Lecture 5: CPU Scheduling

multiprogramming system CPU utilization max করার help করে.

- CPU scheduling is the basis for multi-programming OS

multiprogramming এ আমরা process ready Queue তে রাখা করে, But এটা certain time processor এ একটা single process execute হয়, So, ready Queue তে process অনেক আছে সেটা select করে নিতে হয় কোন process টা কোন সময় processor execution হবে, এই selection process কেই CPU Scheduling

Continuous Cycle:

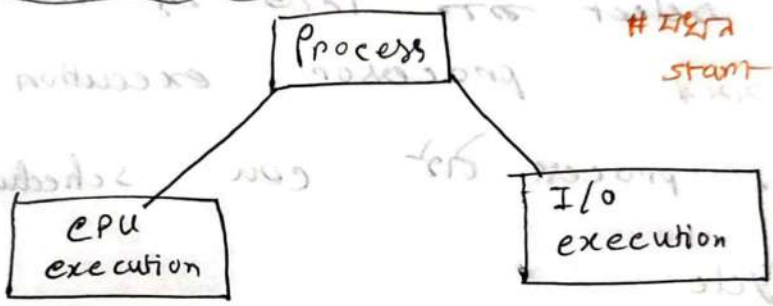
- One process has to wait (I/O)
- OS takes the CPU away
- Give CPU to another process
- This pattern continues

• CPU - I/O Burst Cycle

Process execution consists of a cycle of CPU execution & I/O wait.

- The objective of multiprogramming is to have some process running at all times, to maximize CPU Utilization
- With multiprogramming, 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 - I/O Burst Cycle:



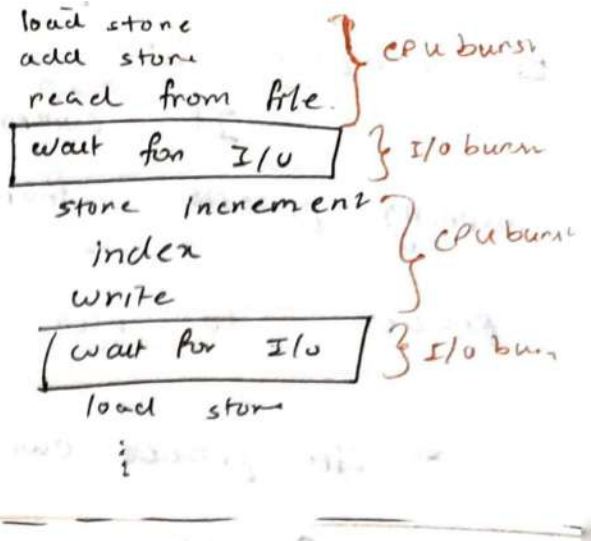
- Almost all processes alternate between two states in a continuing cycle, as shown in figure below:
  - A CPU burst of performing calculations, and
  - A I/O burst of waiting for data transfer in or out of the system
- Process alternate between this two states
- CPU burst distribution is of main concern

CPU burst:

একটা প্রোসেস CPU তে যে time spend করে।

I/O burst:

একটা প্রোসেস I/O তে যে time spend করে



Process starts and ends with CPU burst.

As আমরা process load, store ..... , exit করতে CPU দিয়ে, Terminate() code execute করার পরে CPU নামবে,

Preemptive and Non Preemptive Scheduling:

Non preemptive:

- Not forcefully
- Once CPU given to a process it can't be preempted unhill completes it's CPU burst or finish it's work.
- The process isn't interrupted unhill it's life cycle is complete
- FCFS (First Come first Serve) preemptive.

FCFS → Non preemptive

SJF → Preemptive (SRTF)
↳ Non preemptive

Priority → Preemptive
↳ Non preemptive

RR → Preemptive

Preemptive - (multitasking)

It comes from the ability to remove a running process from CPU and allow another process to run in CPU.

- The process can be interrupted, even before the completion
- Preempted process moves to ready queue
- Preemptive scheduling is a CPU scheduling technique that works by dividing time slots of CPU to a given process. The time slot given might be able complete the whole process or might not be able to do so
- Algorithms
  - RR (Round Robin)

CPU Scheduler :

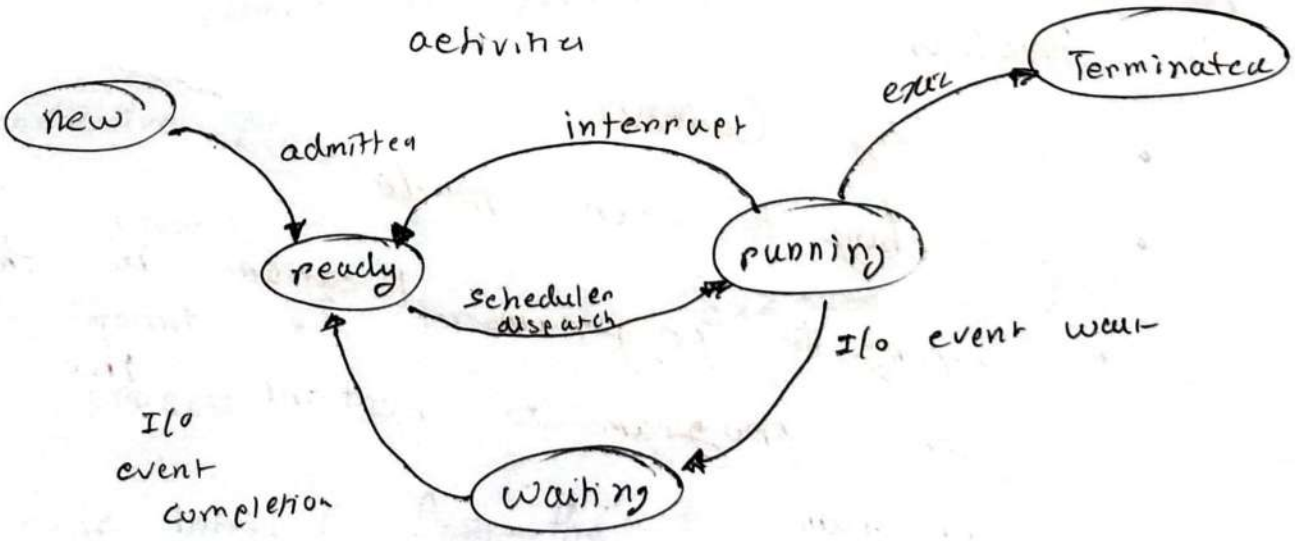
- Selects the processes from ready queue and allocate CPU to them
  - FIFO Queue
  - Priority Queue
  - Unordered linked list
  - Tree

Who → short term scheduler
Where → ready to run &
When → for 4 scenarios

- CPU scheduling decisions may take place when a process :
  - When a process running to waiting state (I/O request)
  - running → ready (Interrupt occurs)
  - Waiting → ready (after completion of I/O)
  - Terminates.

non preemptive.

- Preemptive approach :
  - consider access to shared data
  - " preemption while in kernel mode
  - " interrupts occuring during crucial OS activities



Dispatcher

• Ready থেকে run এ নেয়াতে কাজ করে,

CPU scheduler এর কাজ to select a process from ready queue; which will be executed next by the CPU.

↳ main mem এর ready queue থেকে process select করবে।

Dispatcher এর কাজ: Currently যে process টি CPU তে execute হচ্ছে তাকে সরিয়ে main memory তে নিয়ে আসা। এবং CPU scheduler যখন process select করবে তখন process CPU এর নিয়ে যাওয়া।

- Gives control of CPU to the selected process which was selected by CPU scheduler it involves
  - switching Context
  - switching to user mode
  - jumping to program 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.

Scheduling Criteria

যখন CPU scheduling করা হচ্ছে তখন নিম্নের criteria গুলো follow করা must

- CPU Utilization
- Throughput
- Turnaround Time.
- Waiting time.
- Response time
- Fairness

(এই process করতে কত system wait করছে & execute করছে।)

CPU Utilization : Keeping the CPU as busy as possible

Throughput: Number of processes that complete their execution per unit time.

Turnaround time.

- Amount of time to execute a particular process
- The interval from the time of submission of a process to the time of the completion

- Sum of the periods spent waiting to get into memory, waiting in the ready ~~que~~ queue, executing in the CPU, doing I/O

Waiting Time.

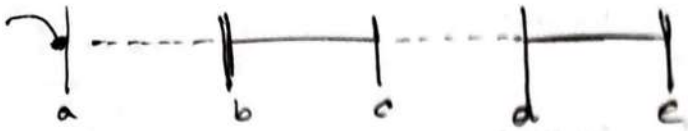
- Amount of time a process has been waiting in the ready queue

Response Time.

- amount of time it takes when a request was submitted untill the first response is produced, not output (for time sharing environment)

Fairness.

Give each process a fair amount of CPU



Waiting time = (b - a) + (d - c).

Response time = (b - a)

Turnaround time = ~~(c - b) + (e - d)~~ (e - a)

Scheduling Algorithm Optimization Criteria

- Max CPU utilization.
- Max throughput.
- Min turnaround time
- Min waiting time
- Min response time

Scheduling Algorithms

- FCFS Scheduling
- SJF Schedulin.
  - ↳ SRTF
- Priority Scheduling
  - ↳ Non preemptive.
  - ↳ preemptive
- Round Robin
- Multilevel Queue.
- Multilevel feedback Queue

FCFS CPU Scheduling

⇒ FCFS ⇒ First come first serve

- The process that allocate the CPU first is allocated CPU first.
- Implementation is easily managed with a FIFO Queue
- The FCFS scheduling algorithm is non-preemptive
  - Once the CPU has been allocated to a process that process keeps the CPU until it releases the CPU. either by terminating or by requesting CPU

BT = Burst time

AT = Arrival time.

CT = Completion time.

RS = Response time

WT = Waiting time

TA = Turn Around time

WT = TA − BT = RS − AT

TA = CT − AT

যদি arrival time সমান হয়ে তাহলে যে প্রসেস আগে আসবে সে অনুযায়ী কাজ হবে।

তার যদি AT না দেয়া থাকে তাহলে সবার arrival time ০ ধরতে হবে।

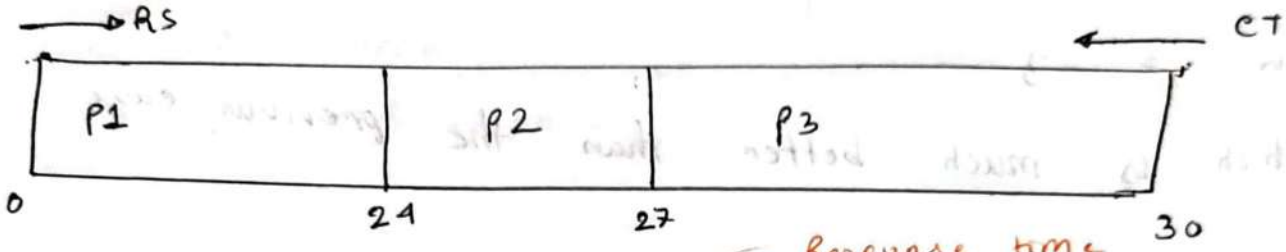
BT = number of CPU time that is requested by the process

Process	BT
P1	24
P2	3
P3	3

FCFS ত usually waiting time বেশি হয়

⇒ We considered that the process arrive in the following order: P1, P2, P3

Gantt Chart:



Response time

Waiting time of P1 = 0 − 0 = 0

" " " P2 = 24 − 0 = 24

" " " P3 = 27 − 0 = 27

∴ average waiting time = $\frac{0+24+27}{3}$ = 17

Turn around time;

∴ TA (P1) = 24 − 0 = 24

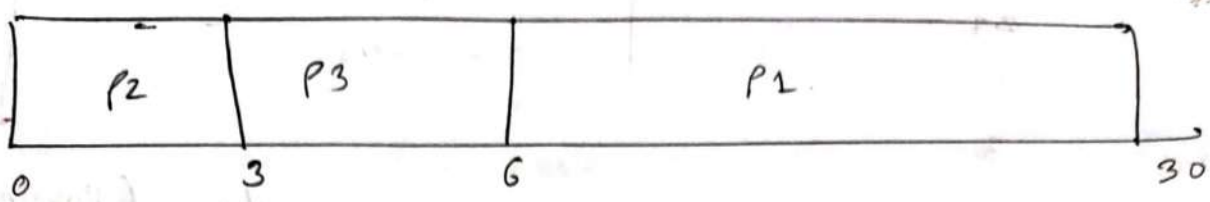
∴ TA (P2) = 27 − 0 = 27

∴ TA (P3) = 30 − 0 = 30

∴ average turn around time = $\frac{24+27+30}{2}$

same math টাই হবে যদি process এর order P2, P3, P1 consider করি তাহলে.

Gantt Chart



WT (P2) = 0 - 0 = 0

WT (P3) = 3 - 0 = 3

WT (P1) = 6 - 0 = 6

TA (P2) = 3 - 0 = 3

TA (P3) = 6 - 0 = 6

TA (P1) = 30 - 0 = 30

∴ avg W.T. = 3

(which is much better than the previous case

The avg W.T. under FCFS is not minimal and may vary substantially if the processes CPU burst time vary greatly.

Advantage:

- Fair & Simple

Consider one CPU bound and many I/O bound

Disadvantage:

- Convoy effect:
  - short process behind long process.
- It would be disadvantageous to allow one process to keep the CPU for an extended period of time.

SJF Scheduling

→ SJF ⇒ Shortest Job First

- Associate with each process the length of its next CPU burst.
  - Use these lengths to schedule the process with the shortest time
  - 2 টি way আছে.
    - Non Preemptive :→ Once CPU given to process it cannot be preempted until completes its CPU burst.
    - Preemptive :
      - ↳ if new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the shortest remaining time first (SRTF)

This term comes from the ability to remove a running process from CPU and allow another process to run in CPU.

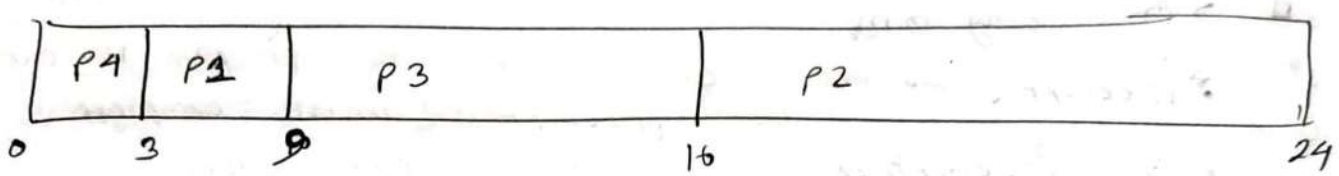
- SJF is optimal - gives minimum average waiting time for a given set of processes

SJF implementation এত difficult because next যে process টা আসবে তার CPU burst কত তা আগে থেকে কিভাবে জানব, এটাই হচ্ছে মূল সমস্যা

Process	BT
P1	6
P2	8
P3	7
P4	3

use SJF

⇒ Gantt Chart:



as সবার same time ↓ arrive ; so, যার burst time কম সবার আগে choose

WT (P1) = (3-0) = 3

WT (P2) = (16-0) = 16

WT (P3) = 9

WT (P4) = (0-0) = 0

TA (P1) = (9-0) = 9

TA (P2) = (24-0) = 24

TA (P3) = 16

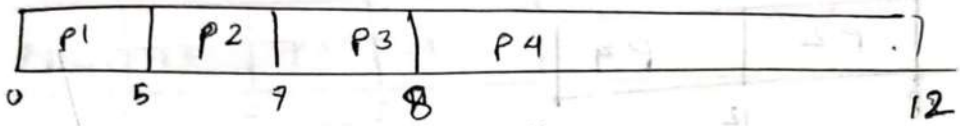
TA (P4) = 3

avg. w.t = $\frac{3+16+9+0}{4}$ = 7

Use FCFS.

Process	AT	BT
P1	0	5
P2	2	2
P3	4	1
P4	6	4

⇒ Gantt Chart:

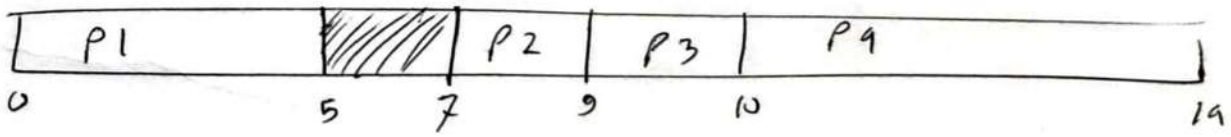


WT (P1) = WT (P2) = WT (P3) = WT (P4) =

Use FCFS:

Process	AT	BT
P1	0	5
P2	7	2
P3	9	1
P4	10	4

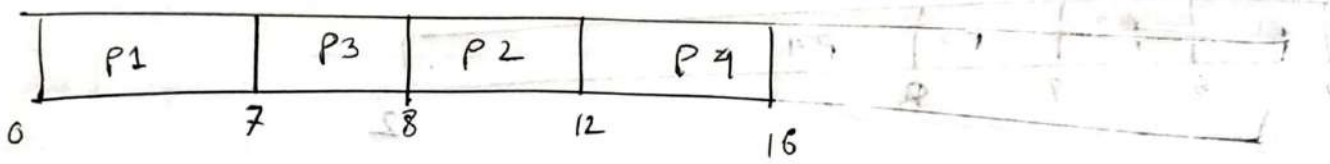
⇒ Gantt Chart:



Use SJF

Process	AT	BT
P1	0	7
P2	2	4
P3	4	1
P4	5	4

⇒ Gantt Chart:



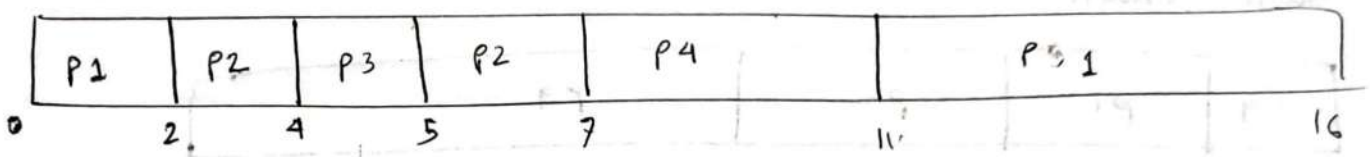
SJF Job Scheduler এ use করা হয়।

next যে process আসবে তার CPU burst time যতো কম possible তা SJF CPU Scheduler এ use হয়

Example of Preemptive SJF (SRTF)

Process	AT	BT
P1	0	5
P2	2	2
P3	4	1
P4	5	4

⇒ Gantt Chart:



WT (P1) = (0-0) + (11-2) = 9

WT (P2) = (2-2) + (5-4) = 1

WT (P3) = (4-4) = 0

WT (P4) = (7-5) = 2.

∴ TA (P1) = 16-0 = 16

∴ TA (P2) = 7-2 = 5

∴ TA (P3) = 5-4 = 1

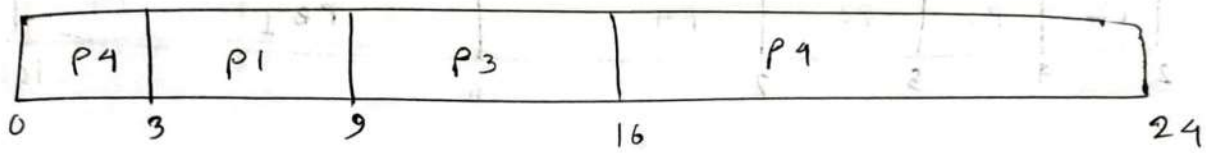
∴ TA (P4) = 11-5 = 6

- first এ start.
- যখনই notun process আসবে currently executing process এর arrival process এর সাথে burst time টাই কে pick হবে,
- আগের process এর remaining time যদি বেশি, এক্ষেত্রে চলবে.
- যখন যে process explored হবে normal SJF-
- SRTF always compare করে remaining time এর

Example of SJF.

Process	AT	BT
P1	0	6
P2	0	8
P3	0	7
P4	0	3

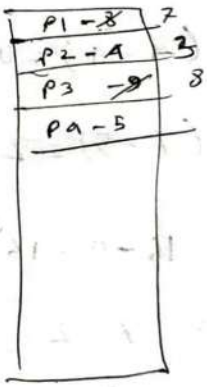
⇒ Gantt Chart:



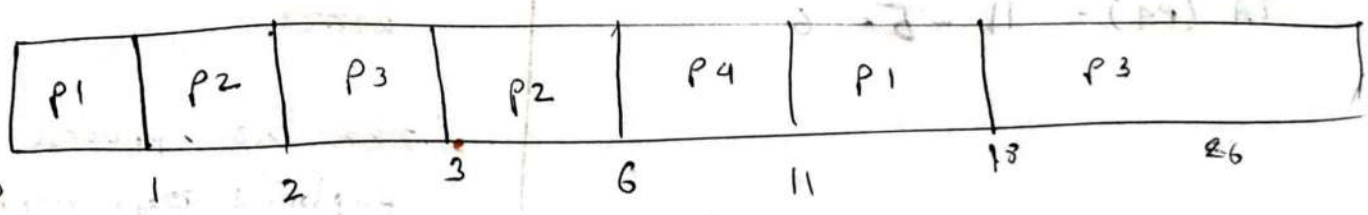
avg. w.t. = (3 + 9 + 16 + 0)/4 = 7

Example of SRTF.

Process	AT	BT
P1	0	~~8~~ 7
P2	1	~~4~~ 3
P3	2	~~9~~ 8
P4	3	5



⇒ Gantt Chart:



Example of SRTF

Process	AT	BT
P1	4	2
P2	7	3
P3	24	2
P4	10	4
P5	1	5

⇒ Gantt Chart:

Example of SRTF.

Process	AT	BT
P1	18	40
P2	29	17
P3	0	28
P4	21	37
P5	12	31

⇒ Gantt Chart

Priority Scheduling

⇒ A priority number is associated with each process.

- The CPU in allocated to the process with the highest priority (smallest integer ≡ highest priority).
  - Preemptive
  - Nonpreemptive.

- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time.

- Priority can be defined either internally or externally.
  - Factors for internal priority assignment:
    - Time limit, memory requirements, the number or open files, etc.
  - Factors for external priority assignment.
    - Importance of the process, the type and amount of funds being paid for computer use, department sponsoring works, etc.

- যদি দুইটা process এর priority same হয় তাহলে FCFS follow করবে

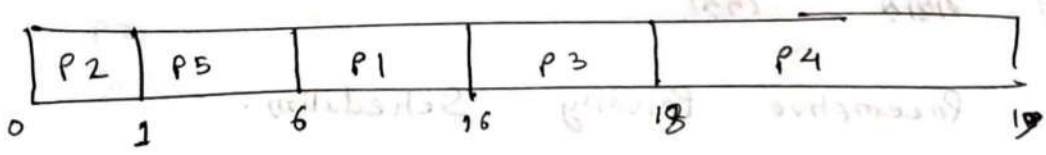
V. Preemptive priority scheduling algo. will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.

Example of Non Preemptive Priority Scheduling.

Process	BT	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

যে সময়ে AT 0 সব এর priority বিচার করে choose.

⇒ Gantt Chart:



WT (P2) = (0-0) = 0

WT (P1) = (6-0) = 6

WT (P3) = 16

WT (P4) = 18

WT (P5) = 1

Priority Scheduling Problem

Starvation

↳ কোন process যদি priority low হলে might never execute.

Priority Scheduling Solution.

Aging

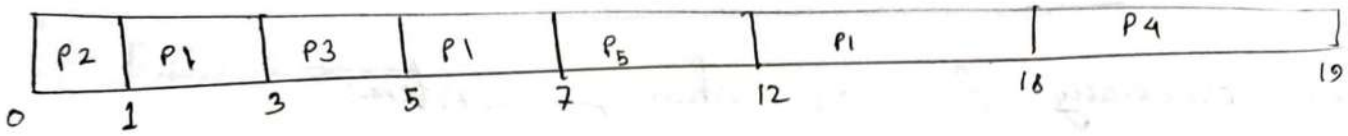
↳ As time progresses increases the priority of the process

↳ যদি কোনো process অনেক সময় ধরে waiting এ থাকে তাহলে specific time পর তার priority বাড়িয়ে দেয়া।

Example of Preemptive Priority Scheduling:

Process	AT	BT	Priority
P1	0	10	4
P2	0	1	1
P3	3	2	3
P4	5	1	5
P5	7	5	2

⇒ Gantt Chart:



WT (P1) = (1-0) + (5-3) + (12-7) = 8

WT (P2) = 0

WT (P3) = 0

WT (P4) = 13

WT (P5) = 0

Example on Preemptive Priority Scheduling

⇒

Process	AT	Priority	BT
P1	0	2	11
P2	5	0	28
P3	12	3	2
P4	2	1	10
P5	9	4	16

⇒ Gantt Chart

Round Robin (RR) algo

- Designed especially for time sharing systems.
- Similar to FCFS ; but preemption is added to switch between processes.
- Each process gets a small amount of time; known as Time Quantum, q ; usually 10 - 100 milliseconds. After this time elapses; process preempt হয় এবং added to the end of ready Queue.

Timer interrupts every quantum to schedule it's process

যদি ready queue তে n number of process থাকে এবং quantum, q . Then each proce gets 1/n of the CPU time in chunks of at most q time units at once. No proces waits (n-1) q time units

↳ Starvation problem নাই,

RR এর gantt chart করার সময় Queue মাথায় রাখা better.

Time Quantum, q বড় হলে queue টা fifo structure. follow করবে.

Time quantum, q অনেক কম হলে, context switch এর সংখ্যা বৃদ্ধি হয়; as a result overhead time. বেড়ে যাবে.

Performance

- q >> হলে Queue follow করবে
- q << হলে context switch বেড়ে যাবে.
- The average waiting time under the RR policy is often long. It depends on the size of time quantum.
- Typically higher average turnaround than SJF; but better response time.
- q should be large compared to context switching time

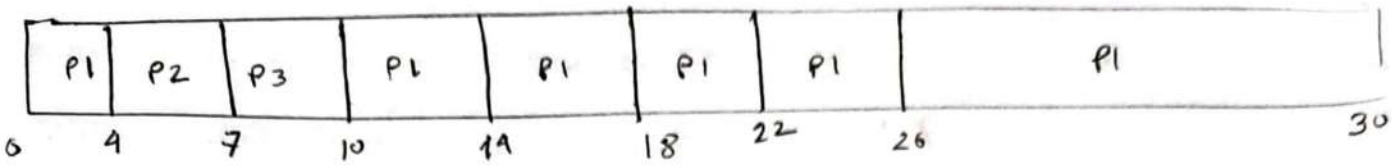
- Advantages:
  - Every process gets an equal amount of CPU.
  - RR is cyclic in nature; so no starvation
  - Performs best in terms of average response time

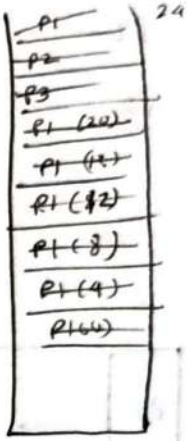
- Disadvantages:
  - Setting the quantum to short; increases the overhead and lowers the CPU efficiency; but setting it too long may cause poor response to short process.
  - No idea of priority.

Time Quantum = 4 ; Do RR scheduling

Process	BT
P1	24
P2	3
P3	3

⇒ Gantt Chart:





- average waiting time = 17/3 = 5.66 ms.

  WT (P1) = (0-0) + (10-4) = 6

  WT (P2) = (4-0) = 4

  WT (P3) = (7-0) = 7

- Typically better average runaround time than SJF; but response better.

- Quantum should be large compared to context switching time.

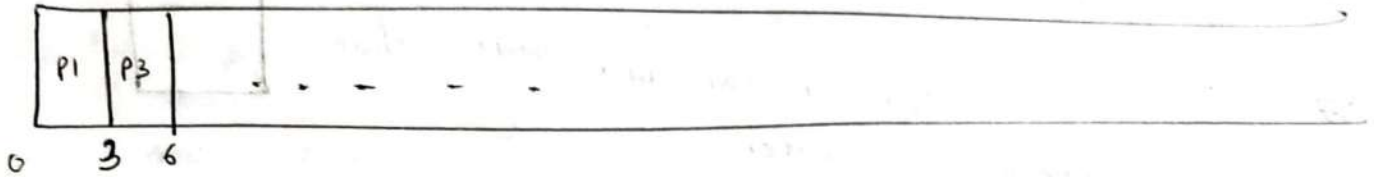
- Quantum usually 10 ms to 100 ms. Context switch < 10 μsec.

  ↳ মানে 10 টার বেশি context switch হবে per second না.

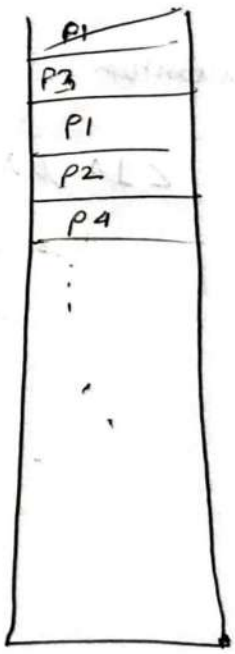
Process	AT	BT
P1	0	~~5~~ 5
P2	5	2
P3	1	~~7~~ 1
P4	6	3
P5	8	5

Tq = 3.

⇒



Context switch:



Throughput এর formula: = $\frac{\text{number of process}}{\Sigma BT.}$

↳ কোনো একটা specific time এ কতটুকু কাজ করতে পারছে.

Context switch বাড়লে হলে time তত বেড়ে যায়.

Multilevel Queue

⇒ এটা ready queue নিয়ে কাজ করে,

⇒ main memory এর queue তে কাজকে ভাগ ভাগ করে; that's why multilevel queue.

- ready queue
- job queue
- device queue

⇒ Multilevel Queue scheduling approach.

⇒ different different process কে fixed level এ assign করা,

⇒ higher priority থেকে শুরু (level 0, level 1, ...) এ lower priority থেকে bottom এ,

foreground process.

- interactive process; priority কম;

Background Process.

- Batch process
- gmail কে drive এ sync করা
- খুব তাড়াতাড়ি কাজ না background এ হতে থাকে,
- ধীর ধীর execute হয়.
- priority কম.

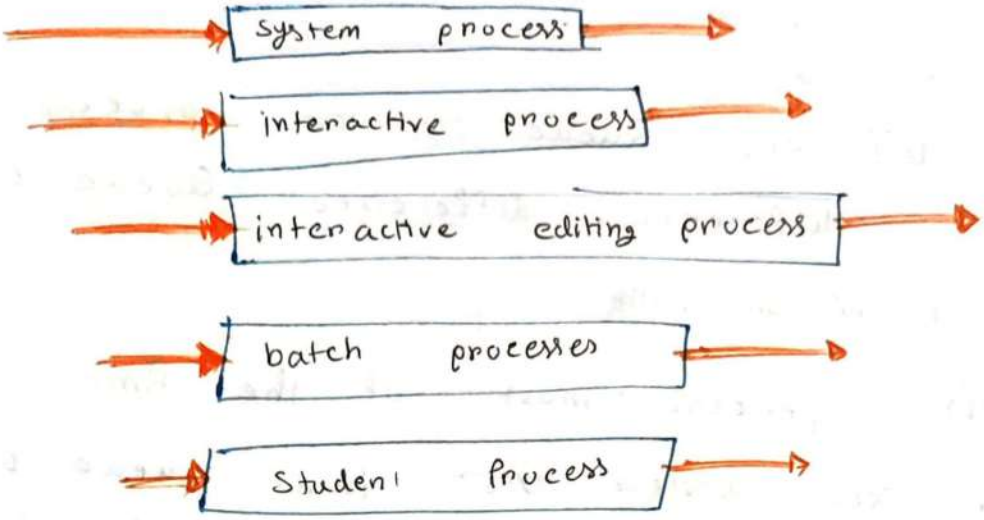
Foreground process কে উপরের level এ রাখতে, background process কে নিচে level এ রাখে

Ilo bound process
cpu bound process
Independent process
Foreground proc
Background proces

কোনো একটা process যে কোনো একটা level এ assign করার পর এ level এ permanent.

- Foreground process এ Round Robin use হয় আর background process এ FCFS use হয়,
- Scheduling must be done between the queues or there must be scheduling among themselves.
- যার priority বেশি তার queue আগে execute হবে : preemptive approach use হয়,
- When a queue is empty only then move to the next queue, for execution.
- Suppose Q3 তে P4 execute হচ্ছিল তখন Q1 এ হঠাৎ কোনো process আসলে direct Q1 এ move করবে, preemptive priority. scheduling করবে, আর উপরে level এর priority বেশি

highest priority



Disadvantage:

Starvation:

↳ lower process গত সুযোগই পায়না বলতে গেলে;

↳ lower priority Queue থাকাতে তারা CPU এর execution পাইবে না ; thats why starvation.

Multilevel Feedback Queue Scheduling.

- Starvation problem নাই;
- difference difference Queue তে যে process থাকে তাকে difference difference Queue তে move এর permission দিচ্ছি.
- একটা process ; most of the time cpu ক খেয়ে বসে থাকলে তাকে lower queue তে move করতে পারি.
- Between the queue, priority scheduling, foreground process RR use, then SJF use করি; then background process FCFS use.

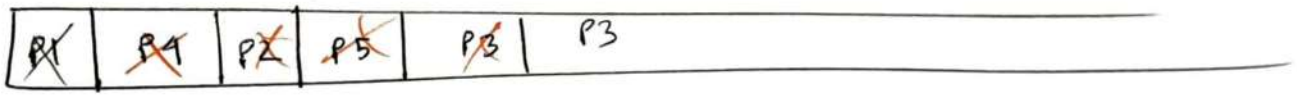
এখানে parameter (number of queue; কোন queue তে কি ধরনের algo চলবে; method to determine when to upgrade a process, when to degrade a process, initially কোন Queue তে কোন process execute arrive হবে তা বলে দিতে হয়.

RR

$Q = 5$.

Process	AT	BT
P1	0	3 0
P4	2	2 0
P2	3	2 0
P5	5	3 0
P3	7	7 2

⇒ Ready Queue



Gantt Chart