

Gantt chart representation of execution of the process



First-Come,First-Serve - Example 2

Turnaround time :

TAT(P1) = **t1**= CT(P1)-AT(P1) = 4-0= **4**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 7-1= **6**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 8-2= **6**

TAT(P4) = **t4**= CT(P4)-AT(P4) = 10-3= **7**

TAT(P5) = **t5**= CT(P5)-AT(P5) = 15-4= **11**

First-Come,First-Serve - Example 2

Average turnaround time = Sum of individual process turnaround time / No of processes

Avg = t1 + t2 + t3 + t4 + t5 / 5

4+6+6+7+11/5 = 34/5 = **6.8 ns**

Avg = 6.8 ns (Average turnaround time)

First-Come,First-Serve - Example 2

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 4-4=0

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 6-3=3

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 6-1=5

Waiting time (P4) = w4 = TAT(P4) - BT(P4) = 7-2=5

Waiting time (P5) = w5 = TAT(P5) - BT(P5) = 11-5=6

First-Come,First-Serve - Example 2

Average waiting time = Sum of individual process waiting time / No of processes

Avg = w1 + w2 + w3 + w4 + w5 / 5

0+3+5+5+6/5 = 19/5 = **3.8 ns**

Avg = 3.8 ns (Average waiting time)

First-Come,First-Serve - Features

Non-preemption : Once the job is arrived, first that jobs is finished then other will start

Sequential processing : The requests are being processed in the order in which they are coming

Waiting time and average turnaround time are long, so **throughput is less (System performance is low)**

First-Come,First-Serve - Features

Convoy effect: When the process blocks the CPU, for a long time, it is called the convoy effect. If lengthy job arrives first, then waiting time and turnaround time increases for the late arriving process

Shortest-Job-First Scheduling

- In Shortest Job First(SJF) Scheduling algorithm, the CPU is allocated to the process having the smallest CPU burst time.
- If the next CPU burst time of two processes is same, FCFS Scheduling is used to break the tie. It can be implemented as both **preemptive** and **non-preemptive**.
- SJF algorithm has the minimal waiting time

Shortest-Job-First Scheduling

Criteria to form queue : Burst time

Mode : Non-preemptive

- As the CPU is allocated to the process, the process keeps the CPU until its releases the CPU voluntarily either by terminating or by I/O request.
- SJF gives optimal average waiting time when compared with FCFS

Shortest-Job-First Scheduling

P.no	Burst time
P1	24 ns
P2	3 ns
P3	3 ns

Applying SJF algorithm for above data

Shortest-Job-First Scheduling

Ready queue - In this example, it is assumed that all the processes arrive in the **ready queue** at the same time

P.no	Burst time
P1	24 ns
P2	3 ns
P3	3 ns

Shortest-Job-First Scheduling

- Gantt chart representation of execution of the processes:

- As P2 is having smallest burst time, it will be allocated to CPU first. It utilizes the CPU for 3 ns. Then, it is the turn of P3 to execute as per the SJF scheduling algorithm.It utilizes the CPU for 3 ns. At last P1 executes for 24 ns.

Shortest-Job-First Scheduling

P.no	Burst time
P1	24 ns
P2	3 ns
P3	3 ns

Calculating Turnaround time & Waiting time for the above example

Shortest-Job-First Scheduling

Turnaround time : Turnaround time is the interval between the submission of the process(AT) and its completion(CT)

TAT(P1) = **t1**= CT(P1)-AT(P1) = 30-0= **30**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 3-0= **3**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 6-0= **6**

Shortest-Job-First Scheduling

Average turnaround time = Sum of individual process turnaround time / No of processes

Avg = t1 + t2 + t3 / 3

30+3+6/3 = 39/3 = **13 ns**

Avg = 13 ns (Average turnaround time)

Shortest-Job-First Scheduling

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 30-24=6

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 3-3=0

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 6-3=3

Shortest-Job-First Scheduling

Average waiting time = Sum of individual process waiting time / No of processes

Avg = w1 + w2 + w3 / 3

6+0+3/3 = 9/3 = **3 ns**

Avg = 3 ns (Average waiting time)

Shortest-Job-First Scheduling

Advantages

- Minimum average waiting time and turnaround time

Disadvantages

- Longer jobs need to starve

Shortest Remaining Time First (SJF-preemptive)

- The preemptive version of the shortest job first is known as the shortest remaining time first scheduling algorithm.
- In this scheme,CPU is allocated to the process having the shortest remaining CPU burst time
- If the remaining CPU burst time of two processes is same, FCFS scheduling is used to break the tie

Shortest Remaining Time First (SJF-preemptive)

Criteria to form queue : Burst time

Mode : Preemptive

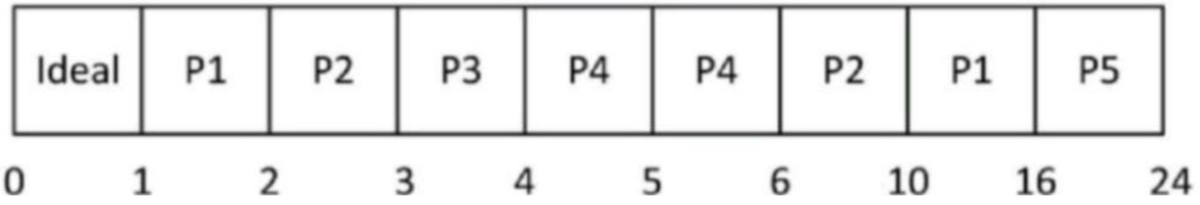
Shortest Remaining Time First

Process Number	Arrival Time	Burst Time
p1	1	7
P2	2	5
P3	3	1
P4	4	2
P5	5	8

Applying SRTF algorithm for above data

Shortest Remaining Time First

- Gantt chart representation of execution of the processes:



- In this algorithm, the process with shortest burst time will be allocated first to the CPU, but preempt the current process from execution and replace with the process having shortest burst time from read queue

Shortest Remaining Time First

Process Number	Arrival Time	Burst Time
p1	1	7
P2	2	5
P3	3	1
P4	4	2
P5	5	8

Calculating Turnaround time & Waiting time for the above example

Shortest Remaining Time First

Turnaround time :

TAT(P1) = **t1**= CT(P1)-AT(P1) = 16-1= **15**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 10-2= **8**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 4-3= **1**

TAT(P4) = **t4**= CT(P4)-AT(P4) = 6-4= **2**

TAT(P5) = **t5**= CT(P5)-AT(P5) = 24-5= **19**

Shortest Remaining Time First

Average turnaround time = Sum of individual process turnaround time / No of processes

Avg = t1 + t2 + t3 + t4 + t5 / 5

15+8+1+2+19/5 = 45/5 = **9 ns**

Avg = 9 ns (Average turnaround time)

Shortest Remaining Time First

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 15-7=8

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 8-5=3

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 1-1=0

Waiting time (P4) = w4 = TAT(P4) - BT(P4) = 2-2=0

Waiting time (P5) = w5 = TAT(P5) - BT(P5) = 19-8=11

Shortest Remaining Time First

Average waiting time = Sum of individual process waiting time / No of process

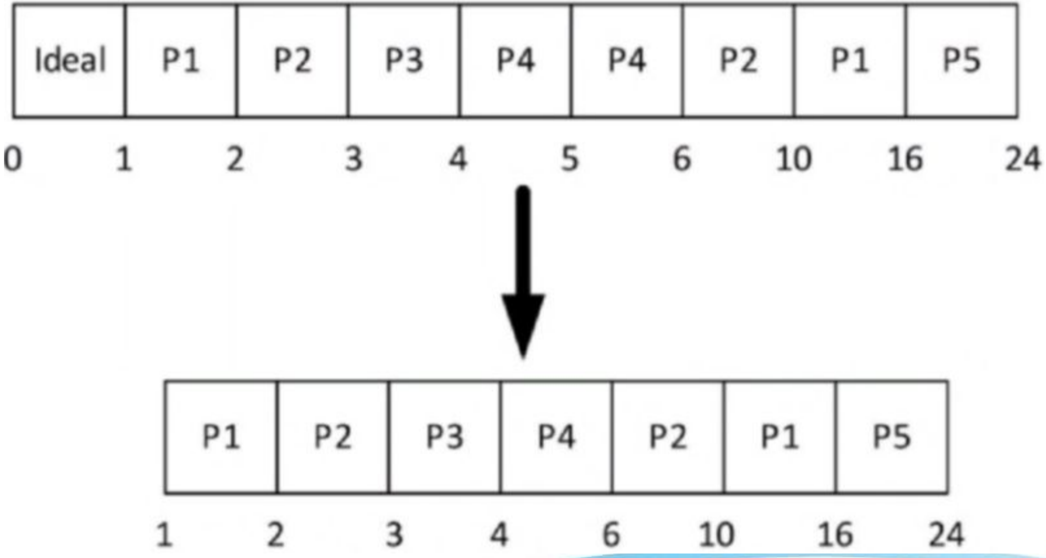
Avg = w1 + w2 + w3 + w4 + w5 / 3

8+3+0+0+11/5 = 22/5 = **4.4 ns**

Avg = 4.4 ns (Average waiting time)

Shortest Remaining Time First

No of Context switches = 6



Shortest Remaining Time First

Disadvantages

- Suffers from starvation

Longest-Job-First Scheduling

- In Longest Job First(LJF) Scheduling algorithm, the CPU is allocated to the process having the largest CPU burst time.
- If the next CPU burst time of two process is same, FCFS Scheduling is used to break the time. It can be implemented as both **preemptive** and **non-preemptive**.

Longest-Job-First Scheduling

Criteria to form queue : Burst time

Mode : Non-preemptive

As the CPU is allocated to the process, the process keeps the CPU until its releases the CPU voluntarily either by terminating or by I/O request.

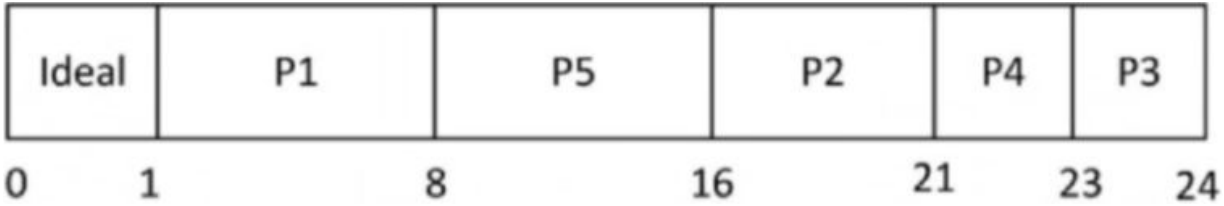
Longest-Job-First Scheduling

Process Number	Arrival Time	Burst Time
p1	1	7
P2	2	5
P3	3	1
P4	4	2
P5	5	8

Applying LJF algorithm for above data

Longest-Job-First Scheduling

- Gantt chart representation of execution of the processes:



- In this algorithm, the process with longest burst time will be allocated the CPU first with respect to arrival time

Longest-Job-First Scheduling

Process Number	Arrival Time	Burst Time
p1	1	7
P2	2	5
P3	3	1
P4	4	2
P5	5	8

Calculating Turnaround time & Waiting time for above example

Longest-Job-First Scheduling

Turnaround time :

TAT(P1) = **t1**= CT(P1)-AT(P1) = 8-1= **7**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 21-2= **19**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 24-3= **21**

TAT(P4) = **t4**= CT(P4)-AT(P4) = 23-4= **19**

TAT(P5) = **t5**= CT(P5)-AT(P5) = 16-5= **11**

Longest-Job-First Scheduling

Average turnaround time = Sum of individual process turnaround time / No of processes

Avg = t1 + t2 + t3 + t4 + t5 / 5

7+19+21+19+11/5 = 77/5 = **15.4 ns**

Avg = 15.4 ns (Average turnaround time)

Longest-Job-First Scheduling

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 7-7=0

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 19-5=14

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 21-1=20

Waiting time (P4) = w4 = TAT(P4) - BT(P4) = 19-2=17

Waiting time (P5) = w5 = TAT(P5) - BT(P5) = 11-8=3

Longest-Job-First Scheduling

Average waiting time = Sum of individual process waiting time / No of processes

Avg = w1 + w2 + w3 + w4 + w5 / 3

0+14+20+17+3/5 = 54/5 = **10.8 ns**

Avg = 10.8 ns (Average waiting time)

Longest Remaining Time First

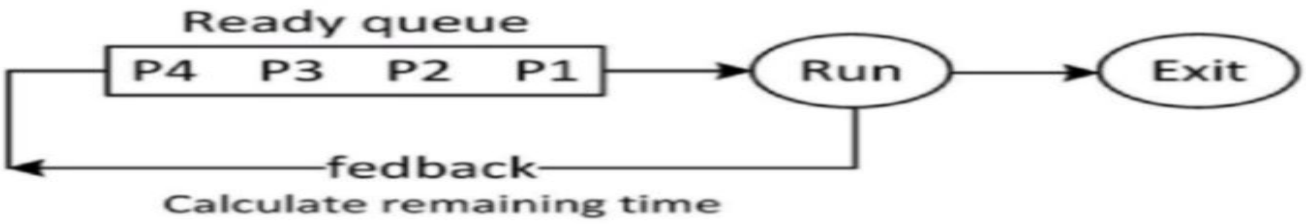
- **Preemptive** version of LJF. The priority is given for longest burst time
- The algorithm implementation is similar to SRTF. However, here priority is given to the process having the longest burst time.

Round Robin Scheduling Algorithm

- Round Robin scheduling algorithm is designed for time-sharing systems.
- It is similar to FCFS scheduling but preemption is added to switch between processes.
- A time quantum is allocated to each process for execution

Round Robin Scheduling Algorithm

Once a process is executed for a given period of time, it is preempted and another process is given a chance to execute for a given period of time. This is a preemptive scheduling algorithm



Round Robin Scheduling Algorithm

Criteria to form queue :

Time quantum/time slice + arrival time

Mode : Preemptive

Preemptive because after the time quantum expires, CPU is allocated to next process.

Round Robin Scheduling Algorithm

Process	Burst Time
P_1	24
P_2	3
P_3	3

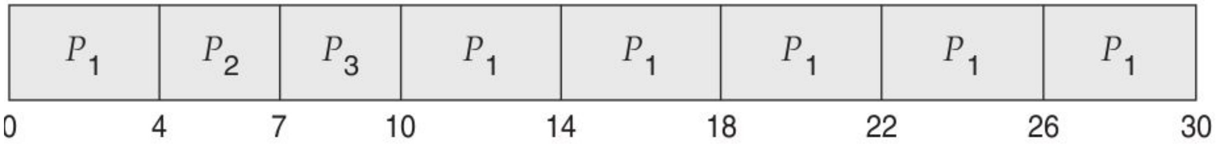
Applying Round Robin with time quantum 4ns for above data

Round Robin Scheduling Algorithm

Every process will get the access to CPU for execution of 4ns at one time. If the process has a larger burst time than 4ns, it will be preempted and will be placed in the ready queue at the end. The next job would be executed in the next turn

Round Robin Scheduling Algorithm

Gantt chart representation of execution of the processes:



Round Robin Scheduling Algorithm

Completion order by using RR : P2 P3 P1

Turnaround time :

TAT(P1) = **t1**= CT(P1)-AT(P1) = 30-0= **30**

TAT(P2) = **t2**= CT(P2)-AT(P2) = 7-0= **7**

TAT(P3) = **t3**= CT(P3)-AT(P3) = 10-0= **10**

Average TAT = 30+7+10/3 = 15.6ns

Round Robin Scheduling Algorithm

Waiting time : Waiting time is the interval between turnaround time and burst time

Waiting time (P1) = w1 = TAT(P1) - BT(P1) = 30-24=6

Waiting time (P2) = w2 = TAT(P2) - BT(P2) = 7-3=4

Waiting time (P3) = w3 = TAT(P3) - BT(P3) = 10-3=7

Round Robin Scheduling Algorithm

Average waiting time = Sum of individual process waiting time / No of processes

Avg = w1 + w2 + w3 / 3

6+4+7/3 = 17/3 = **5.6 ns**

Avg = 5.6 ns (Average waiting time)

Round Robin Scheduling Algorithm

- Effect of **time quantum** on the performance of a system in RR.
- If **time quantum is less,** then the number of **context switches increases**.
- If the **time quantum is too large,** then the number of **context switches decrease** and it is same as **FCFS** policy

Priority Scheduling Algorithm

- The implementation of Priority Scheduling Algorithm can be in preemptive and non-preemptive.
- The overall implementation of algorithm is based on priority of the process. It is similar to SJF. However, here criteria is **priority of the process** instead of Burst time

Thank you