OS Assignment-1

[Anmol Baranwal --2820208]

Ans 1.a.

Preemptive Scheduling	Non-Preemptive scheduling
1. In this resources (CPU cycle) are allocated to a process for a limited time.	1. Once resources (CPU cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
2. Process can be interrupted in between.	2. Process cannot be interrupted until it terminates itself or its time is up.
3. If a process having high priority frequently arrives in the ready queue, a low priority process may starve.	3. If a process with a long burst time is running CPU, then later coming process with less CPU burst time may starve.
4. It has overheads of scheduling the processes.	4. It does not have overheads.
5. flexible.	5. rigid
6. cost associated	6. no cost associated.
7. The CPU utilization is high.	7. The CPU utilization is low.
8. It affects the design of the OS Kernel.	8. It doesn't affect the design of the OS Kernel.
9. e.g. Round Robin and Shortest Remaining Time First.	9. e.g. First come First Serve and Shortest Job First.

Ans 1.6.

Multilevel queue scheduling	Multilevel feedback queue scheduling
1. In this queue algorithm, ready queue is partitioned into several smaller queues and processes are assigned permanently into queues. The processes are divided on basis of their intrinsic characteristics such as memory size, priority etc.	1. In this algorithm, ready queue is partitioned into smaller queues on basis of CPU burst characteristics. The processes are not permanently allocated to one queue and are allowed to move between queues.
2. In this algorithm, queue are classified into two groups, first containing background processes and second containing foreground processes. 80% CPU time is given to foreground queue using Round Robin algorithm and 20% time is given to background processes using FCFS algorithm.	2. Here, queues are classified as higher priority queue and lower priority queues. If process takes longer time in execution it is moved to lower priority queue. Thus, this algorithm leaves I/O bound and interactive processes in higher priority queue.
3. The priority is fixed in this algorithm. After all processes in one queue get executed completely then only processes in other queue are executed. Thus, starvation can occur.	3. The priority for process is dynamic as process is allowed to move b/w queue. A process taking longer time in lower priority queue can be shifted to higher priority queue and vice versa.
4. Since, processes do not move b/w queues, it has low	4. Since, processes are not allowed to move b/w

4. scheduling overhead and is inflexible.

4. queues, it has high scheduling overhead and is flexible.

Ans2. Multiprogramming

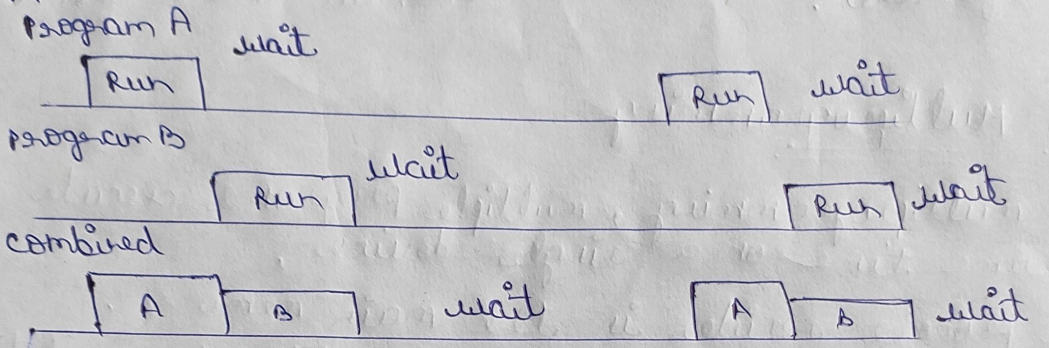
- In multiprogramming, multiple programs execute at a same time on a single device.
- The process resides in the main memory.
- It uses batch OS. The CPU is utilized completely while execution.
- The processing is slower, as a single job resides in the main memory while execution.

Multi programmed system's working

- As soon as one job goes for an I/O task, the OS interrupts that job, chooses another job from the job pool (waiting queue), gives CPU to this new job and starts its execution. The previous job keeps doing its I/O operation while this new job does CPU bound tasks. Now say the second job also goes for an I/O task, the CPU chooses a third job and starts executing it. As soon as a job completes its I/O operation and comes back for CPU tasks, the CPU is allocated to it.
- In this way, no CPU time is wasted by the system waiting for the I/O tasks to be completed.

e.g. program A runs for some time and then goes to waiting state. In the mean time program B begins its execution. so the CPU does not waste

its resources and gives program B an opportunity to run.



Multitasking

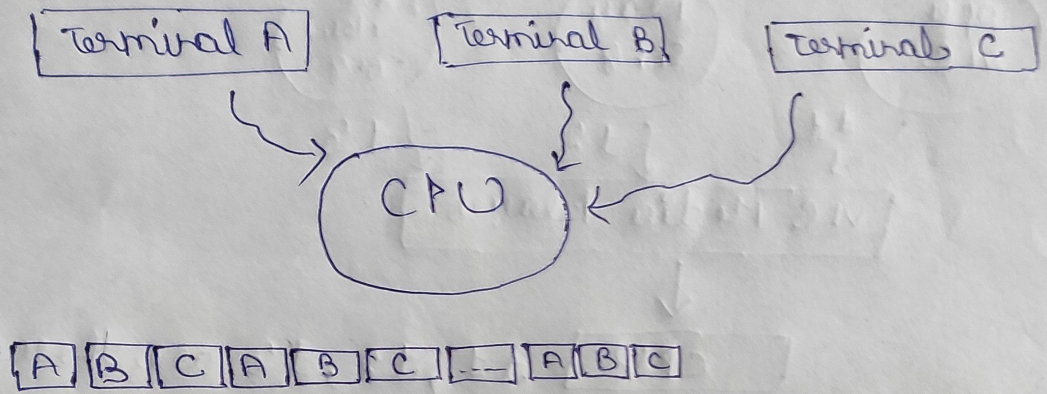
- A single resource is used to process multiple tasks.
- The process resides in the same CPU.
- It is time sharing as the task assigned switches regularly.
- It follows the concept of context switching.

Multitasking system's working

- In a time sharing system, each process is assigned some specific quantum of time for which a process is meant to execute. Say there are 4 processes P1, P2, P3, P4 ready to execute. So, each of them are assigned some time quantum for which they will execute e.g. time quantum of 5 ns. As one process begins execution (say P2), it executes for that quantum of time. After 5 ns, the CPU starts the execution of the other process (say P3) for the specified quantum of time.
- Thus the CPU makes the processes to share time slices b/w them and execute accordingly. As soon as time quantum of one process expires, another

process begins its execution.

In a more general sense, multitasking refers to having multiple programs, processes, tasks, threads running at the same time.



As depicted, at any time the CPU is ~~waiting~~ executing only one tasks while other tasks are waiting for their turn. The illusion of parallelism is achieved when the CPU is reassigned to other tasks. i.e. all the three tasks A, B and C are appearing to occur simultaneously because of time sharing.

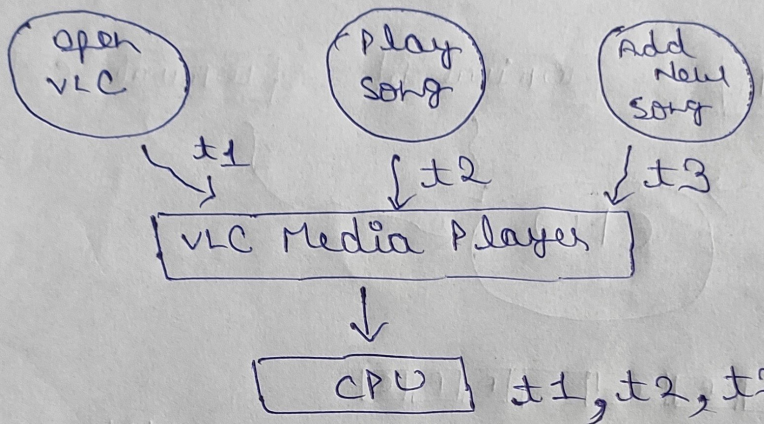
Multi threading

A thread is a basic unit of CPU utilization. Multi threading is an execution model that allows a single process to have multiple code segments (i.e. threads) running concurrently within the "context" of that process.

e.g. VLC media player, where one thread is used for opening the VLC media player, one thread for playing a particular song and another thread for adding new songs to the playlist.

28/02/08 (6)

Multi threading is the ability of a process to manage its use by more than one user at a time and to manage multiple requests by the same user without having to have multiple copies of the program.



t1, t2, t3 are threads

- Since there are multiple threads in a program, so if one thread is taking too long to execute or if it gets blocked, the rest of the threads keep executing without any ~~problem~~ problem. This the whole program remains responsive to the user by means of remaining threads.
- It is also less costly as threads share resources of the parent process, creating threads and switching b/w them is comparatively easy.

Ans3.

Process	A.T. (ms)	B.T. (ms)	Priority
P1	0	4	1
P2	1	5	2
P3	2	2	3
P4	3	1	4
P5	4	6	3
P6	6	3	5

28/2/2023 (7)

First Come First Serve ⇒

This scheduling algorithm is on the basis of arrival time.

The terms will be used as -

AT = Arrival Time, BT = Burst Time

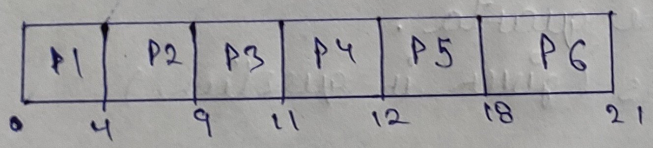
CT = completion Time, WT ⇒ Waiting Time ⇒ TAT − BT

RT = Response Time

TAT = Turn Around Time ⇒ CT − AT

Process	AT	BT	CT	TAT	WT	RT
P1	0	4	4	4	0	0
P2	1	5	9	8	3	3
P3	2	2	11	9	7	7
P4	3	1	12	9	8	8
P5	4	6	18	14	8	8
P6	6	3	21	15	12	12
				59	38	38

Gaant Chart ⇒



WTavg ⇒ $\frac{38}{6}$ ⇒ 6.33 ms

RTavg ⇒ 6.33 ms, TATavg ⇒ $\frac{59}{6}$ ⇒ 9.83 ms

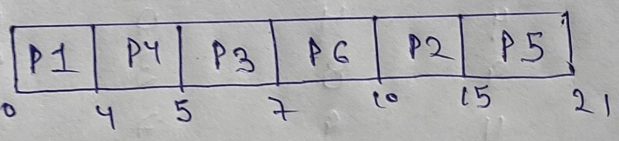
Shortest Job First (SJF) ⇒

The idea behind SJF is to pick the quickest little job that needs to be done i.e. BT.

28/2/2023 (8)

Process	AT	BT	CT	TAT	WT	RT
P1	0	4	4	4	0	0
P2	1	5	15	14	9	9
P3	2	2	7	5	3	3
P4	3	1	5	2	1	1
P5	4	6	21	17	11	11
P6	6	3	10	4	1	1
				46	25	25

Gaant Chart ⇒



WT avg ⇒ $\frac{25}{6}$ ⇒ 6.167 ms , TAT avg ⇒ 7.67 ms

RT avg ⇒ 6.167 ms (RT is same as WT because every process is terminated at once)

Round Robin Scheduling ⇒

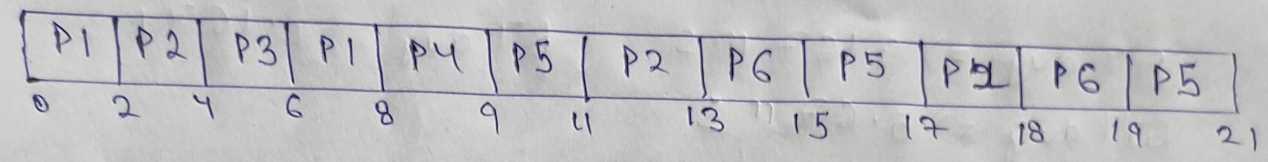
It is same as FCFS, except CPU bursts are assigned with time quanta.

Time Quantum = 2 ms (given in question)

Process	AT	BT	~~TAT~~ CT	TAT	WT	RT
P1	0	4	8	8	4	0
P2	1	5	18	17	12	1
P3	2	2	6	4	2	2
P4	3	1	9	6	5	5
P5	4	6	21	17	11	5
P6	6	3	19	13	10	7
				65	44	20

Ready Queue ⇒ P1 P2 P3 P1 P4 P5 P2 P6 P5 P2 P6 P5

Gaant chart ⇒



TAT avg ⇒ $\frac{65}{6}$ ⇒ 10.83 ms

WT avg ⇒ $\frac{44}{6}$ ⇒ 7.33 ms

RT avg ⇒ $\frac{20}{6}$ ⇒ 3.33 ms

Priority Scheduling

Every job is assigned a priority.

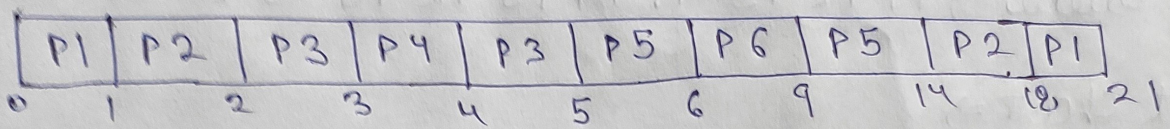
According to question: High priority number means high priority

Note :- If 2 jobs have same priority then they are processed according to FCFS.

Process	AT	BT	Priority	CT	TAT	WT	RT
P1	0	4	1	21	21	17	0
P2	1	5	2	18	17	12	0
P3	2	2	3	5	3	1	0
P4	3	1	4	4	1	0	0
P5	4	6	3	14	10	4	1
P6	6	3	5	9	3	0	0
					55	34	1

2820208 (10)

Gaant Chart =>



TAT avg => 9.167 ms , RT avg => 0.167 ms

WT avg => 5.67 ms