Process Scheduling

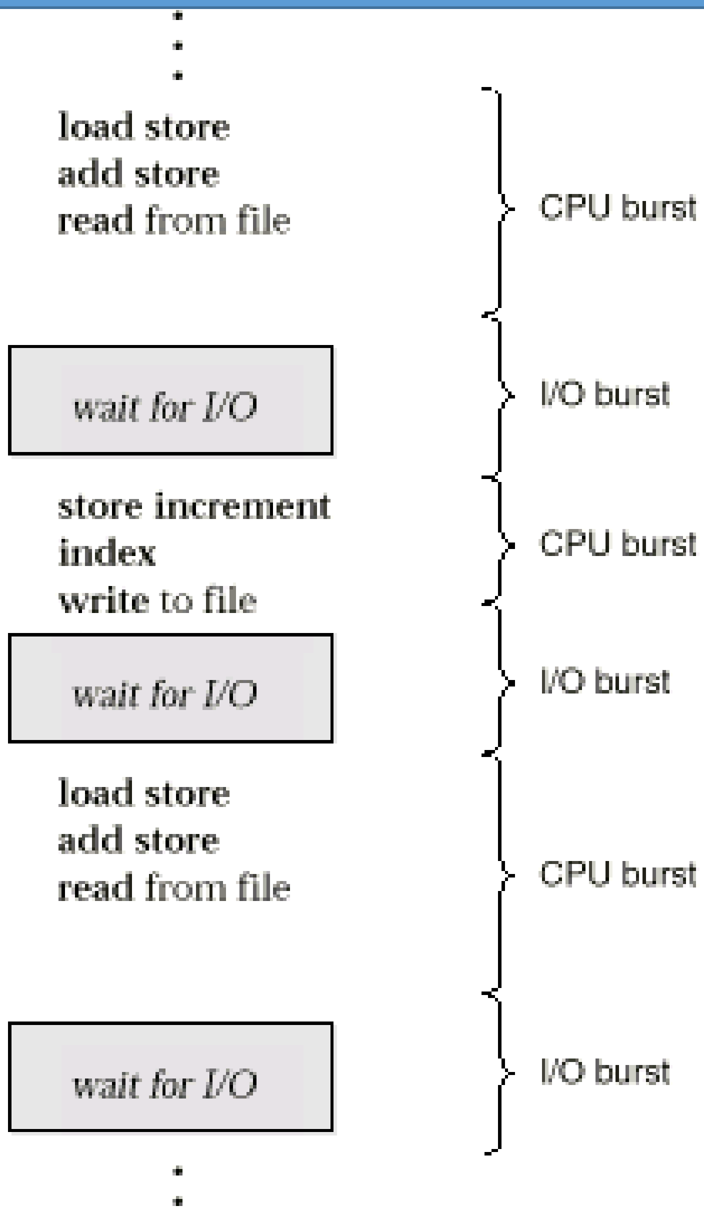
CPU Scheduling

- ***Basic Concepts***
- ***Scheduling Criteria***
- ***Scheduling Algorithms***

Basic Concepts

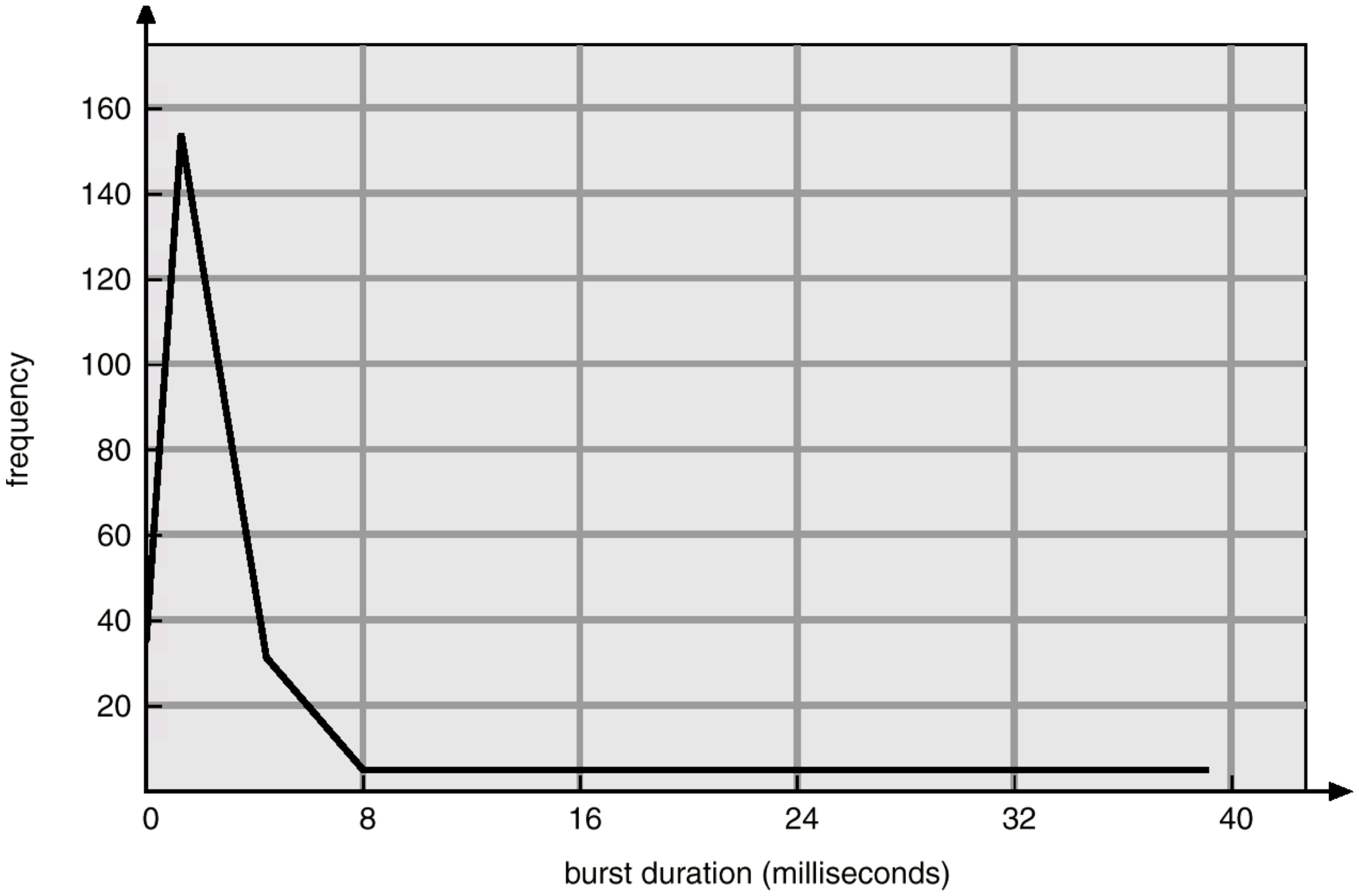
1. *CPU scheduling is the basis of* ***multiprogrammed operating systems.***
2. *Maximum CPU utilization obtained with* ***multiprogramming*** *and can make the computer* ***more productive***.
3. *Objective of multiprogramming is to have* ***some process running all the time*** *to maximize CPU utilization.*

Alternating Sequence of CPU And I/O Bursts



1. *Process execution consists of a cycle of **CPU execution** and **I/O wait**.*
2. ***I/O bound** program typically has many short CPU burst.*
3. *A **CPU-bound** program might have a few long CPU burst.*

Histogram of CPU-burst Times



Basic Concepts

- ***Several processes*** *are kept* ***in memory*** *at one time.*
- *When One process has to wait, the* ***OS takes the CPU away from that process*** *and* ***gives the CPU to another process***.

CPU Scheduler

- *Select one of the process in memory that are ready to execute, and allocates the CPU to one of them.*
- *CPU scheduling decisions may take place when a process:*
  - *Switches from running to waiting state. (I/O request or invocation of wait())*
  - *Switches from running to ready state. (Interrupt occurs)*
  - *Switches from waiting to ready (completion of I/O).*
  - *Process Terminates.*
- *Scheduling under 1 and 4 is* ***nonpreemptive or cooperative****.*
- *All other scheduling is* ***preemptive.***

CPU Scheduler

- ***Preemptive*** *means stopping what you're doing to address something more urgent.*
- ***Non-preemptive*** *means you finish what you're currently doing before switching to something else, regardless of any interruptions.*

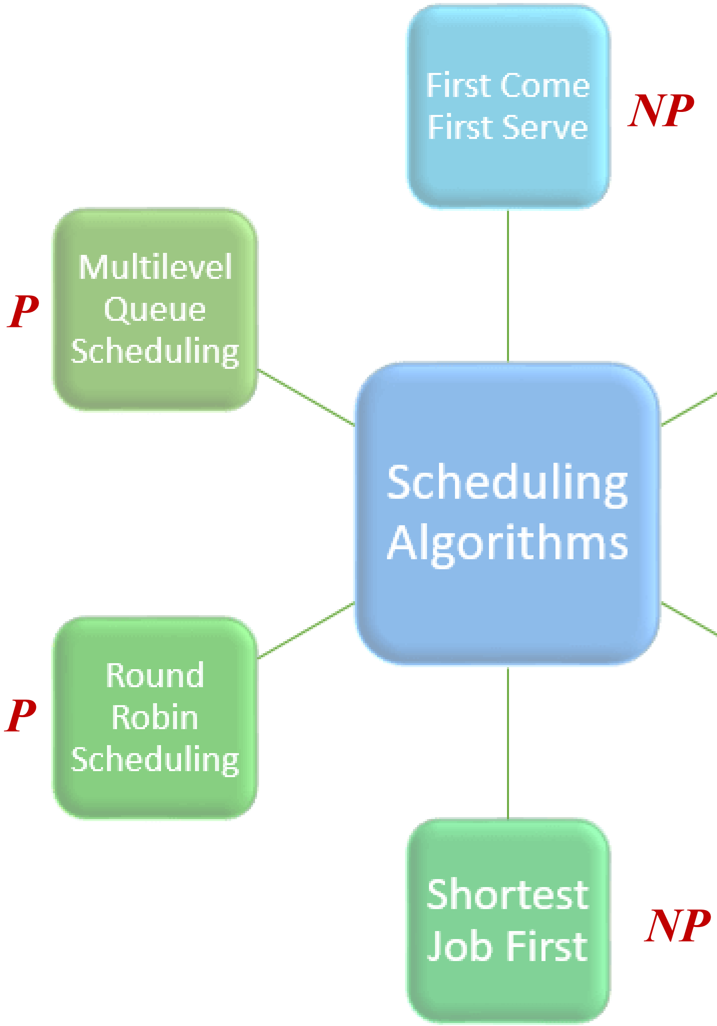
CPU Scheduler

- ***Non-preemptive Scheduling:*** *Once the CPU is allocated to process, the process will keep CPU until it releases the CPU either by terminating or by switching to the waiting state.*
- ***Preemptive Scheduling:*** *The executing process in preemptive scheduling is interrupted in the middle of execution process switches from the running state to the ready state or from the waiting state to the ready state, when a higher priority one comes. Allocated to the process for a limited amount of time.*

Dispatcher

- ***Dispatcher*** *module gives control of the CPU to the process selected by the short-term scheduler; this involves:*
  - *switching context*
  - *switching to user mode*
  - *jumping to the proper location in the user program to restart that program*
- ***Dispatch latency*** *– Time it takes for the dispatcher to stop one process and start another running.*

CPU Scheduling Algorithms



Scheduling Criteria

1. **Burst time/execution time/running time**: *The time process require for running on CPU.*

2. **Arrival time:** *Time at which a process enters the ready queue.*

3. **Exit time:** *When process completes execution and exit from the memory.*

4. **Turnaround time:** *Total time spend in system.*

   TAT = E.T – A.T = B.T + W.T

5. **Waiting time:** *Waiting time is the amount of time spent by a process waiting in the ready queue for getting the CPU*

   Waiting time = Turn around time – CPU burst time

6. **Response time:** *Response time is the amount of time after which a process gets the CPU for the first time after entering the ready queue.*

   Response Time = Time at which process first gets the CPU – Arrival time

Scheduling Criteria

7. ***CPU utilization*** *– keep the CPU as busy as possible*

CPU Utilization = Total CPU busy time/Total time required to execute process

8. ***Throughput*** *– No of processes that complete their execution per time unit*

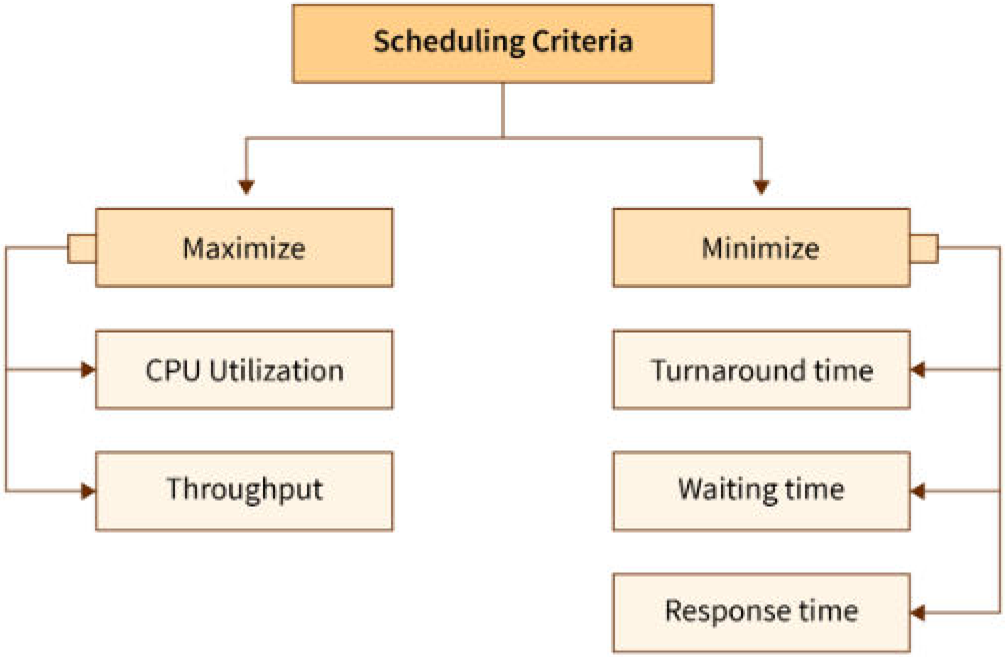
Throughput = total number of processes /(Max exit time – min arrival time)

9. ***Response Ratio -***

Response Ratio = (Waiting Time + Burst time) / Burst time

Optimization Criteria

1. *Max CPU utilization*
2. *Max throughput*
3. *Min turnaround time*
4. *Min waiting time*
5. *Min response time*



First-Come, First-Served (FCFS) Scheduling

First-Come, First-Served (FCFS) Scheduling

- *It is the simplest scheduling algorithm.*
- *The process which will request the CPU first will get the CPU First.*
- *When a process is entered into the ready queue its PCB is Linked onto the tail of the FIFO queue.*
- *Easy to understand and easily implemented by FIFO queue data structure.*
- *Always non-preemptive in nature.*

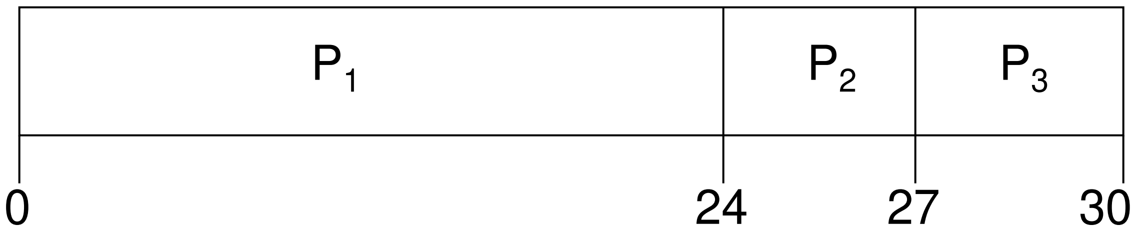
First-Come, First-Served (FCFS) Scheduling

Example:

Process	***Burst Time***	***AT***
P_1	24	0
P_2	3	0
P_3	3	0

Suppose that the processes arrive in the order: P_1 , P_2 , P_3.

First-Come, First-Served (FCFS) Scheduling



- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- ***Average waiting time:*** $(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order **P_2 , P_3 , P_1**

- *The Gantt chart for the schedule is:*

Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$

- ***Average waiting time: (6 + 0 + 3)/3 = 3:*** *Much better than previous case.*

FCFS Scheduling - Problem 1

Process	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Gantt Chart

P4	P3	P1	P5	P2

0 3 11 17 21 23

Problem 1

P4	P3	P1	P5	P2	
0	3	11	17	21	23

Process	Burst time	Arrival time	Exit Time	Waiting Time	TAT	Response Time
P1	6	2	17	9	15	9
P2	2	5	23	16	18	16
P3	8	1	11	2	10	2
P4	3	0	3	0	3	0
P5	4	4	21	13	17	13

A.W.T = (9+16+2+0+13)/5 = 8

Problem 2

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	***Arrival time***	***Burst time***
P1	*3*	*4*
P2	*5*	*3*
P3	*0*	*2*
P4	*5*	*1*
P5	*4*	*3*

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Ready –

P3		P1	P5	P2	P4

0 2 3 7 10 13 14

Problem 2

P3		P1	P5	P2	P4

0 2 3 7 10 13 14

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 4 = 0
P2	13	13 – 5 = 8	8 – 3 = 5
P3	2	2 – 0 = 2	2 – 2 = 0
P4	14	14 – 5 = 9	9 – 1 = 8
P5	10	10 – 4 = 6	6 – 3 = 3

Now,

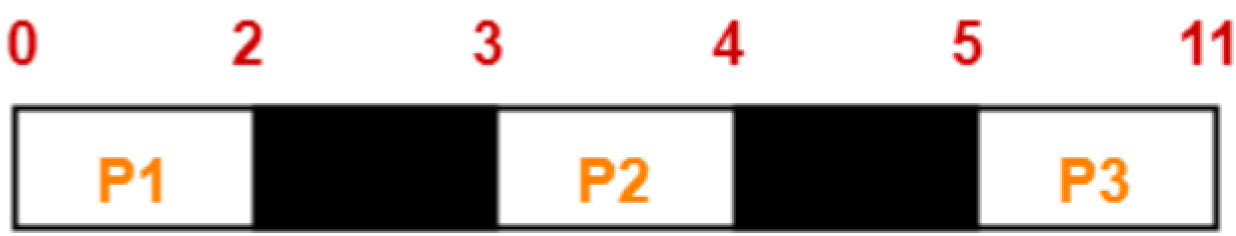
- Average Turn Around time = (4 + 8 + 2 + 9 + 6) / 5 = 29 / 5 = 5.8 unit
- Average waiting time = (0 + 5 + 0 + 8 + 3) / 5 = 16 / 5 = 3.2 unit

Problem 3

Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	*Arrival time*	*Burst time*
P1	*0*	*2*
P2	*3*	*1*
P3	*5*	*6*

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.



Gantt Chart

Problem 3

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	2	2 – 0 = 2	2 – 2 = 0
P2	4	4 – 3 = 1	1 – 1 = 0
P3	11	11- 5 = 6	6 – 6 = 0

Now,

- Average Turn Around time = (2 + 1 + 6) / 3 = 9 / 3 = 3 unit
- Average waiting time = (0 + 0 + 0) / 3 = 0 / 3 = 0 unit

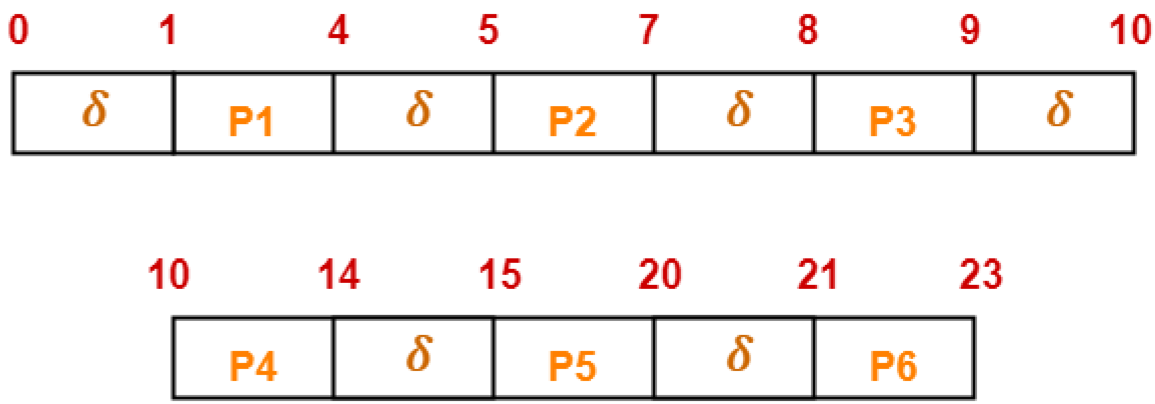
Problem 4

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	3
P2	1	2
P3	2	1
P4	3	4
P5	4	5
P6	5	2

If the CPU scheduling policy is FCFS and there is 1 unit of overhead in scheduling the processes, find the efficiency of the algorithm.

Problem 4



Gantt Chart

Now,

- Useless time / Wasted time = 6 x δ = 6 x 1 = 6 unit
- Total time = 23 unit
- Useful time = 23 unit – 6 unit = 17 unit

Efficiency (η)

= Useful time / Total Total

= 17 unit / 23 unit

= 0.7391

FCFS Scheduling (Cont.)

Convoy effect :

- *Consider processes with higher burst time arrived before the processes with smaller burst time.*
- *Then, smaller processes have to wait for a long time for longer processes to release the CPU.*
- *This effect results in lower CPU and device utilization.*

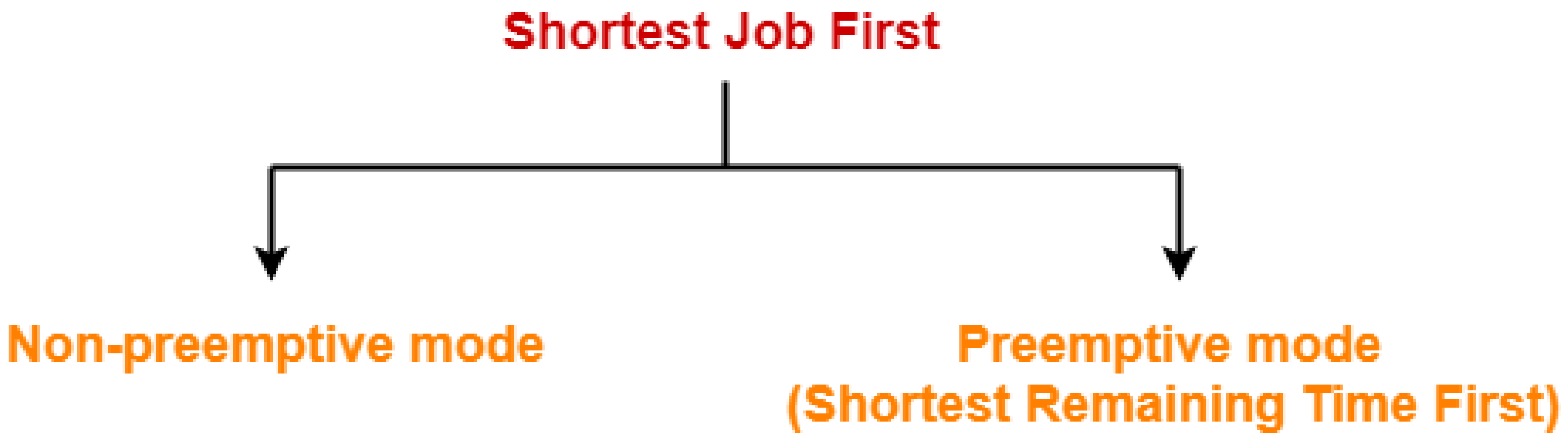
FCFS Scheduling (Cont.)

- ***Advantages:*** *Simple, easy to use, easy to understand, easy to implement, must be used for background processes where execution is not urgent.*
- ***Disadvantages:*** *Suffer from convoy effect, normally higher average waiting time, no consideration of priority and burst time, should not be used for interactive systems.*
- ***No starvation, only convoy effect.***

Shortest-Job-First (SJF) Scheduling

Shortest-Job-First (SJF) Scheduling

- *Out of all available process, CPU is assigned to the process having* ***small burst time requirement****.*
- *If there is a tie, FCFS is used to break the tie.*



Shortest-Job-First (SJF) Scheduling

- ***Two schemes:***
  - ***Nonpreemptive*** – *Once CPU given to the process it cannot be preempted until completes its CPU burst.*
  - ***Preemptive*** – *If a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as* ***Shortest – remaining – time - first (SRTF)****.*

Problem1

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	***Arrival time***	***Burst time***
P1	*3*	*1*
P2	*1*	*4*
P3	*4*	*2*
P4	*0*	*6*
P5	*2*	*3*

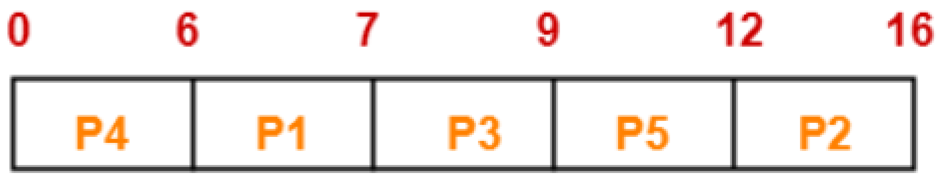
If the CPU scheduling policy is ***SJF non-preemptive****, calculate the average waiting time and average turn around time.*

Ready-

P4	P1	P3	P5	P2

0 6 7 9 12 16

Problem 1



Gantt Chart

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

- Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
- Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

Problem 2 (SRTF)

Process Id	*Arrival time*	*Burst time*
P1	*3*	*1/0*
P2	*1*	*4/3/2/0*
P3	*4*	*2/0*
P4	*0*	*6/5/0*
P5	*2*	*3/0*

Ready-

P4	P2	P1	P2	P3	P5	P4

0 1 3 4 6 8 11 16

Waiting time - P1 = 0, P2 = 0+1=1, P3= 2, P4 = 10, P5 =6

SRTF Scheduling

P4	P2	P1	P2	P3	P5	P4

0 1 3 4 6 8 11 16

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 3 = 1	1 – 1 = 0
P2	6	6 – 1 = 5	5 – 4 = 1
P3	8	8 – 4 = 4	4 – 2 = 2
P4	16	16 – 0 = 16	16 – 6 = 10
P5	11	11 – 2 = 9	9 – 3 = 6

Now,

- Average Turn Around time = (1 + 5 + 4 + 16 + 9) / 5 = 35 / 5 = 7 unit
- Average waiting time = (0 + 1 + 2 + 10 + 6) / 5 = 19 / 5 = 3.8 unit

Shortest-Job-First (SJR) Scheduling

- ***SRTF is optimal*** *– Gives* ***minimum average waiting time*** *for a given set of processes. Greedy about CPU Burst.*
- *The real difficulty with SJF algorithm is* ***knowing the size of the next CPU burst.***

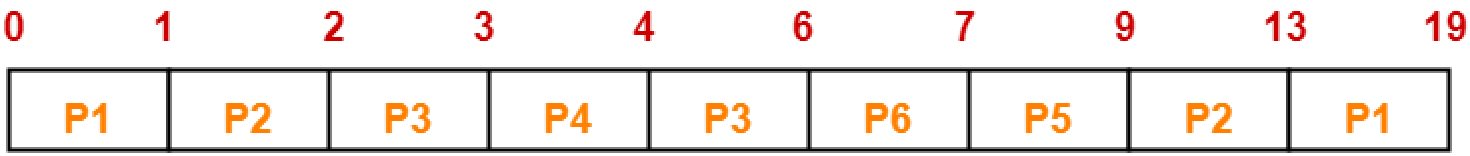
SRTF Scheduling

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time and average turn around time if we have 2 CPUs.

SRTF Scheduling



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	19	19 – 0 = 19	19 – 7 = 12
P2	13	13 – 1 = 12	12 – 5 = 7
P3	6	6 – 2 = 4	4 – 3 = 1
P4	4	4 – 3 = 1	1 – 1 = 0
P5	9	9 – 4 = 5	5 – 2 = 3
P6	7	7 – 5 = 2	2 – 1 = 1

Now,

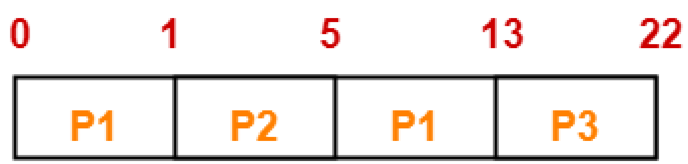
- Average Turn Around time = (19 + 12 + 4 + 1 + 5 + 2) / 6 = 43 / 6 = 7.17 unit
- Average waiting time = (12 + 7 + 1 + 0 + 3 + 1) / 6 = 24 / 6 = 4 unit

SRTF Scheduling

Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	9
P2	1	4
P3	2	9

If the CPU scheduling policy is SRTF, calculate the average waiting time and average turn around time.



Gantt Chart

SRTF Scheduling

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	13	13 – 0 = 13	13 – 9 = 4
P2	5	5 – 1 = 4	4 – 4 = 0
P3	22	22- 2 = 20	20 – 9 = 11

Now,

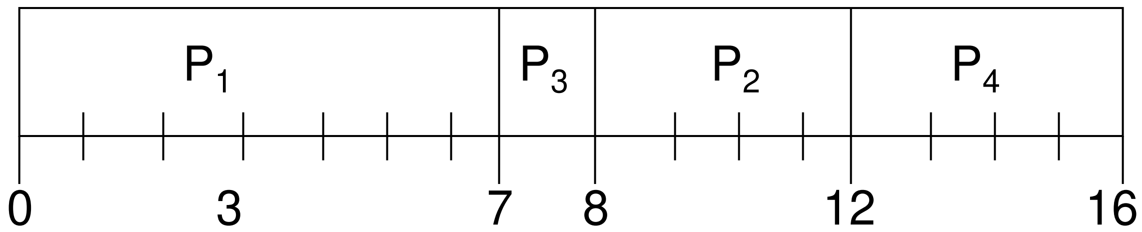
- Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = 12.33 unit
- Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = 5 unit

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

Example of Non-Preemptive SJF

- *SJF (non-preemptive)*



- *Average waiting time = (0 + 6 + 3 + 7)/4 - 4*

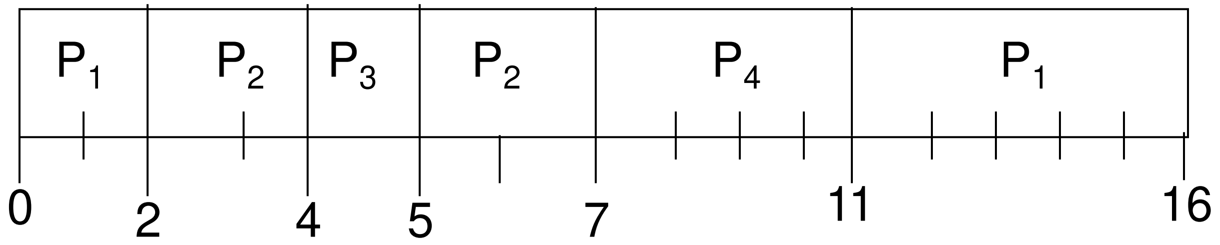
Example of Preemptive SJF

Process	*Arrival Time*	*Burst Time*
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- *SJF (preemptive)*

Example of Preemptive SJF

- *SJF (preemptive)*



- *Average waiting time = (9 + 1 + 0 +2)/4 = 3*

Example of Preemptive SJF

Consider the set of 4 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	20
P2	15	25
P3	30	10
P4	45	15

If the CPU scheduling policy is SRTF, calculate the waiting time of process P2.

Gantt Chart

Thus,

- Turn Around Time of process P2 = 55 – 15 = 40 unit
- Waiting time of process P2 = 40 – 25 = 15 unit

Determining Length of Next CPU Burst

- *Next CPU Burst is generally predicted using exponential averaging of previous CPU bursts.*

1. t_n = actual lenght of n^{th} CPU burst
2. τ_n = predicted value for the n^{th} CPU burst
3. τ_{n+1} = predicted value for the next CPU burst
4. α, smoothening factor, $0 \leq \alpha \leq 1$

Exponential Average is:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\tau_n.$$

Examples of Exponential Averaging

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n .$$

- $\alpha = 0$

  $\tau_{n+1} = \tau_n$

  Recent history does not count.

- $\alpha = 1$

  $\tau_{n+1} = t_n$

  Only the actual last CPU burst counts.

- $\alpha = 1/2$

  Recent history and past history are equally weighted.

- τ_0 *can be defined as a constant or as an overall system average.*

Examples of Exponential Averaging

- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\ t_n + (1 - \alpha)\ \alpha\ t_{n-1} + \ldots\ldots + (1 - \alpha)^j\ \alpha\ t_{n-j} + \ldots + u\ (1 - \alpha)^{n+1}\ \tau_0$$

- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor.

Examples of Exponential Averaging

*Calculate the predicted burst time using exponential averaging for the fifth process if the predicted burst time for the first process is **10 units** and actual burst time of the first four processes is **4, 8, 6 and 7** units respectively. Given **α = 0.5**.*

Given-

- Predicted burst time for 1st process = 10 units
- Actual burst time of the first four processes = 4, 8, 6, 7
- α = 0.5

Predicted burst time for 2nd process

= α x Actual burst time of 1st process + (1-α) x Predicted burst time for 1st process

= 0.5 x 4 + 0.5 x 10

= 2 + 5

= 7 units

Examples of Exponential Averaging

Predicted burst time for 3rd process

= α x Actual burst time of 2nd process + (1-α) x Predicted burst time for 2nd process

= 0.5 x 8 + 0.5 x 7

= 4 + 3.5

= 7.5 units

Predicted burst time for 4th process

= α x Actual burst time of 3rd process + (1-α) x Predicted burst time for 3rd process

= 0.5 x 6 + 0.5 x 7.5

= 3 + 3.75

= 6.75 units

Examples of Exponential Averaging

Predicted burst time for 5th process

= α x Actual burst time of 4th process + (1-α) x Predicted burst time for 4th process

= 0.5 x 7 + 0.5 x 6.75

= 3.5 + 3.375

= 6.875 units

SJF/SRTF

Advantages -

- *SRTF is optimal and guarantees the minimum average waiting time.*
- *It provides a standard for other algorithms since no other algorithm performs better than it.*

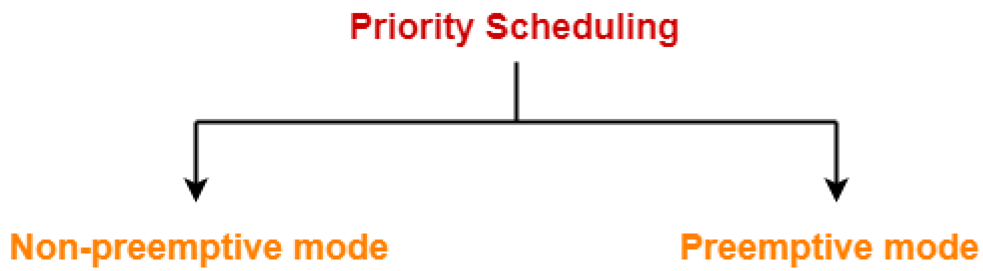
Disadvantages -

- *It can not be implemented practically since burst time of the processes can not be known in advance.*
- *It leads to starvation for processes with larger burst time.*
- *Priorities can not be set for the processes.*
- *Processes with larger burst time have poor response time.*

Priority Scheduling

Priority Scheduling

- *A **priority** (integer number) is associated with each process.*
- *Out of all the available processes, CPU is assigned to the process having the **highest priority** (smallest integer ≡ highest priority or highest integer ≡ highest priority)..*
- *In case of a tie, it is broken by [**FCFS Scheduling**].*
- *No importance to arrival time and burst time.*
- *Priority Scheduling can be used in both **preemptive** and **non-preemptive mode**.*



Preemptive priority Scheduling

Preemptive Priority scheduling: *CPU will be preempt if priority of the newly arrived process is higher than the priority of currently running process.*

Priority Scheduling

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

*If the CPU scheduling policy is **priority non-preemptive**, calculate the average waiting time and average turn around time. (Higher number represents higher priority).*

Ready –

Priority Scheduling

P1	P4	P5	P3	P2

0 4 9 11 12 15

Process Id	Exit time	Turn Around time	Waiting time
P1	4	4 – 0 = 4	4 – 4 = 0
P2	15	15 – 1 = 14	14 – 3 = 11
P3	12	12 – 2 = 10	10 – 1 = 9
P4	9	9 – 3 = 6	6 – 5 = 1
P5	11	11 – 4 = 7	7 – 2 = 5

Now,

- Average Turn Around time = (4 + 14 + 10 + 6 + 7) / 5 = 41 / 5 = 8.2 unit
- Average waiting time = (0 + 11 + 9 + 1 + 5) / 5 = 26 / 5 = 5.2 unit

Preemptive Priority Scheduling

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

If the CPU scheduling policy is ***priority preemptive****, calculate the average waiting time and average turn around time. (Higher number represents higher priority).*

Ready – *Running –* *Terminated-*

P1	P2	P3	P4	P5	P2	P1

0 1 2 3 8 10 12 15

Preemptive Priority Scheduling

P1	P4	P5	P3	P2

0 4 9 11 12 15

Process Id	Exit time	Turn Around time	Waiting time
P1	15	15 – 0 = 15	15 – 4 = 11
P2	12	12 – 1 = 11	11 – 3 = 8
P3	3	3 – 2 = 1	1 – 1 = 0
P4	8	8 – 3 = 5	5 – 5 = 0
P5	10	10 – 4 = 6	6 – 2 = 4

- Average Turn Around time = (15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = 7.6 unit
- Average waiting time = (11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = 4.6 unit

Priority Scheduling

- ***Blocked Process:*** *Process is ready to run but waiting for the CPU.*
- ***Major Problem*** *:* ***Indefinite blocking or starvation***.
- *Priority scheduling algorithm can leave some low priority processes waiting indefinitely.*
- *In a heavily loaded computer system a higher priority process can prevent low priority process from ever getting the CPU.*

Priority Scheduling

- ***Solution*** $\equiv$ ***Aging***
- *Aging ensures that processes with lower priority will eventually complete their execution.*
- *By gradually increasing the priority of processes that wait in the system for long time.*

Priority Scheduling

Advantages-

- *It considers the priority of the processes and allows the important processes to run first specially for system processes.*
- *Priority scheduling in preemptive mode is best suited for real time operating system.*

Disadvantages-

- *Processes with lesser priority may **starve** for CPU.*
- *There is no idea of response time and waiting time.*

Round Robin (RR)

Round Robin (RR)

- *This algorithm is designed for* ***time sharing system*** *where it is not necessary to complete one process and then start another process, but to be* ***responsive.***
- *CPU is assigned to the process on the basis of* ***FCFS*** *for a fixed amount of time in one go. Within which either the process will terminate and releases the CPU or after time quantum expires, running process will be* ***preempted*** *and sent to the ready queue and there it will wait for next chance, ready queue will be a* ***circular queue****. The processor is assigned to the next arrived process.*

Round Robin (RR)

- *This fixed amount of time is called as* ***time quantum*** *or time slice.*
- *Round robin is always* ***preemptive*** *in nature.*

Round Robin (RR)

- *Each process gets a small unit of* ***CPU time (time quantum),*** *usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.*

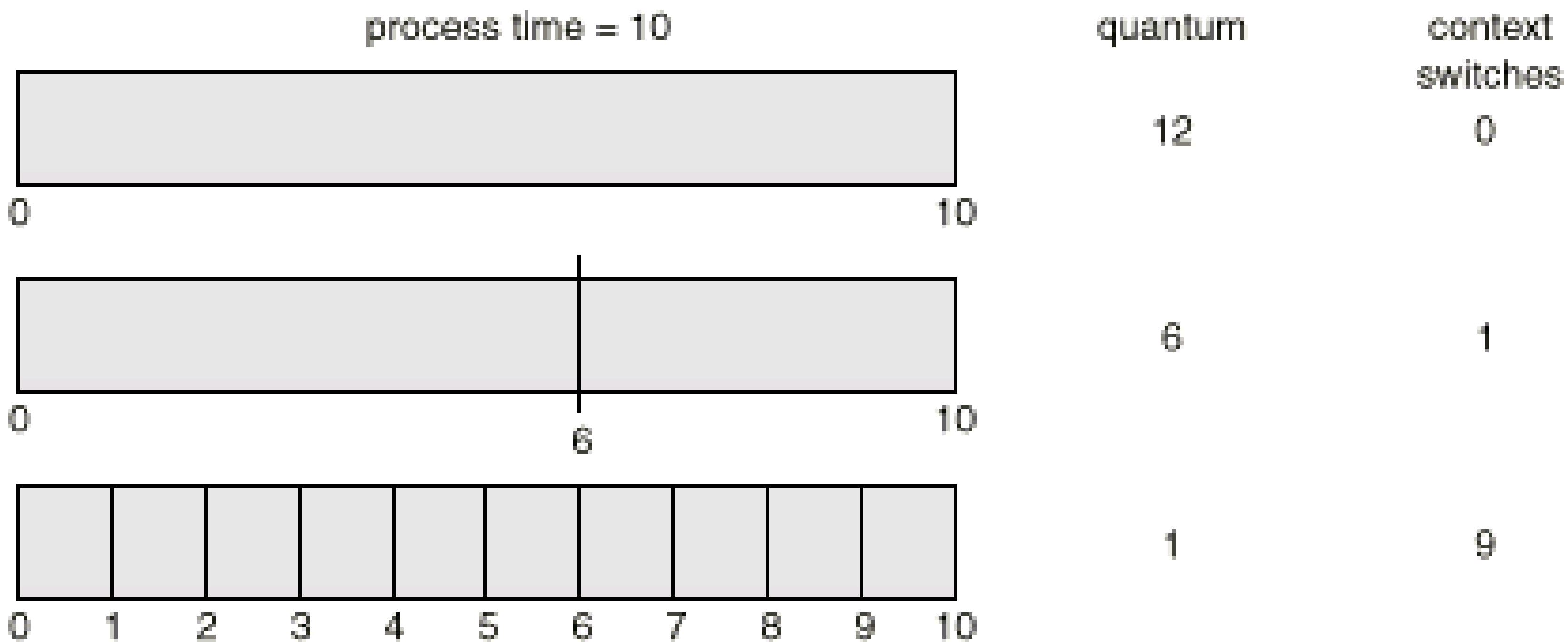
- *If there are n processes in the ready queue and the time quantum is q, then each process gets* ***1/n of the CPU time*** *in chunks of at most q time units at once. No process waits more than* ***(n-1)q*** *time units.*

Round Robin (RR)

- ***Performance***
  - *q large $\Rightarrow$ FCFS*
  - *q small $\Rightarrow$ Large number of context switches.*
  - *Thus the time quantum (q) must be large with respect to context switch, otherwise overhead is too high, and it should not be too large.*

- ***Rule of thumb is that 80% of CPU bursts should be shorter than the time quantum.***

How a Smaller Time Quantum Increases Context Switches



Thus we want the time quantum to be large with respect to the context switch time.

Round Robin (RR)

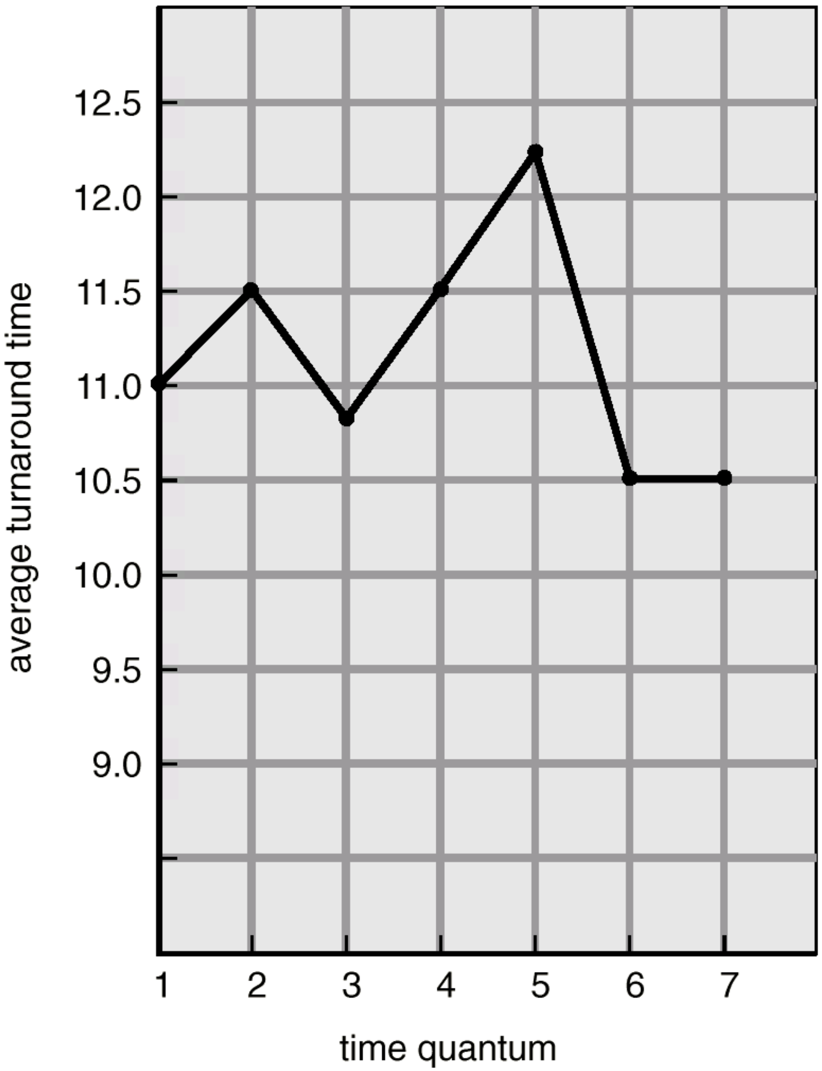
Advantages

- *Perform well in terms of avg. response time.*
- *Works well in time sharing systems, client server architecture and interactive system.*

Disadvantages

- *Performance depends heavily on time quantum.*
- *No idea of priority.*

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Problem 1

There are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is ***4 units****. Schedule processes using RR algorithm.*

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

Ready Queue – *Running –* *Terminate-*

P1	**P2**	**P3**	**P4**	**P5**	**P1**	**P6**	**P2**	**P5**	
0	4	8	11	12	16	17	21	23	24

Problem 1

There are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is ***4 units****. Schedule processes using RR algorithm.*

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	6
P3	2	3
P4	3	1
P5	4	5
P6	6	4

Ready Queue – P3,P4,P5,P1,P6 *Running –* *Terminate-*

P1	P2	P3	P4	P5	P1	P6	P2	P5	
0	4	8	11	12	16	17	21	23	24

Problem 1

P1	P2	P3	P4	P5	P1	P6	P2	P5	
0	4	8	11	12	16	17	21	23	24

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	0	5	17	17	12
2	1	6	23	22	16
3	2	3	11	9	6
4	3	1	12	9	8
5	4	5	24	20	15
6	6	4	21	15	11

Example 2: RR with Time Quantum = 20

Process	*Burst Time*
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 20 37 57 77 97 117 121 134 154 162

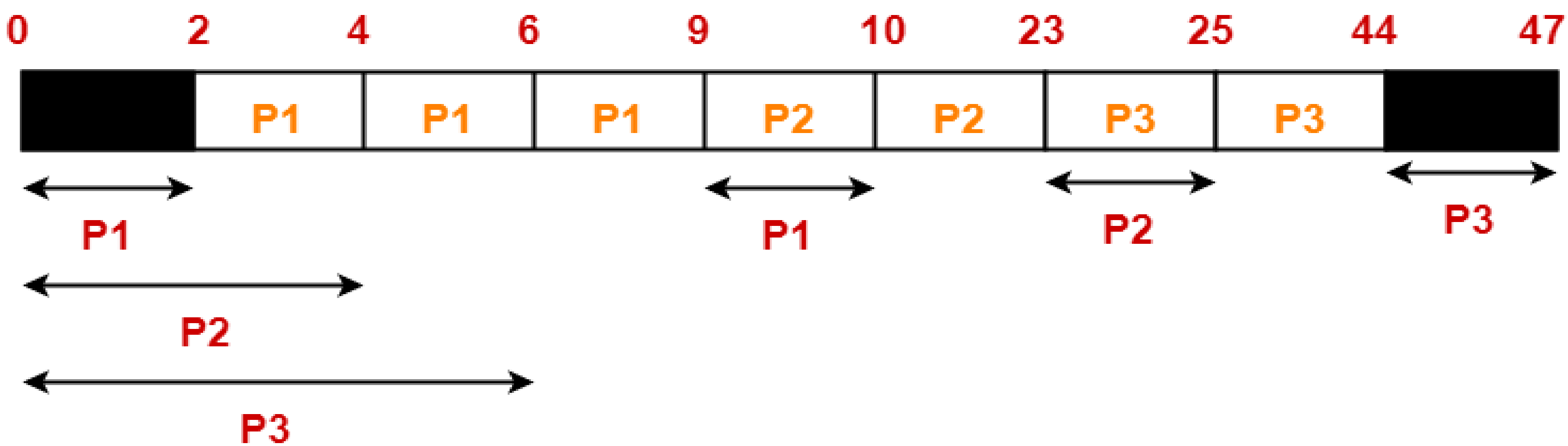
- Typically, higher average turnaround than SJF, but better *response*.

Example3:

Q: Consider three processes, all arriving at time zero, with total execution time of ***10, 20 and 30*** units, respectively. Each process spends the first ***20%*** of execution time doing I/O, the next ***70%*** of time doing computation, and the last ***10%*** of time doing I/O again. The operating system uses a ***shortest remaining time first*** scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle?

Process	***A.T***	***Process time***	***i/o, cpu time, i/o time***
P1	*0*	*10*	*2, 7, 1*
P2	*0*	*20*	*4, 14 , 2*
P3	*0*	*30*	*6, 21, 3*

Process	A.T	Process time	i/o, cpu time, i/o time
P1	0	10	2, 7, 1
P2	0	20	4, 14 , 2
P3	0	30	6, 21, 3



Gantt Chart

- ***Exit time:***

  P1- 10, P2- 25, P3- 47

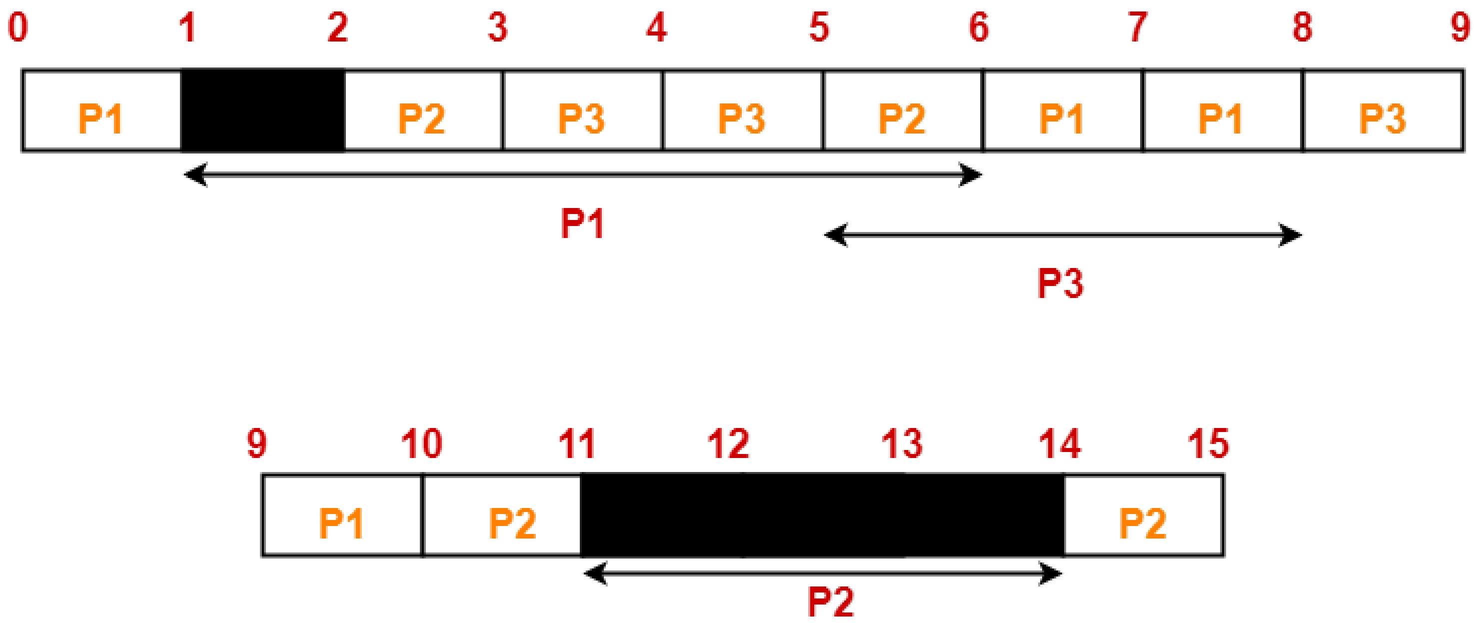
- ***CPU Utilization:*** ***(42/47)*100***

Example 4:

Q4: *Consider the set of* ***3 processes*** *whose arrival time and burst time are given below- If the CPU scheduling policy is* ***Preemptive Priority Scheduling****, calculate the average waiting time and average turn around time. (Lower number means higher priority)*

Process No.	Arrival Time	Priority	Burst Time		
			CPU Burst	I/O Burst	CPU Burst
P1	0	2	1	5	3
P2	2	3	3	3	1
P3	3	1	2	3	1

Example:



Gantt Chart

Example 5:

If the CPU scheduling policy is ***Shortest Remaining Time First****, calculate the average waiting time and average turn around time*

Process No.	***Arrival Time***	***Burst Time***		
		CPU Burst	***I/O Burst***	***CPU Burst***
P1	*0*	*3*	*2*	*2*
P2	*0*	*2*	*4*	*1*
P3	*2*	*1*	*3*	*2*
P4	*5*	*2*	*2*	*1*

Example 5:

If the CPU scheduling policy is ***Shortest Remaining Time First****, calculate the average waiting time and average turn around time*

Process No.	***Arrival Time***	***Burst Time***		
		CPU Burst	***I/O Burst***	***CPU Burst***
P1	*0*	*3*	*2*	*2*
P2	*0*	*2*	*4*	*1*
P3	*2*	*1*	*3*	*2*
P4	*5*	*2*	*2*	*1*

P2	P3	P1	P2	P4	P3	P4	P1

0 2 3 6 7 9 11 12 14