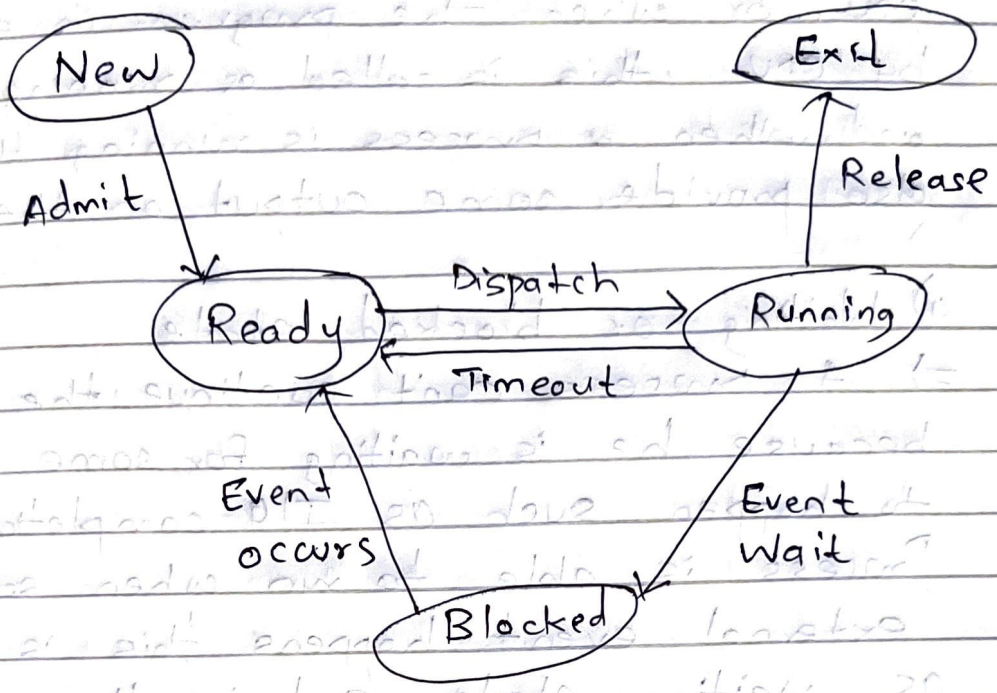
2. Process & Process Scheduling

Q.1 Draw process state diagram and explain following states.

⇒



- A process can have one of the following ~~process~~ state at a time :-

1) New State

⇒ A process that has been just created but not yet been loaded into Main memory. Every new operation which is requested to the system is known as new born process.

2) Ready State

⇒ When process is ready to execute but it is waiting for CPU being assigned to it is called as Ready state.

3) Running State

⇒ The process is currently being executed. When the process is running under the CPU, or when the program is executed by CPU, this is called as running process and when a process is running this will also provide some output on the screen.

4) Waiting or blocked state

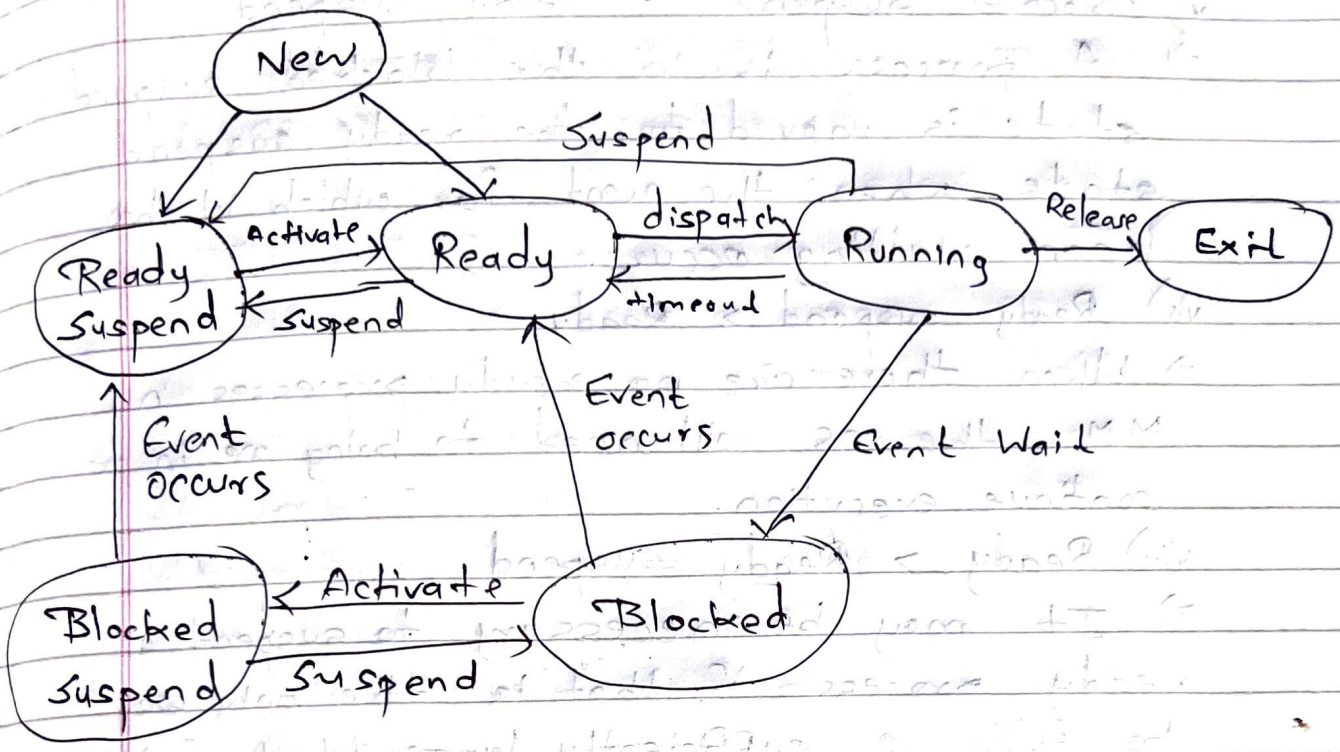
⇒ A process can't continue the execution because he is waiting for some even to happen such as I/O completion. Process is able to run when some external event happens this is called as waiting state and in this process is not under the execution instead the process is stored out of memory and when the user will provide the input and then this will again be on ready state.

5) Terminated

⇒ After the completion of the process, the process will be automatically terminated, this is called as terminated state of process.

Q.6 Draw and explain 7 state process transition program.

⇒



i) Ready

⇒ The process is in MM and available for execution.

ii) Blocked

⇒ The process is in MM and awaiting an event I/O event.

iii) Blocked suspend

⇒ The process is in secondary memory and awaiting an event.

iv) Ready suspend

⇒ The process is in secondary memory but is available for execution as soon as it is loaded into MM.

v) Blocked > Blocked Suspend

⇒ If there are no ready process, then at

least one blocked process is swapped out to make room for another process that is not blocked.

vi) Blocked suspend > Ready suspend

⇒ A process in the blocked suspend state is moved to the ready suspend state when the event for which it has been waiting occurs.

vii) Ready suspend > Ready

⇒ When there are no ready processes in MM, the OS will need to bring one in to continue execution.

viii) Ready > Ready suspend

⇒ It may be necessary to suspend a ready process if that is the only way to free a sufficiently large block of MM.

ix) New > Ready suspend & New > Ready

⇒ When a new process is created, it can either be added to the ready queue or the ready, suspend queue.

x) Blocked suspend > blocked

⇒ A process terminates, freeing some main memory. There is a process in the blocked suspend queue with a higher priority than any of the processes in the ready suspended queue.

xi) Running > ready suspend

⇒ If the OS is preempting the process because a higher-priority process on the blocked suspend queue has just become unblocked,

Q.7 Explain three types of Scheduler.

⇒ • There are three types of schedular to schedule a process.

Schedulers
- 1) Long-term
- 2) Short-term
- 3) Medium-term

1) Long Term Scheduler (LTS)

⇒

i) When the programs are submitted to the system for the purpose processing, LTS comes to know about it.

ii) It's job is to choose the processes from the queue and place them into MM for execution purpose.

iii) CPU bound process require more ~~time~~ CPU time and less I/O time until execution completes.

iv) On contrary, I/O bound process require less CPU time and more I/O time for computation.

v) The main job of LTS is to provide balance mix of I/O bound & CPU bound jobs.

vi) LTS controls the degree of multiprogramming.

2) Short Term Scheduler (STS)

=>

i) Process which are in ready queue wait for CPU.

ii) A STS chooses the process from ready queue and assigns it to the CPU based on some policy.

iii) These policies can be FCFS, SJF, priority based and round robin, etc.

iv) Main objective is increasing system performance by keeping CPU busy.

v) STS is faster than LTS and should be invoked more frequeeently compare to LTS.

3) Medium Term Scheduler (MTS)

=>

i) IF the degree of multiprogramming increases, MTS swap out the processes from MM.

ii) The swapped-out processes again swapped in by MTS.

iii) This is done to control the degree of multiprogramming or free up a memory.

Q.2 Comparison between LTS, STS, MTS

⇒

LTS	STS	MTS
1) Selects process from queue and loads them into memory for execution.	1) Selects process from ready queue and assigns it to the CPU.	1) Swaps in and out the processes from memory.
2) Speed is less than STS	2) Speed is very fast and invoked Frequently than LTS	2) Speed is in between both STS and LTS.
3) Transition of process state from New to ready.	3) Transition of process state from Ready to running.	3) No process state transition
4) Not present in time sharing system.	4) Minimal in time sharing system.	4) Present in time sharing system.
5) It controls the degree of multiprogramming through placing process in ready queue.	5) It has control over degree of multiprogramming as it allocates processes to CPU for execution	5) Reduces the degree of multiprogramming by swapping processes in and out.
6) Also called as job scheduler.	6) Also called as CPU scheduler	6) Swapping Scheduler.

Q.8 Explain 5 process scheduling criteria

⇒ i) CPU Utilization

⇒ CPU utilization is amount of time CPU remains busy. It ranges from 0% to 100%. Typically it ranges from 40% (for lightly loaded system) to 90% (for highly loaded system).

ii) Throughput

⇒ If the CPU is busy executing processes, the work is being done.
Number of jobs processed per unit time is called as throughput.
It varies from 1 process per hour to 10 processes per second.

iii) Turnaround Time

⇒ Turnaround time is the time elapsed between submission of jobs and completion of its execution.

iv) Waiting time

⇒ The amount of time that a process spends in a ready queue.
Sum of time spents in ready queue is waiting time

v) Response time

⇒ It is the time from submission till the first response is produced.

Q.3 What is process and explain the structure of PCB.

⇒ • The term process refers to program code that has been loaded into a computer's memory so that it can be executed by the control process unit.

• Structure of PCB

⇒

Process state
Process Number
Program Counter
Registers
Memory Limits
List of open files
⋮

i) Process state

⇒ This specifies the process state i.e., new, ready, running, waiting or terminated

ii) Process Number

⇒ This shows the number of particular process.

iii) Program counter

⇒ This contains address of next instruction that needs to be executed in the process.

iv) Registers

⇒ This specifies the registers that are used by process. They may include accumalator, index register, stack pointer, general purpose register, etc.

v) List of Open files
⇒ These are the files that are associated with the process.

vi) CPU Scheduling Information
⇒ The process priority, pointers to scheduling queues, etc. the CPU scheduling information that is contained in the PCB.

vii) Accounting Information
⇒ The time limits, account numbers, amount of CPU used, process numbers, etc. are all a part of the PCB accounting information.

viii) I/O status Information
⇒ I/O devices allocated to process.

Basic Scheduling Algorithm

1) First Come First Served (FCFS)

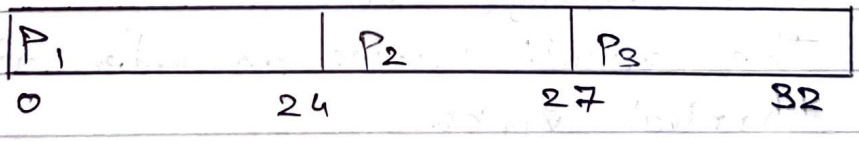
⇒

- This is a Non-preemptive scheduling algorithm. FCFS strategy allocates the CPU to processes in the order of their arrival.
- This algorithm treats ready queue as FIFO. A process does not give up.
- If we assume the arrival time is zero,

Consider the following example :

Process	Burst time
P_1	24
P_2	03
P_3	05

The Gantt chart shows the result -



TAT for P_1 = 24 - 0 = 24

P_2 = 27 - 0 = 27

P_3 = 32 - 0 = 32

∴ Avg. TAT = $\frac{24+27+32}{3}$ = 27.67

WT for P_1 = 0

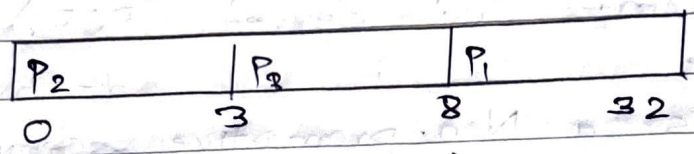
P_2 = 24

P_3 = 27

∴ Avg. WT = $\frac{0+24+27}{3}$

= 17

- If the order of arrival is changed and considered as P_2, P_3, P_1



TAT $P_2 = (3-0) = 3$

$P_3 = (8-0) = 8$

$P_1 = (32-0) = 32$

$\therefore$ Avg TAT $= \frac{3+8+32}{3} = 14.33$

WT $P_2 = 0$

$P_3 = 3$

$P_1 = 8$

$\therefore$ Avg. WT $= \frac{0+3+8}{3} = 3.66$

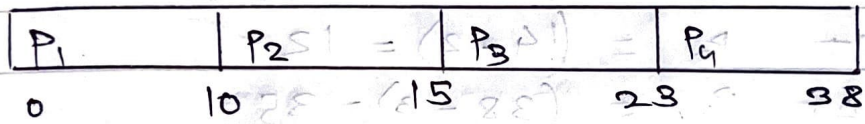
- The result show that there is a significant reduction in average TAT and average WT.
The result varies as order of job arrival varies.

2) Shortest Job First (SJF).

=) • SJF may be preemptive or non-preemptive Reordering the jobs so as to run the SJF improves the avg. response time.
• Ready queue is maintained in order of increasing job lengths.
• For example, non-preemptive SJF algorithm

⇒

Process	Arrival time	Burst time
P_1	0	10
P_2	1	5
P_3	2	8
P_4	3	15



TAT $P_1 = (10-0) = 10$

$P_2 = (15-1) = 14$

$P_3 = (23-2) = 21$

$P_4 = (38-3) = 35$

$\therefore$ Avg TAT $= \frac{10+14+21+35}{4} = 20$

WT $P_1 = (0-0) = 0$

$P_2 = (10-1) = 9$

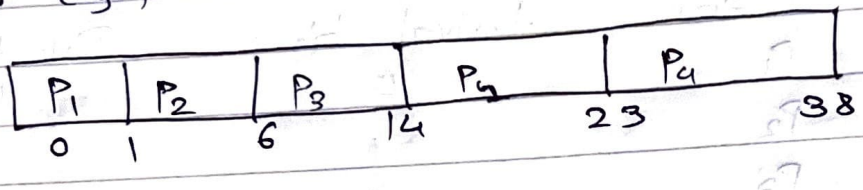
$P_3 = (15-2) = 13$

$P_4 = (23-3) = 20$

$\therefore$ Avg WT $= \frac{(0+9+13+20)}{4} = 10.5$

- In preemptive SJF, if newly arrived process has less execution time with compare to currently executing process, then the CPU will be given newly arrived process
- The preemptive SJF is called as Shortest Remaining Time Next Scheduling (SRTN).

For e.g.)



TAT $P_1 = (23-0) = 23$
TAT $P_2 = (6-1) = 5$
TAT $P_3 = (14-2) = 12$
TAT $P_4 = (38-3) = 35$

$\therefore$ Avg. TAT $= \frac{23+5+12+35}{4} = 18.75$

WT $P_1 = (14-1) = 13$
WT $P_2 = (1-1) = 0$
WT $P_3 = (6-2) = 4$
WT $P_4 = (23-3) = 20$

$\therefore$ Avg. WT $= \frac{13+4+20}{4}$

$= 9.25$

3) Priority scheduling

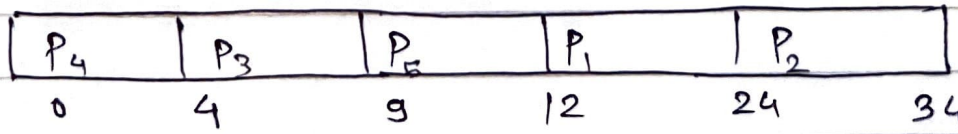
⇒

- In priority scheduling, each processor has a priority which is integer value assigned to it.
- Smallest integer considered as highest priority and largest integer is considered as lower priority.
- Preemptive and non-preemptive SJF is a priority scheduling where priority is the shortest execution time of job.
  In this algorithm, low priority processes may never execute. This is called starvation.
- Solution to this starvation problem is aging.
  In aging as time progresses, increase the priority of the process so that lowest priority processes gets converted to highest priority gradually.

For e.g.;

Process	Burst time	Priority
P_1	12	4
P_2	10	5
P_3	5	2
P_4	4	1
P_5	3	3

⇒ The Gantt chart -



TAT P_1 = (24 - 0) = 24

TAT P_2 = (34 - 0) = 34

TAT P_3 = (9 - 0) = 9

TAT P_4 = (4 - 0) = 4

TAT P_5 = (12 - 0) = 12

$$\therefore \text{Avg. TAT} = \frac{24+34+9+4+12}{5} = 16.6$$

WT P_1 = 12

WT P_2 = 24

WT P_3 = 4

WT P_4 = 0

WT P_5 = 9

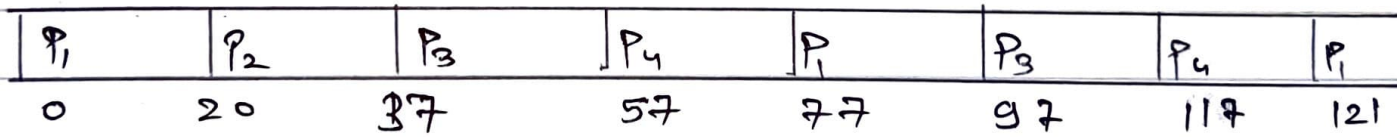
$$\therefore \text{Avg. WT} = \frac{12+24+4+0+9}{5} = 9.8$$

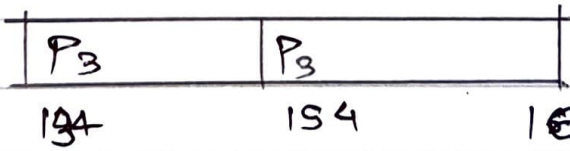
4) Round Robin Scheduling

⇒
- The Round-robin (RR) scheduling algorithm is designed especially 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 of time slice, is defined.
- A time quantum is generally from 10 to 100 milliseconds.
- The ready queue is treated as a circular queue.

For e.g., Time quantum = 20

Process	Burst time	
P_1	53	// 33, 13, 0
P_2	17	// 0
P_3	68	// 48, 28, 8, 0
P_4	24	// 4, 0





$$\text{Avg. TAT} = \frac{134 + 37 + 162 + 121}{4} = 113.5 \text{ ms}$$

Avg. WT ⇒

WT. P_1 = 0 + (77 − 20) + (121 − 97) = 81

WT P_2 = 20

WT P_3 = 37 + (97 − 57) + (134 − 117)

~~+ 154~~ = 94

WT For P_4 = 57 + (117 - 77)

= 57 + 40

= 97.

∴ Avg. WT = $\frac{81 + 20 + 94 + 97}{4}$

= 73 ms.

Q. 3 Discuss importance of Multithreading

=> ~~i) In OS system supports~~

i) In OS that supports multithreading, can consists of many threads. These threads run in parallel improves the application performance.

ii) Each such thread has its own CPU state and stack, but they share the address space of the process and the environment.

iii) Considering the advantages of user level and Kernel level threads, a hybrid threading model using both types of threads can be implemented.

iv) The solaris OS supports this hybrid model.

v) In this implementation, all the thread management functions are carried out by user level thread package at user space.

vi) So operations on thread do not require Kernel intervention.

Q.4 Difference between Process and Threads.

⇒

Process	Threads
i) Program in execution is called as process.	i) Thread is a part of process.
ii) It is also called as heavy weight process.	ii) It is also called as light weight process.
iii) Process context switch takes more time as compared to thread context switch because it needs interface of OS.	iii) Thread context switch takes less time as compared to process context switch because it needs only interrupt to kernel only.
iv) New process creation takes more time compared to new thread creation.	iv) New thread creation takes less time compared to new process creation.
v) New process termination takes more time as compared to new thread termination.	v) New thread termination takes less time as compared to new process termination.
vi) Each process executes the same code but has its own memory and file resources.	vi) All threads can share same set of open files, child processes.

Q. 5 Difference between user & kernel ~~and~~ level threads.

→

User Level	Kernel Level
1) Kernel is unware of the thread.	1) The thread management is carried out by kernel.
2) All of the work of thread management is done by thread package.	2) All thread management activities are carried in kernel space.
3) Kernel threads are generally requires more time to create and manage than the user thread.	3) Creating and destroying threads requires less time.
4) These are the platform independent	4) Theseare the platform dependent.