CPU Scheduling

Scheduling concepts.

Scheduling is the concept of sequencing a task on CPU, thereby making full use of CPU.
The aim of CPU scheduling is to make the system efficient, fast and fair.

It consists of two scheduling queues.

① Job Queue :- which is also known as Job pool, It is the set of all processes created at a time in the system.

② Ready Queue :- It is the set of all processes residing in the main memory which are ready to execute and waiting for the CPU allotment in sequential manner

Performance criteria of Scheduling.

1. Arrival Time (AT) :- It is the arriving time of the process in the system.
2. Completion Time (CT) :- The time at which process completes its execution
3. Burst Time (BT)/(ET) :- It is also known as Execution Time, i.e., time taken by the process on CPU
4. Waiting Time (WT) :- The amount of time the process spend waiting for the allotment of CPU
5. Turn Around Time (TAT) :- The amount of Time to execute a particular process

$TAT = (WT + ET)$ or $(CT - AT)$

Teacher's Signature.....

WT = (TAT − ET) or (CPU enter time − AT)

Date
Page

⑥ Response Time (RT):- Time from the Submission of Request until the first response is produced.

RT = [(Time at which process get CPU 1st time) − (AT)]

⑦ Throughput (TP):- The no. of processes that are completed per unit time.

$$TP = \frac{\text{No. of processes completed}}{\text{Total time taken for completion of all processes}}$$

⑧ CPU Utilization:- It is a concept of keeping CPU Busy, also it ranges from 0 to 100%

$$\text{CPU Utilization in percent} = \frac{\text{Processor Busy}}{\text{Processor Busy + Processor Idle}} \times 100$$

Objectives of CPU Scheduling

1) Efficiency must be very high (CPU Utilization ↑)
2) Maximize Throughput (Throughput ↑)
3) Minimize Response time (Response Time ↓)

CPU Scheduling

Pre-emptive Scheduling	Non-Pre-emptive Scheduling
Processor can be pre-empted to execute a different process in the middle of execution of any current executing process.	• once execution of a process starts it must finish it, before executing the other. It can't be paused in the middle.

Teacher's Sig. ature.....

Schedulers

It is a special System S/w by which Scheduling of processes are achieved.

There are 3 types of Scheduler

① Long term scheduler (Job Scheduler)

These Scheduler selects the process from the Job queue and brought them into the ready Queue

These type of Scheduler controls the degree of multi-programming i.e., no., of processes residing into the memory.

② Short term scheduler (CPU scheduler by its algorithm)

These type of Scheduler are responsible for CPU Scheduling. This Scheduler selects the processes from the ready queue and allocates it to the CPU. This is done by using Scheduling algorithm.

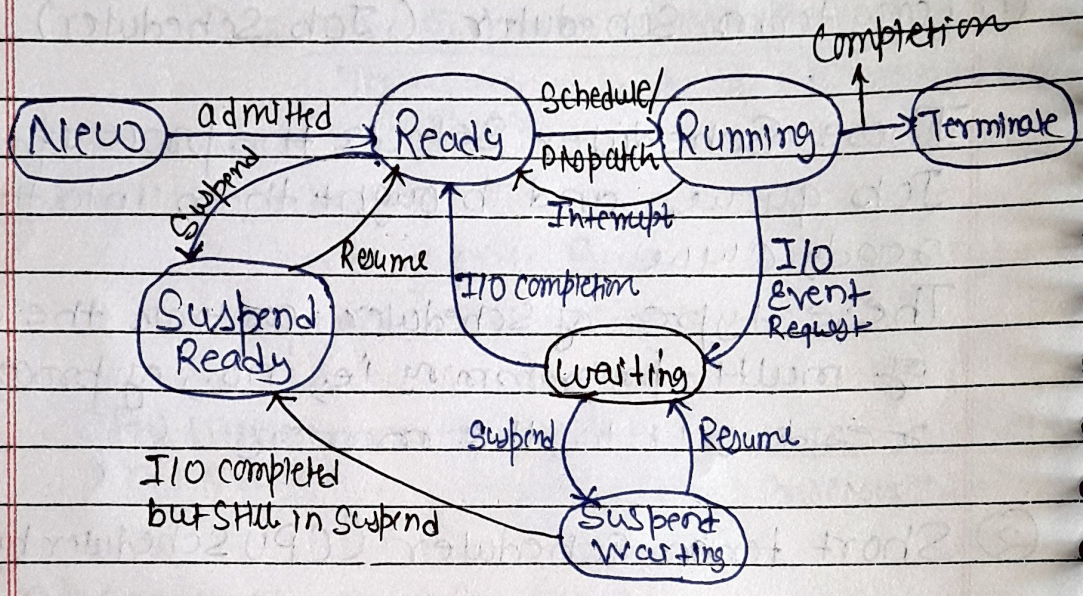
③ mid term Scheduler (process Swapping / context Switching)

These are introduced to handle context Switching i.e., Swapping of processes in and out from the CPU It reduces the degree of multiprogramming

Teacher's Signature.....

Note:-
Scheduler works with the dispatcher that gives control of the CPU to the process selected by the short term scheduler.

Process (State / Transition) Diagram



a) A process is an executing program which includes the values of program counter, registers and variables, It is different from a program because program is a group of instructions and processes are activity for executing a segment of program.

b) A process can be described as I/O count process which spends max time on I/O than CPU and another is CPU count process which spends max time on CPU computation.

c) In other words Process is a part of program in execution.

Teacher's Sig. ature.....

Process States

(1) New :- A program which is going to be picked by O.S into the main memory is called new Stat process

(2) Ready :- A process which is Ready for execution and Resides in the main memory is called Ready state process.

Note :- O.S picks new process from Secondary memory into main memory by using a Scheduler.

(3) Running - A process using CPU known to be in Running state. A process will be choosen from Ready state by the O.S using Scheduling algorithm which decides the Sequence of the process executing in the CPU

(4) Terminate :- When a process finishes execution it comes to the terminate state i.e., Content of the process (PCB) Process control Block will be deleted from the main memory.

(5) Wait State / Blocked State :- From Running State a process can make a transition to the block or wait state depending on the behaviour of the process. When a process requires or wait for a certain Resource to be assigned from I/p or O/p then these processes are block from CPU utilization and wait for the I/O cycle.

Teacher's Signature.....

Date
Page

Note:-
Ready state and wait state use a buffer memory to store the process in Ready Queue or Waiting Queue. Whenever a process arise it is stored into the buffer and if there is No space in to the buffer, then these processes are send to suspended state. It can be suspend ready state for ready Queue or suspend wait state for wait Queue.

Process Control Block

- There is a process control Block (PCB) for each process enclosing all the information about the processes.
- It is a data structure which contains the following.

1) Naming of the process (Process ID)
2) State of the process (Process State)
3) Resources allocated to the process
4) memory allocated to the process.
5) Scheduling information
6) I/O devices associated with process

PCB Diagram.

Process State
Process ID
Program Counter
Registers
memory limits
list of open files
miscellaneous block

(1) Process State:- It defines the state in which process resides in the realtime, in the process state diagram.

(2) Process ID:- It is the process no. /ID used by kernels to uniquelly Identify a process i.e., the process belongs to which kind of program and what is the no. of process in the sequence of currently running program

(3) Program Counter (PC):- It Stores and Indicates the address of Next Instructions.

(4) Registers:- The no. of Registers and its types depending upon the computer architecture is defined in the Register block
- Storage area near CPU (Group of flip-flops) fast

(5) memory limits:- It defines the memory management information such as base Registers and its limits. and pasing and Segmentation information.

Teacher's Signature.....

Date
Page

(6) List of open files:- It defines the list of resources currently in use by the process.

(7) Miscellaneous block:- It consists the processor scheduling information, I/O status information and accounting information of the process (Amount of CPU to be used, time limits, no. of threads etc.)

Context Switching.

- A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread.
- It allows for one CPU to handle multiple process or threads with Response Time ↓ and no need of more processor.
- It is a type of mechanism to store and restore the state or context of a CPU from the same point at a later time.

Process Address Space

- An address space is range of valid address in a memory that are accessible for a program or process for executing Instructions alongwith storing data.
- The memory can either be physical or virtual

Teacher's Sig. ature.....

a) Physical address space :- This type of address space is created in RAM

b) Virtual address space :- This type of address space is created outside the main memory it can be (HDD or SSD). of virtual memory.

Thread

- A thread is a flow of execution / Path of execution through the process code with its own program counter, system register, and stack.
- A process can contain multiple threads.
- Also known as light weight process.
- Idea is to achieve parallel computation by dividing a process into multiple threads.

Adv.

1) Thread minimize context switching.
2) Provide concurrency within a process.
3) Efficient communication and economical too.

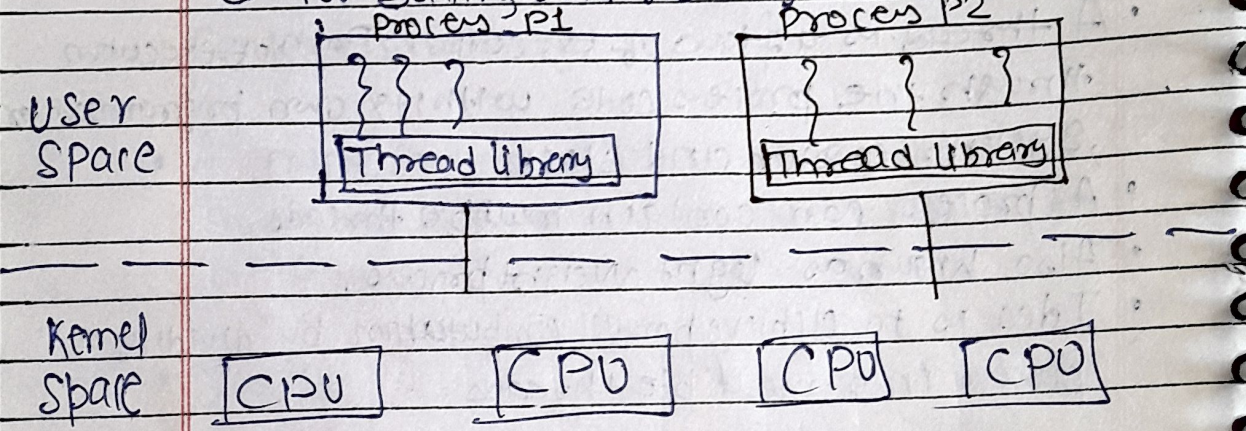
Process	Thread
1) Process is heavy weight	Thread is light weight
2) Process switching needs interaction with O.S	Thread switching doesn't need to interact with O.S
3) If one process is blocked then no other process can execute	When one thread is block, then second process of same task can run
4) In multiple process implimentation each process has its own memory and file resources	All threads can share same set of open files, child process etc, one thread can read, write or even completely wipe out another threads stack.
5) Each process operates independently of the other	

Teacher's Signature.....

Date
Page

Types of Threads.

1) User Level Threads.
- The application starts with a single thread.
- Thread library contains code for
  1. creating and destroying threads
  2. for passing message and data b/w threads
  3. for scheduling thread execution
  4. for saving and restoring thread contexts.



Adv.
1) Thread switching doesn't require kernel mode privileges
2) User level thread can run on any OS
3) User level thread are fast to create and manage
4) Scheduling can be application specific.

Dis
1) In some OS, most System calls are blocking
2) Multithreaded application can't take advantage of multiprocessing.

Teacher's Signature.....

2) Kernel level threads.

① Thread management is done by the kernel as it is directly supported by O.S.

② The kernel maintains context information for the process as a whole and for individual threads within the process.

③ Scheduling by the kernel is done on a thread basis.

④ The kernel performs the following
   a) Thread creation
   b) Scheduling and management in kernel space.

⑤ Kernel threads are generally slower to create and manage than the user threads.

Adv.

① If one thread in a process is blocked, the kernel can schedule another thread of the same process.

② Kernel can simultaneously schedule multiple threads from the same process

Dis.

① Transfer of control from one thread to another within same process requires a mode switch to the kernel.

② → Slower to create than user threads.

Thread Cancellation.

It is the task of terminating a thread before it has completed. for ex:- if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

Teacher's Signature.....

Date
Page

Scheduling Algorithmy.

Way of selecting a process from Ready Queue and put it on the CPU

(1) First come first Serve Algorithm (FCFS)

a) Jobs are executed on first come first serve basis
b) Easy to understand and Implement
c) Poor in performance as avg. WT is high.

Adv.
a) better for long processes.
b) NO Starvation and Simple method.

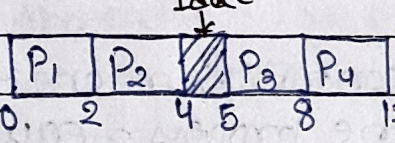
DM.
a) Every process has to wait for its turn.
b) Throughput is not emphasized

Q.

Process	AT	ET	CT	TAT	WT	RT
P_1	0	2	2	2	0	0
P_2	1	2	4	3	1	1
P_3	5	3	8	3	0	0
P_4	6	4	12	6	2	2.

TAT = WT + ET

Gantt. Chart



Criteria :- AT
Mode :- Non Primitive

Avg. TAT = $\frac{2+3+3+6}{4} = 3.5$

Avg. WT = $\frac{0+1+0+2}{4} = 0.75$

CPU Utilization = $\frac{11}{12} \times 100$

$\Rightarrow 91.67\%$

Teacher's Sig. ature

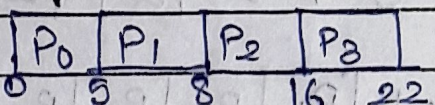
A gantt. Chart is horizontal bar chart used in project management for visual representation.

Date
Page

Q.

Proc.	AT	BT	CT	TAT	WT
P_0	0	5	5	5	0
P_1	1	3	8	7	4
P_2	2	8	16	14	6
P_3	3	6	22	19	13

Gantt. Chart.



$$\text{Avg. TAT} = \frac{5+7+14+19}{4} = 11.25$$

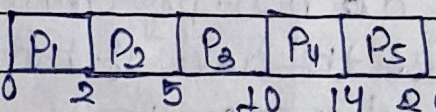
$$\text{Avg. WT} = \frac{0+4+6+13}{4} = 5.75$$

$$\text{CPU Utilization} = \frac{22}{22} \times 100 = 100\%$$

Q.

Proc.	AT	BT	TAT	WT	CT
P_1	0	2	2	0	2
P_2	1	3	4	1	5
P_3	2	5	8	3	10
P_4	3	4	11	7	14
P_5	4	6	16	10	20

Gantt. Chart



$$\text{Avg. TAT} = \frac{2+4+8+11+16}{5} = 8.2$$

$$\text{Avg. WT} = \frac{0+1+3+7+10}{5} = 4.2$$

$$\text{CPU Utilization} = \frac{20}{20} \times 100 = 100\%$$

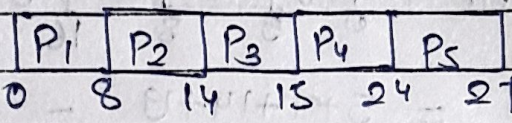
Teacher's Signature.....

Date
Page

Q

Proc.	B.T	AT	TAT	WT	CT
P1	8	0	8	0	8
P2	6	0	14	8	14
P3	1	0	15	14	15
P4	9	0	24	15	24
P5	3	0	27	24	27

Gantt. Chart.



$$\text{Avg. TAT} = \frac{8+14+15+24+27}{5} = 17.6$$

$$\text{Avg. WT} = \frac{0+8+14+15+24}{5} = 12.2$$

$$\text{CPU utilization} = \frac{27}{27} \times 100 = 100\%$$

(2) Shortest Job First (SJF) Scheduling Algorithm

- Shortest job first (SJF) is a scheduling Algorithm that selects the waiting process with the smallest execution time to execute next.
- SJF is a non-preemptive algorithm
- SJF has the adv. of having min. avg. WT among all scheduling algorithm.
- It is a greedy algorithm.

DIS.
- It may cause starvation, if shorter processes keep coming.

Adv.
- Throughput is high.

Criteria → BT / ET
Mode → Non-premptive

Teacher's Sig. ature.....

Q

Process	AT	ET	CT	WT	TAT
P_1	0	8	8	0	8
P_2	1	4	20	15	19
P_3	2	2	11	7	9
P_4	3	1	9	5	6
P_5	4	3	16	9	12
P_6	5	2	13	6	8

$$\text{Avg. TAT} = \frac{8+19+6+9+12+8}{6}$$
$$= 10.33.$$
$$\text{Avg WT} = \frac{0+15+7+5+9+6}{6}$$
$$= 7$$

P_1	P_4	P_3	P_6	P_5	P_2
0	8	9	11	13	16	20

Q.

Process	BT	AT
P_1	21	0
P_2	3	0
P_3	6	0
P_4	2	0

P_4	P_2	P_3	P_1
0	2	5	11	32

$$\text{Avg wt} = \frac{0+2+5+11}{4} = 4.5.$$

(3) SRTF (Shortest Remaining Time First)

Similar to SJF but in this Premption of Process takes place in one matching unit of time.

Q

Proc.	AT	ET	CT	TAT	WT
P_1	0	8 1	20	20	0+12
P_2	1	4 3	10	9	0+5
P_3	2	2 1	5	3	0+1
P_4	3	1	4	1	0+
P_5	4	3	13	9	6
P_6	5	2	7	2	0+

$$\text{Avg TAT} = \frac{20+9+3+1+9+2}{6}$$
$$= 7.33$$
$$\text{Avg WT} = \frac{12+5+1+0+6+0}{4}$$
$$= 4.$$

CS CS CS

P_1	P_2	P_3	P_4	P_3	P_6	P_2	P_5	P_1
0	1	2	3	4	5	7	10	13	20

Teacher's Signature.....

Q.

Proc	ET	AT	CT	TAT	WT
P1	21 20	0	32	32	0+11
P2	9 1	1	6	5	0+2
P3	6	2	12	10	4
P4	2	3	5	2	0

Criteria :- ET
mode is Preemptive

C.S C.S

P1	P2	P4	P2	P3	P1
0	1	3	5	6	12	32

(4) Priority Scheduling Algorithm.

- Each process is assigned a priority. Process with highest priority gets executed first and soon.
- Processes with same priority are executed on FCFS basis.
- It can be of both type preemptive as well as Non preemptive, if not mentioned take Non-preemptive

Q.

Process	AT	ET	Priority	CT	TAT	WT
P0	0	5	1	5	5	0
P1	1	9 1	2	14	13	10
P2	2	8	1	22	20	12
P3	3	6	3	11	8	2

- Non-Preemptive
- 3 is high priority

P0	P3	P1	P2
0	5	11	14	22

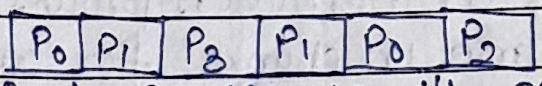
Avg WT
$= \frac{0+10+12+2}{4}$
$= 6$

Avg. TAT
$= 5+13+20+8 = 11.5$

Teacher's Sig. ature

Criteria → Priority
mode → Preemptive or Non-preemptive.

Q. Solve the same with preemptive style



	CT	WT	TAT
P0	14	0+9	14
P1	10	0+6	9
P2	22	12	20
P3	9	0	6

Avg WT = (9+6+12+0)/4
= 6.75

Avg TAT = (14+9+20+6)/4 = 12.25.

Adv. a) The priority of a process can be selected based on memory requirment, time requirment or user requirment.

b) Good mechanism to precisely define importance of each process.

DM.

a) If high priority use up a lot of CPU Time low priority process may starve and postponed indefinitely

(5) Round Robin Scheduling Algorithm (RR)

(1) Based on Time Quantum / TimeShare / Time Slice concept

(2) In RR algorithm each process gets a small unit of CPU Time known as Time Quantum.

(3) After this time has been completed the process is preempted and added to the end of Ready queue

(4) It is also known as FCFS with preemption Scheduling

(5) It is used in Time Sharing and multiuser O.S

Teacher's Signature

Date
Page

→ If there are n no. of processes and Time quantum is Q. Then each process gets 1/n of CPU Time in chunks for at most Q time unit at once, no processes will wait more than (n-1)q time

* Condition
→ When Time Quantum is large then RR algo behaves as FCFS algorithm.
→ If Time Quantum is very small then no. of context switches increases as well as the overall WT and TAT but response Time (RT) will be reduced

Note:-
It is recommended for O.S to select the Time Quantum which is neither big nor small

Adv.
1) Fair Treatment to all the processes
2) Good response time

DNo.
1) Care must be taken in choosing quantum value.
2) Throughput is low if time quantum is too small.

Criteria → Time Quantum
Mode → Preemptive

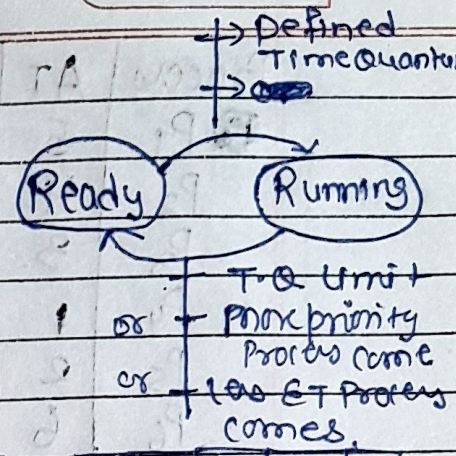
Teacher's Sig. ature

Date
Page

T.Q = 2.

Q.

Process	AT	ET	CT	TAT	WT
A	0	4 2	6	6	2
B	2	7 5 3	22	20	13
C	8	3 1	15	12	9
D	3.5	3 1	16	12.5	9.5
E	4	5 3 1	21	17	12



Ready Queue: A B A C D E B C D E B E B

Running Queue: A B A C D E B C D E B E B

0 2 4 6 8 10 12 14 15 16 18 20 21 22

Q.

Process	AT	ET	CT	TAT	WT	RT
P_1	0	5 8	12	12	7	0
P_2	1	4 2	11	10	6	1
P_3	2	2	6	4	2	2
P_4	4	1	9	5	4	4

T.Q = 2

TAT = CT - AT

WT = TAT - ET

Ready Queue: P_1 P_2 P_3 P_1 P_4 P_2 P_1

0 C.S C.S C.S C.S C.S C.S

Running Queue (Gantt Chart): P_1 P_2 P_3 P_1 P_4 P_2 P_1

0 2 4 6 8 9 11 12

Context Switch = 6

Teacher's Signature.....

T.

Date
Page

Process	AT	ET	CT	TAT	WT
P1	5	25	32	27	22
P2	4	6	21	28	17
P3	3	7	23	30	23
P4	1	9	30	29	20
P5	2	2	6	4	2
P6	6	3	21	15	12

T.Q = 3

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

Running Queue: P4 P5 P3 P2 P4 P1 P6 P3 P2 P4 P1 P3

0 1 4 6 9 12 15 18 21 24 27 30 32 33

Miscellaneous type

Mix of Burst Time (CPU & IO Both)

In this type Processes are CPU as well as IO bound while execution

Mode:- Preemptive or Non-preemptive as per Ques. Criteria : FCFS or Priority as mentioned

Pro.	AT	Priority	CPU.Ti	IO	CPU	CT
P1	0	2	1 0	5	3	10
P2	2	3	3 2 1	3	1	15
P3	3	1	2	3	1	9
P4	3	4	2	4	1	18

Criteria :- Priority
mode :- Preemptive

P2 I/O

P1 | P2 | P3 | P2 | P1 | P3 | P1 | P2 | P4 | P3 | P4

0 1 2 3 5 6 8 9 10 11 13 14 15 16 17 18

P1 I/O

P3 I/O

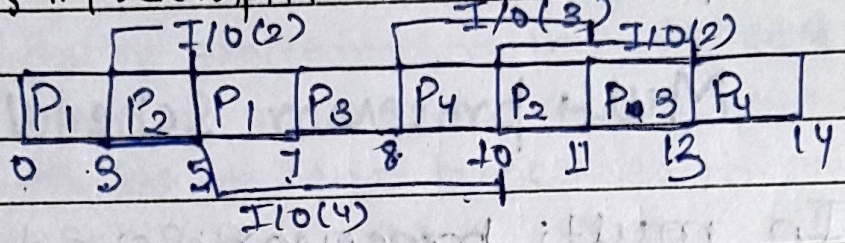
P4 I/O

Teacher's Sig. ature.....

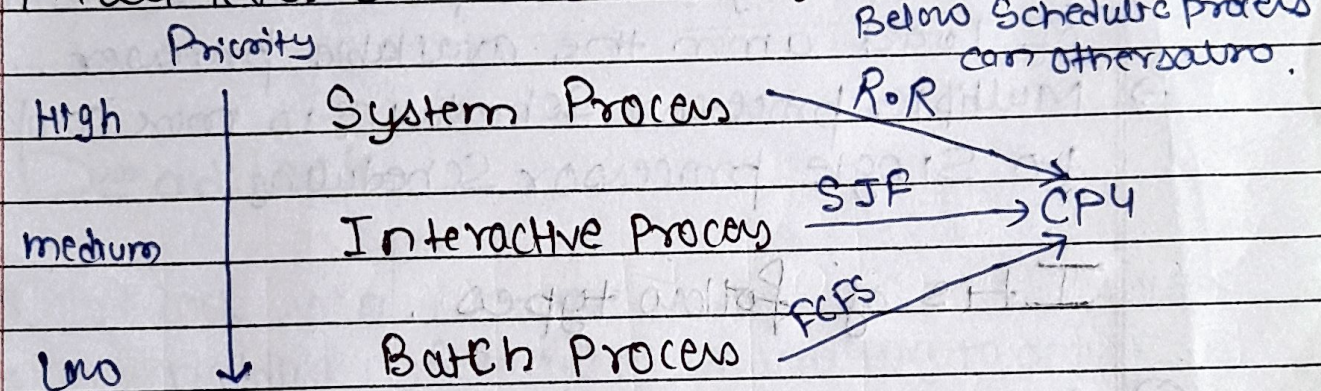
Proc.	CPU Time	I/O Burst	CPU Burst	AT	CT	TAT	WT
P_1	3	2	2	0	7	7	0+0
P_2	2	4	1	0	11	11	3+4
P_3	1	3	2	2	13	11	5+0
P_4	2	2	1	5	14	11	3+1

Criteria :- Arriv. Time.

mode :- Non-Preemptive.



(6) Multilevel Queue Scheduling.



- Multilevel Queue Scheduling divides the processes in two category: foreground (interactive) & background (batch process)
- It allows each process to have its own Ready Queue. The Queues are arranged in multilevel with pre-emption
- It can be customize to meet the Specific requirement of different type of processes.

Teacher's Signature.....

Date
Page

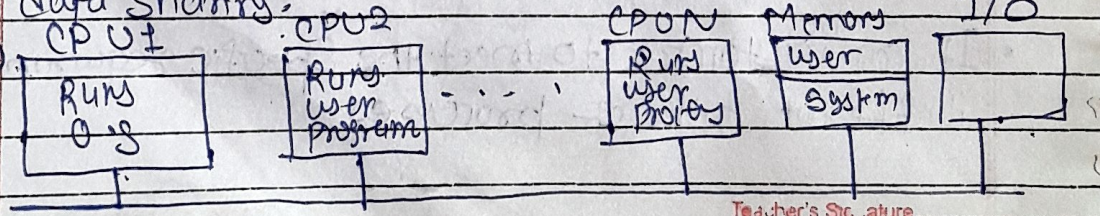
- Each Ready Queue has its own CPU Scheduling algorithm.
- To avoid Starvation we use other variant of it's called as Multilevel feed back Queue. based on feedback concept which increases the priority after considerable time and allows processes to switch b/w the multiple Ready Queue.

Multi processor Scheduling

→ In multi processor Scheduling multiple CPUs are available
→ Load sharing becomes possible as distribution of load amon the available processor.
→ Multiple processor Scheduling is more complex to Single processor Scheduling.

It is of follow types.

① Asymmetric Scheduling :-
In this Scheduling approch follow takes place :-
a) All Scheduling decisions, I/O processing and resource allocation are handled by a Single processor. which is called as master processor.
b) Other rest processor executes only the user code.
c) Simple architecture. which reduces the need of data sharing.



Teacher's Sig. ature.....

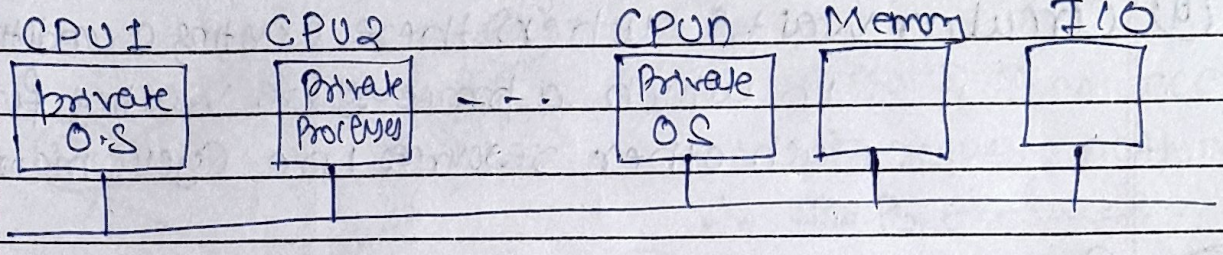
② Symmetric Scheduling:

In this Scheduling approach follow takes place

a) Every processor has its own O.S and self scheduling

b) Every processor may have Common Ready Queue or may have their own private Ready Queue.

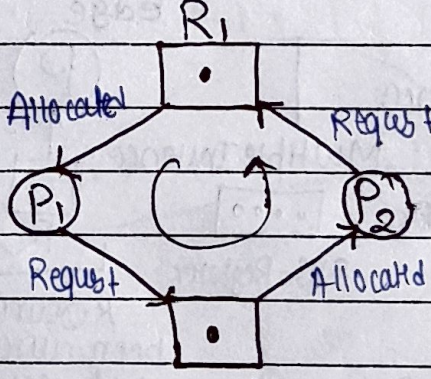
c) The Scheduling proceeded by the scheduler for each processor, examines the ready queue and selects ~~to exec~~ the processes to execute.



Dead lock

- Dead lock is a condition where a set of process each holding a Resource and waiting to acquire another resource held by some other process.

- In simple terms, can be called as Infinite 2 or more way Stamation



Kind of Dead lock diagram

Circular wait ✓
Hold and wait ✓
Availability of R is X
So
Dead lock.

Teacher's Signature.....

Date
Page

(A) Necessary condition which characterised the dead lock.

(1) Mutual exclusion → When only one process at a time can use a resource.

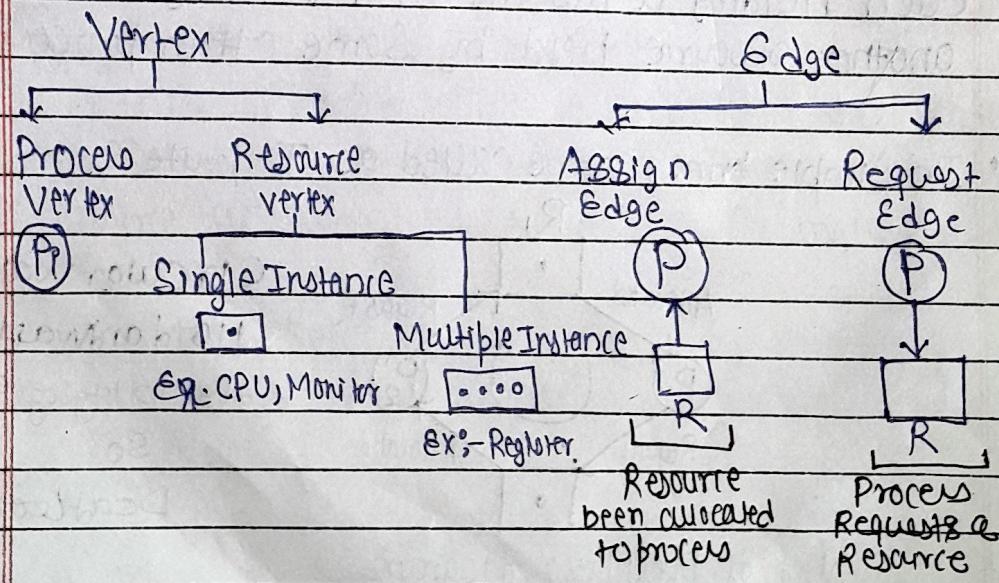
(2) No pre-emption → A resource can be released only voluntary by the process holding it. after completing its execution

(3) hold and wait → A process holding at the one resource and is waiting to acquire additional resource held by another process

(4) Circular wait :- When there existing a condition in which a process are waiting for each other resources in a cyclic manner

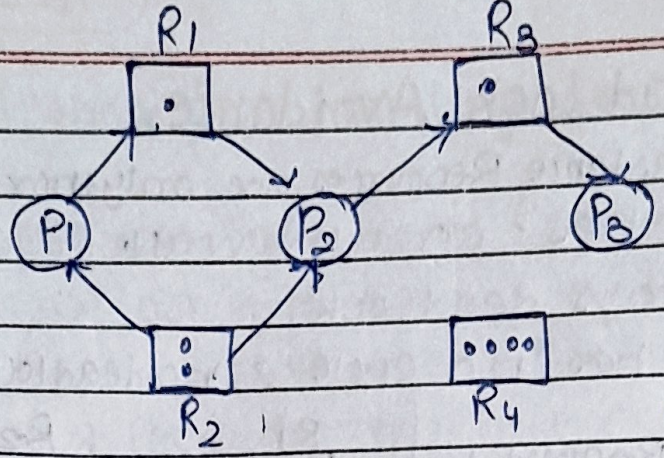
(B) Resource Allocation Graph (RAG)

It is a set of processes and resources which describes the relationship b/w these sets.



Teacher's Sig. ature.....

Q.



	Allocate			Request		
	R_1	R_2	R_3	R_1	R_2	R_3
P_1	0	1	0	1	0	0
P_2	1	1	0	0	0	1
P_3	0	0	1	0	0	0

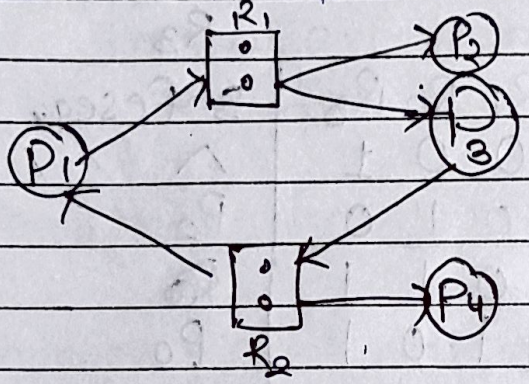
(S1) P_3 doesn't Request anything so it will be executed

(S2) After P_3 execution R_3 is free. ∴ P_2 also gets executed.

(S3) R_1 is now free hence P_1 gets execute. ∴ No deadlock

Safe State:
$\langle P_3\ P_2\ P_1 \rangle$

Q



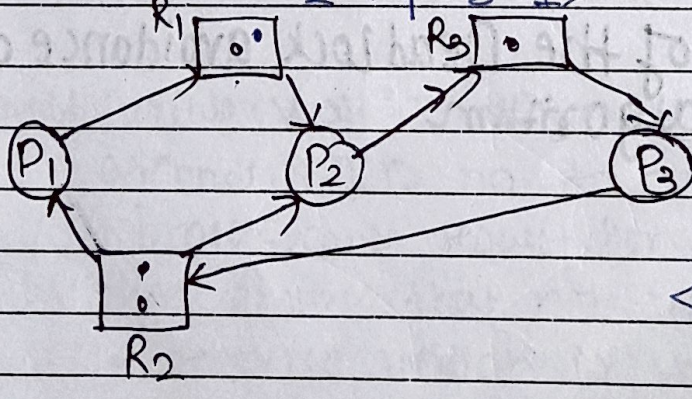
(S1) R_1 has 2 Instances also P_2 requests nothing. ∴ P_2 executed.

(S2) R_2 has 2 Instance, also P_4 requests Nothing ∴ P_4 executed.

(S3) Now P_1 and P_3 can execute due to optimal free Instances of R_1 and R_2 ∴ No deadlock.

Safe State.
$\langle P_2\ P_4\ P_3\ P_1 \rangle$

Q.



It is a deadlock state.
If R_1 has 1 Instance
Let R_1 has 2 Instance.
Safe State Sequence.
$\langle P_1\ P_3\ P_2 \rangle$

Teacher's Signature.....

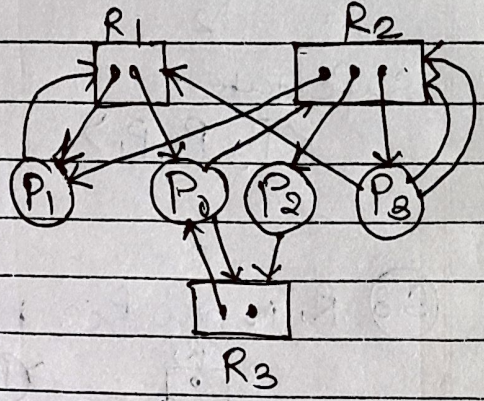
Date
Page

Note:- Dead Lock Avoidance

- If Single Instance Resources are only there.
  also If RAG has circular wait (cycle)
  ∴ Always deadlock.
  also If RAG has no cycle ∴ no deadlock

Q.

Pro.	Alloc. R_1 R_2 R_3	Request R_1 R_2 R_3
② × P_0	1 0 1	0 1 1
③ × P_1	1 1 0	1 0 0
① × P_2	0 1 0	0 0 1
④ × P_3	0 1 0	1 2 0



Current availability

R_1	R_2	R_3	Safeseq.
0	0	1	^
+ 0	1	0	P_2
0	1	1	
+ 1	0	1	P_0
1	1	2	
1	1	0	P_1
2	2	2	
0	1	0	P_3
2	3	2	V

∴ Safesequence is $\langle P_2 P_0 P_1 P_3 \rangle$

RAG is one of the Deadlock avoidance and Detection algorithm

Teacher's Signature

Methods for handling deadlock

1) Deadlock Ignorance (Ostrich Method)
2) Deadlock prevention.
3) Deadlock Avoidance (Bankers Algo) (RAG)
4) Deadlock detection and recovery.

Deadlock prevention

(1) Avoid mutual Exclusion :- Process Synchronization with Sharable Resources hence mutual exclusion not required but it must hold for non-sharable resources.

(2) Avoid hold and wait :- Done by preventing resources to be allocated to a process in one by one manner i.e.) processes must have all the resources before its execution or allow process to request for resource only when process has none of them. Avoid.

(3) No preemption :- If a process is holding some resource request that can't be allocated immediately then process should be preempted

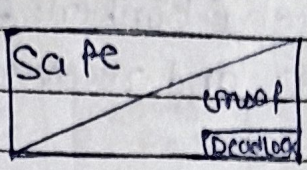
(4) Avoid Circular wait :- Assign a priority or ascending ID no. to each of the resources. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is in use by some other process. Hence no cycle will be formed

Teacher's Signature.....

Date
Page

Dead lock Avoidance

1 Whenever trying to give resource to a process we will check whether there is safe situation or not.



Do not start a process if its demands might lead to dead lock.

1. Safe state :- When the system can allocate resources to each process in some order and avoid a dead lock.
2. Unsafe state :- If the system didn't follow the safe sequence of resource allocation from the beginning, may lead to a dead lock.
3. Dead lock state :- If the system has some, mutual exclusion, hold and wait, No preemption and circular wait condition for some processes then it is dead lock.

Algorithms

① RAG (Resource Allocation Graph)

- Cycle → may be dead lock
- no cycle → no dead lock

} Correct for Single Instance Resources.

Teacher's Sig. ature.....

(2) Bankers Algo

Safety Algo :-

Let work W & finish f[i] be 2 vectors of length m & n.

1. W = Available.
   f[i] = false ∀ i = 0 to n-1
2. find i such that both.
   f[i] = false & need ≤ W
3. W = W + Alloc.
   f[i] = true Go to step 2
4. If f[i] == true ∀ i = 0 to n-1 then stop.

Resource Request Algo.

If Req = k then process Pi wants k instances of Rj

1) If Req ≥ Need, Go to step 2, else max claim execute
2) If Req ≤ Available, Go to step 3, else Pi must wait
3) Update value of Resources
   Available = Available − Req.
   Alloc = Alloc + Req.
   Need = Need − Req.

Teacher's Signature.....

Q Suppose there are 5 processes and 3 Resources ABC max instances are 10,5,7 respectively. Given is the table of Proces Request a resource and the values already allocated. A = 10 3 B = 5 3 C = 7 2

Proc.	Allocation A B C	MAxNeed A B C	Available A B C	Need A B C	
P_1	0 1 0	7 5 3	3 3 2	7 4 3	x 4
P_2	2 0 0	3 2 2	5 3 2	1 2 2	x 1
P_3	3 0 2	9 0 2	7 4 3	6 0 0	x 5
P_4	2 1 1	4 2 2	7 4 5	2 1 1	x 2
P_5	0 0 2	5 3 3	7 5 5	5 3 1	x 3
	7 2 5		10 5 7		

must equal to max. Instances

Safe Sequence.

$\langle P_2, P_4, P_5, P_1, P_3 \rangle$

- If any proces need can't be fullfied then it's deadlock
- When P_2 gets executed first then it will return all allocated Resources
- Current availability ≥ Remaining Need. (Need)
  (Availability) (MaxNeed − Allocations)

Recovery from dead lock

① Process Termination (kill the process)
- abort all deadlocked processes
- abort one process at a time until the deadlock cycle is eliminated

② Resource Preemption → a) Select the resource to preempted
b) rollback the process to some safe state and restart it from that state.

Teacher's Sig. alure.....