SCHEDULING ALGORITHMS - I

Scheduling Algorithms:

1. First come First Serve:

FCFS is the simplest **non-preemptive** algorithm. Processes are assigned the CPU in the order they request it. That is the process that requests the CPU first is allocated the CPU first. The implementation of FCFS is policy is managed with a FIFO(First in first out) queue. When the first job enters the system from the outside in the morning, it is started immediately and allowed to run as long as it wants to. As other jobs come in, they are put onto the end of the queue. When the running process blocks, the first process on the queue is run next. When a blocked process becomes ready, like a newly arrived job, it is put on the end of the queue.

Advantages:
1.Easy to understand and program. With this algorithm a single linked list keeps track of all ready processes.
2.Equally fair.,
3.Suitable specially for Batch Operating system.

Disadvantages:
1.FCFS is not suitable for time-sharing systems where it is important that each user should get the CPU for an equal amount of arrival time.

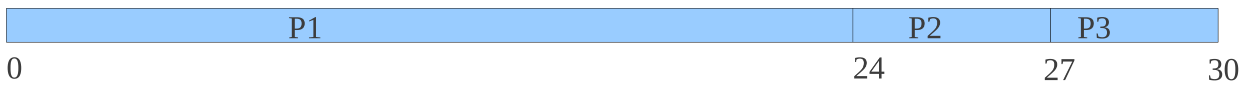
Consider the following set of processes having their burst time mentioned in milliseconds. CPU burst time indicates that for how much time, the process needs the cpu.

Process	Burst Time
P1	24
P2	3
P3	3

Calculate the average waiting time if the processes arrive in the order of:
a). P1, P2, P3

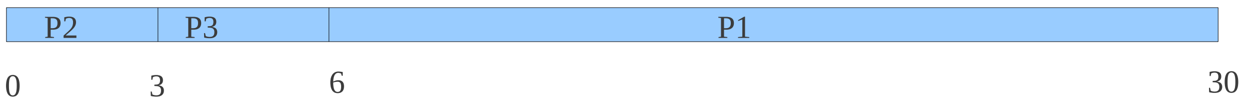
b). P2, P3, P1

a. The processes arrive the order P1, P2, P3. Let us assume they arrive in the same time at 0 ms in the system. We get the following gantt chart.



Waiting time for P1= 0ms , for P2 = 24 ms for P3 = 27ms
Avg waiting time: (0+24+27)/3= 17

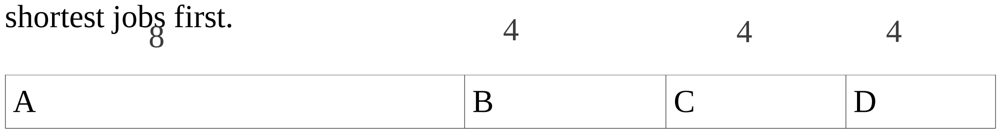
b.) If the process arrive in the order P2,P3, P1



Average waiting time: (0+3+6)/3=3. Average waiting time vary substantially if the process CPU burst time vary greatly.

2. Shortest Job First:

When several equally important jobs are sitting in the i/p queue waiting to be started, the scheduler picks the shortest jobs first.



Original order: (Turn Around time)

Here we have four jobs A,B,C ,D with run times of 8 , 4, 4 and 4 minutes respectively. By running them in that order the turnaround time for A is 8 minutes, for B 12 minutes, for C 16 minutes and for D 20 minutes for an average of 14 minutes.
Now let us consider running these jobs in the shortest Job First.

B C D and then A.

the turnaround times are now , 4, 8, 12 and 20 minutes giving the average of 11. Shortest job first is probably optimal.

Consider the four jobs with run times of a, b,c,d. The first job finished at a, the second at a+b and so on. So the mean turnaround time is (4a+3b+2c+d)/4. It is clear that the a contributes more to the average than any other. So it should be the shortest one.

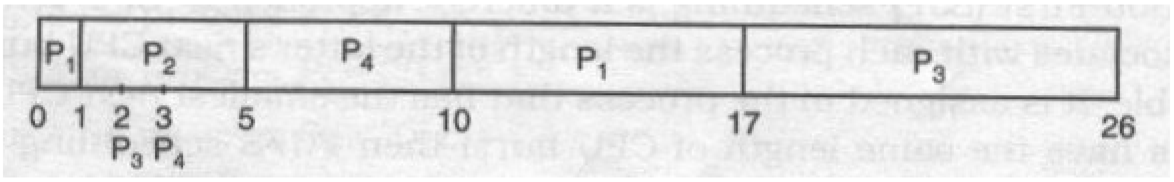
The disadvantages of this algorithm is the problem to know the length of time for which CPU is needed by a process. The SJF is optimal when all the jobs are available simultaneously.

The SJF is either preemptive or non preemptive. Preemptive SJF scheduling is sometimes called **Shortest Remaining Time First** scheduling. With this scheduling algorithms the scheduler always chooses the process whose remaining run time is shortest.
When a new job arrives its total time is compared to the current process remaining time. If the new job needs less time to finish than the current process, the current process is suspended and the new job is started. This scheme allows new short jobs to get good service.

Q). Calculate the average waiting time in 1). Preemptive SJF and 2). Non Preemptive SJF
Note: SJF Default: (Non Preemptive)

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

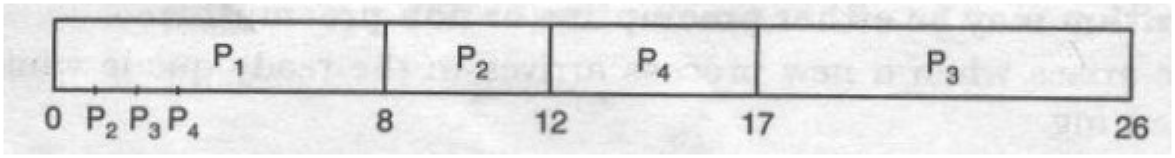
a. Preemptive SJF (Shortest Remaining Time First):



At t=0ms only one process P1 is in the system, whose burst time is 8ms; starts its execution. After 1ms i.e., at t=1, new process P2 (Burst time= 4ms) arrives in the ready queue. Since its burst time is less than the remaining burst time of P1 (7ms) P1 is preempted and execution of P2 is started.
Again at t=2, a new process P3 arrive in the ready queue but its burst time (9ms) is larger than remaining burst time of currently running process (P2 3ms). So P2 is not preempted and continues its execution. Again at t=3 , new process P4 (burst time 5ms) arrives . Again for same reason P2 is not preempted until its execution is completed.

Waiting time of P1: 0ms + (10 – 1)ms = 9ms
Waiting time of P2: 1ms – 1ms = 0ms
Waiting time of P3: 17ms – 2ms = 15ms
Waiting time of P4: 5ms – 3ms = 2ms
Avg waiting time: (9+0+15+2)/4 = 6.5ms

Non-preemptive SJF:



Since its non-preemptive process is not preempted until it finish its execution.
Waiting time for P1: 0ms
Waiting time for P2: (8-1)ms = 7ms
Waiting time for P3: (17 – 2) ms = 15ms
Waiting time for P4: (12 – 3)ms = 9ms

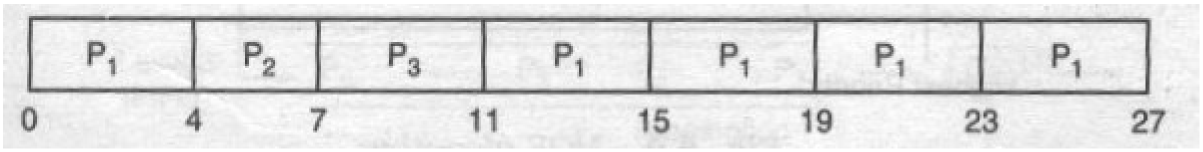
Average waiting time: (0+7+15+9)/4 = 7.75ms

3. Round-Robin Scheduling Algorithms:

•One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR).
•In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum.
•If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.
•If the the process has blocked or finished before the quantum has elapsed the CPU switching is done.
•Round Robin Scheduling is preemptive (at the end of time-slice) therefore it is effective in time-sharing environments in which the system needs to guarantee reasonable response times for interactive users.
•The only interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates FCFS.
•In any event, the average waiting time under round robin scheduling is on quite long.
Consider the following set of processes that arrives at time 0 ms.

Process	Burst Time
P1	20
P2	3
P3	4

If we use time quantum of 4ms then calculate the average waiting time using R-R scheduling.



According to R-R scheduling processes are executed in FCFS order. So, firstly P1(burst time=20ms) is executed but after 4ms it is preempted and new process P2 (Burst time = 3ms) starts its execution whose execution is completed before the time quantum. Then next process P3 (Burst time=4ms) starts its execution and finally remaining part of P1 gets executed with time quantum of 4ms.

Waiting time of Process P1: 0ms + (11 – 4)ms = 7ms
Waiting time of Process P2: 4ms
Waiting time of Process P3: 7ms

Average Waiting time: (7+4+7)/3=6ms

4. Priority Scheduling:

Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf