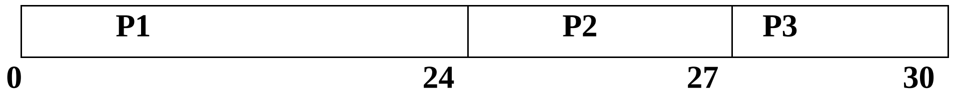
FCFS Scheduling Algorithm (non-preemptive)

Example:

Let processes P1, P2, and P3 arrive to the ready queue in that order and let the run times (CPU burst times) of these processes be as follows:

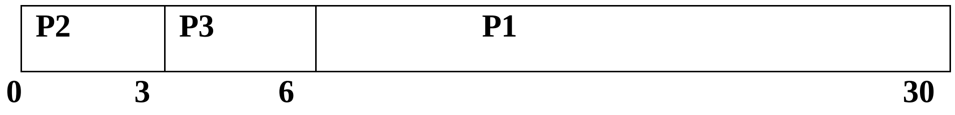
Process	**CPU burst**
P1	**24**
P2	**3**
P2	**3**

These processes are served by the CPU in FCFS order. The result can be shown in the following Gantt chart:



The waiting time for P1 is: **W (P1) = 0 ms**
The waiting time for P2 is**:** **W (P2) = 24 ms**
The waiting time for P3 is: **W (P3) = 27 ms**
→ The average waiting time is: W = (0+24+27)/3 = 17 ms.

If these processes arrived in the following order: P2, P3, P1 then the Gantt chart would be as follows:



The average waiting time is now: W = (0+3+6)/3 = 3 ms

Exercise:

Consider the following processes with the relative CPU bursts which arrive together in the order given.

Process	**CPU burst**
A	**16**
B	**4**
C	**1**

Draw the Gantt charts, and find out the average waiting time for FCFS CPU scheduling algorithm if:

a) Coming order is A, B, C.
b) Coming order is C, B, A.

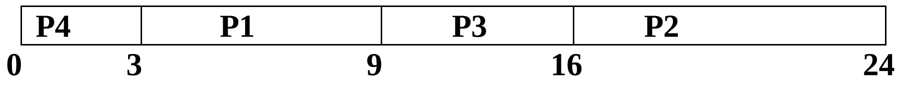
SJF Scheduling Algorithm (non-preemptive)

The Shortest Job First Scheduling Algorithm chooses the process that has the smallest next CPU burst.

Example:
Assume there are 4 ready processes with their next CPU burst time as follows:

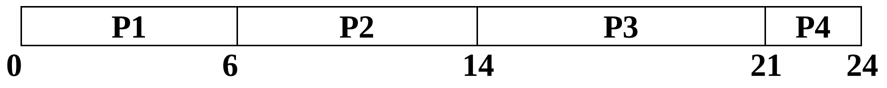
Process	Next CPU burst time
P1	6
P2	8
P3	7
P4	3

Using SJF scheduling algorithm, these processes are scheduled according to the following Gantt Chart:



So, the average waiting time is (3+16+9+0) / 4 =7 ms

NOTE:
If the processes (in order P1, P2, P3, P4) are scheduled using the FCFS algorithm then the average waiting time is (0+6+14+21) /4 = 10.25 ms.



→ SJF minimizes the average wait time because it services smaller processes before large ones.

- The main problem with the SJF algorithm is to know for each process the next CPU burst time!!! However some techniques exist to predict this next CPU burst time.
- Another problem is that starvation of long processes may occur.

Exercise:
Consider the following processes which are in the ready queue in the given order with the relative next CPU bursts.

Process	Next CPU burst time
P1	7
P2	14
P3	3
P4	6

Draw the Gantt charts, fill in the following table:

	Average tat	Average W
FCFS scheduling		
SJF scheduling		

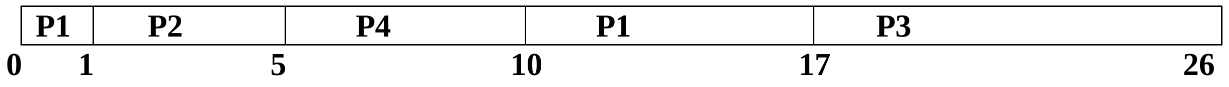
SRTF: Shortest Remaining Time First (preemptive)

This is the preemptive version of SJF. The currently executing process will be preempted from the CPU if a process with a shorter CPU burst time is arrived.

Example:

Consider the following four processes:

Process	Arrival time	CPU burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5



W (P1) = 10-1 = 9 ms
W (P2) = 0 ms
W (P3) = 17-2 = 15 ms
W (P4) = 5-3 = 2 ms

Avg. waiting time:
= (9+0+15+2) / 4
= 26/4
= 6.5 ms

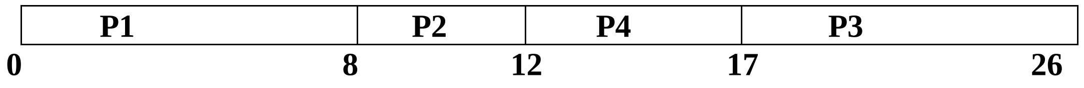
tat (P1) = 17 ms
tat (P2) = 5-1 = 4 ms
tat (P3) = 26-2 = 24 ms
tat (P4) = 10-3 = 7 ms

Average tat = 13 ms

How to process preemptive scheduling:

time	P1	P2	P3	P4
0	8			
1	7	4		
2	7	3	9	
3	7	2	9	5
…				

If the SJF were used which is non-preemptive:



W (P1) = 0 ms **(P2) = 8-1 = 7 ms**
W (P3) = 17-2 = 15 ms **(P4) = 12-3= 9 ms**

Average waiting time = (0+7+15+9) /4 = 31/4 = 7.75 ms

=> SJF gives smaller wait time than FCFS since shorter processes are executed before longer ones!

Exercise:

Consider the following four processes

Process	Arrival time	CPU burst time
P1	0	15
P2	2	2
P3	2	9
P4	10	3

Draw the Gantt charts and fill in the following table:

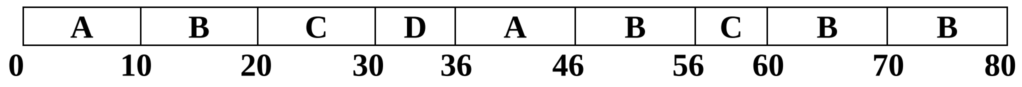
	Average tat	Average W
SJF scheduling		
SRTF scheduling		

Round Robin Scheduling (preemptive)

- This scheduling algorithm is designed specially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes.
- A small unit of time, called a *time quantum*, or time slice, is assigned to each process. Usually a quantum is 10 to 100 ms.

Example:
Consider the following set of process that arrive at time 0 with the order A,B,C,D and the following CPU burst time. Find the average waiting time with RR of quantum : 10 ms

Process	**CPU burst time**
A	**20**
B	**40**
C	**14**
D	**6**

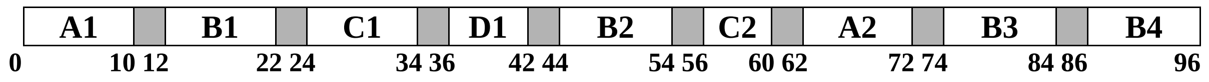


W (A) = 36-10 =26 ms
W (B) = (10-0) + (46-20) + (60-56)= 40 ms
W (C) = (20-0) + (56-30) = 46 ms
W (D) = 30-0 = 30 ms

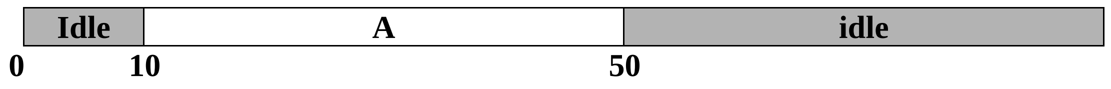
Average waiting time = (26+40+46+30) /4 = 142/4 = 35.5 ms

Example:
Redo the previous example by introducing switching time of 2 ms and some I/O. i.e. A has 10 ms CPU burst – 40 ms I/O burst -10 ms CPU burst , and the others are as before.

CPU:



I/O:



Wait (A) = 62-50 = 12 ms
Wait (B) = 12+22+20+2 = 56 ms
Wait (C) = 24 + 22 = 46 ms
Wait (D) = 36 ms

Average waiting time = 150/4 = 37.5 ms

Important note: smaller time quantum increases the number of context switches! So, time quantum should be large with respect to the context switching time.

Note: if time of quantum = ∞, =>RR becomes FCFS.

Priority scheduling (preemptive and non-preemptive)

The basic principle of the priority scheduling algorithm is to assign a priority order to each ready process. The ready process with the highest priority is allowed to run. Equal priority processes are scheduled randomly. Priority scheduling can be of:

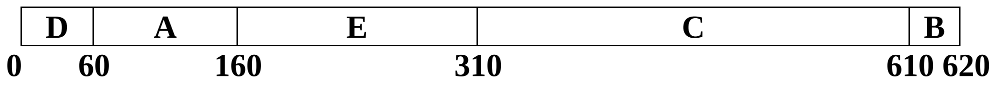
- Non-preemptive (a process runs up to the end of its CPU burst time)
- Preemptive (a running process can be preempted can be preempted by a process with a higher priority which joined the ready queue later)

Example:

Process	Arrival time	CPU burst time	Priority
A	0	100	3
B	0	10	1
C	0	300	3
D	0	60	5
E	80	150	4

Draw the Gantt chart and compute the average waiting time in case of:
a) Non-preemptive
b) Preemptive.

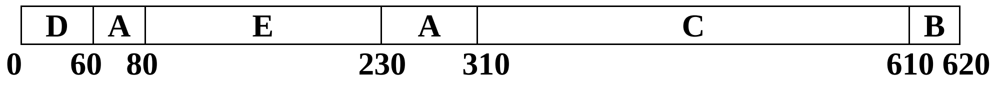
a) Non-preemptive:



W (A) = 60 ms
W (B) = 610 ms
W (C) = 310ms
W (D) = 0 ms
W (E) = 80 ms

=> Average waiting time: 212 ms

a) Preemptive:



Wait (A) = 60+150 = 210 ms
Wait (B) = 610 ms
Wait (C) = 310 ms
Wait (D) = 0 ms
Wait (E) = 0 ms

=> Average waiting time: 226 ms

Note: SJF is a case of the priority scheduling where priority = 1/ (next CPU burst time).

- A major problem with the priority scheduling is **starvation**: a low priority process may wait for ever before being executed. (MIT IBM 7094 crashed in 1973. They found a ready process submitted in 1967 and still waiting for execution).

  Solution: Gradually increase the priority of processes that wait for a long time (this technique is called **aging**).

1. Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time.

Process	**Arrival Time**	**Burst Time**
P1	0.0	8
P2	0.4	4
P3	1.0	1

a) What is the average turnaround time for these processes with the FCFS scheduling algorithm?

P1 arrives and starts executing at 0.0 and terminates at 8, thus turnaround time is (8-0.0) = 8
P2 arrives at 0.4, starts executing at time 8 and terminates at 12, thus turnaround time is (12-0.4) = 11.6
P3 arrives at 1.0, starts executing at 12 and terminates at 13, and thus has a turnaround time is (13-1)=12.
Average turnaround time = (8 + 11.6 + 12)/3 = 10.53

b) What is the average turnaround time for these processes with the SJF (no preemption) scheduling algorithm?

P1 arrives and starts executing at time 0.0 and terminates at 8, thus the turnaround time is (8-0) = 8.
P2 arrives at 0.4, starts executing at 9 (after P3 has executed), and terminates at 13, thus the turnaround time is (13-0.4) = 12.6.
P3 arrives at 1.0, starts executing at 8 and terminates at 9, thus the turnaround time is (9-1) = 8.
Average turnaround time = (8+12.6+8)/3 = 9.53

c) The SJF algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first1 time unit and then SJF scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

Remember that the CPU is left idle for the first 1 time unit.
P1 arrives time 0.0, starts executing at 6 (after P2 and P3 have executed) and terminates at 14, thus the turnaround time is (14-0) = 14.
P2 arrives at 0.4, starts executing at 2.0 (after P3 has executed), and terminates at 6, thus the turnaround time is (6-0.4) = 5.6.
P3 arrives at 1.0, starts executing at 1 (it has the shortest burst and starts executing first) and terminates at 2, thus the turnaround time is (2-1) = 1.
Average turnaround time = (14+5.6+1)/3 = 6.87

d) What is the average turnaround time for these processes with the preemptive SJF (SRTF – shortest remaining time first) scheduling algorithm?

P1 arrives time 0.0,
- **starts executing at 0,**
- **is preempted by P2 at 0.4 with 7.6 t.u. to execute,**
- **resumes only after P2 and P3 have executed at time 5.4,**
- **and terminates at time (5.4+7.6) 13,**
- **thus the turnaround time = 13-0 = 13.**

P2 arrives at 0.4,
- **starts executing at 0.4,**
- **is preempted by P3 at 1 with 3.4 t.u. to execute,**
- **resumes after P3 is done at time 2,**
- **and terminates at (2+3.4) = 5,4,**
- **thus the turnaround time is (5.4 – 0.4) = 5.**

P3 arrives at 1.0, preempts P2 that is executing and starts executing at 1 and terminates at 2, thus the turnaround time is (2-1) = 1.
Average turnaround time = (13+5+1)/3 = 6.33
To be fair to the other algorithms, the number of context switches should be taken into account and added to the above turnaround times. The preemptive SJF will generate the most context switches.

2. Consider three processes P1, P2, and P3 with the following CPU burst times:

Burst times for P1: 14, 12, 17
Burst times for P2: 2, 2, 2, 3, 2, 2, 2, 3
Burst times for P3: 6, 3, 8, 2, 1, 3, 4, 1, 9, 7

All three arrive at time 0, in order P1, P2, P3. Each CPU burst is followed by an I/O operation taking 6 time units, except for the last burst after which the process terminates. Using the provided tables, simulate the execution of these processes using the following scheduling algorithms. Assume that overhead time for process switching and scheduling functions are negligible (assumed to be 0).

a) FCFS
b) SJF (no pre-emption)
c) Preemptive SJF
d) Round robin with a quantum of 5 time units.
e) Round robin with a quantum of 5 time units with priority: P2=P3 > P1.
f) Multi-level feedback queuing using three queues with the following parameters:
    i. Queue 0 – quantum of 2 time units (after which process is moved to Queue 1)
    ii. Queue 1 – quantum of 4 time units (after which process is moved to Queue 2)
    iii. Queue 2 – FCFS
    iv. All process that becomes ready is added to Queue 0.
    v. A process that arrives in Queue 0 preempts any executing process that belongs to either Queue 1 or Queue 2.
    vi. Processes in Queue 1 are allocated to the CPU only when Queue 0 is empty.
    vii. Processes in Queue 2 are allocated to the CPU only with both Queue 0 and Queue 1 are empty.