Date : / /
Page No.

Unit - (3)

Process :-

(1) A process is basically a program in execution. To put in simple terms, when we write our computer program in a text file and when we execute this program, it becomes a process which perform all the tasks mentioned in the program.

(2) When a program is loaded into the memory and it becomes a process, it can be divided into four sections

(3) It is an instance of a running program.

* Object Program :- Code to be executed is known as object program.
* Data :-

Computer Prog. → In text file. → $ M

↓ Execute

Main Memory → Process

Which Performs all task mentioned In Program.

Stack heap text data.

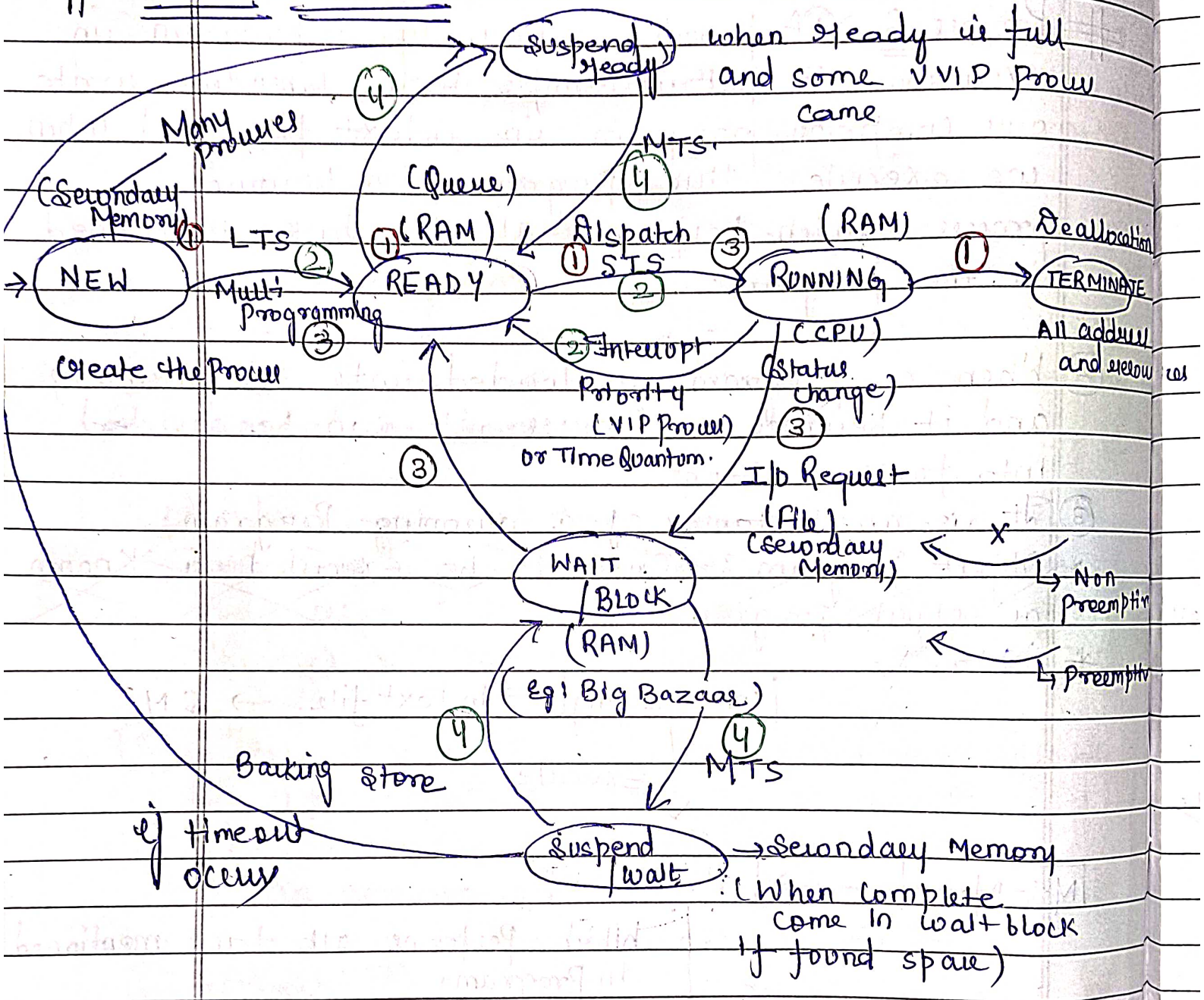
Layout Structure of Process in Main Memory

Stack	→ Contains temp data (func, local Variables, Parameter
heap	→ dyanmically allocate memory to process during run time
Data	← Contains Static Variables & Global Variable
text	→ Program counter & contents of Processor reg

MTS :- Medium Term Scheduler
LTS :- Long term Scheduler
STS :- Short term Scheduler

Date : / /
Page No.

Process States



When process executes, it changes state. Process state is defined as the current activity of the process. Process State Contains five states. Each process is in one of the states.

(1) NEW :- A process that has just been created.

(2) READY :- Ready Processes are waiting to have the processor allocated to them by

ps command → All info about proces.

Date: / /
Page No.

the OS so that they can run.

(3) RUNNING :- The process that is currently being executed. A Running process possesses all the resources needed for its execution, including the processor.

(4) Waiting :- A process that can't execute until some event occurs such as completion of I/O operation. The running process may become suspended by invoking an I/O routine.

(5) TERMINATED :- A process that has been released from pool of executable processes by an OS.

Schedulers :-

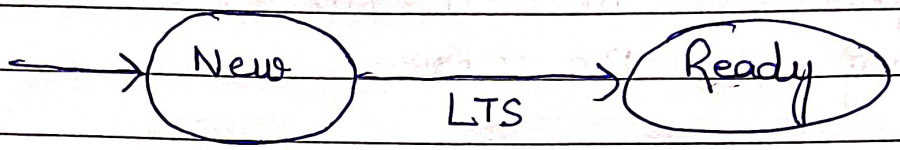
Schedulers are of three types :-

(1) Long term Scheduler :- It is also called Job Scheduler. Long term Scheduler determines which programs are admitted to system for processing. It selects processes from queue and loads them into memory for execution. The primary objective of Job Scheduler is to provide a balanced mix of jobs, such as I/O bound & CPU bound.

It controls the degree of multiprogramming. If the degree of multiprogramming is

Date : / /
Page No.

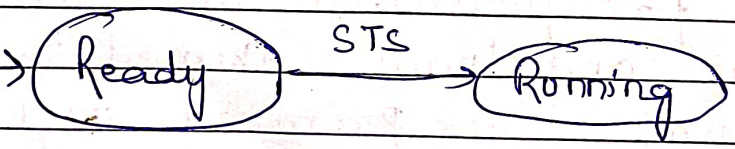
Stable then average rate of process creation must be equal average departure rate of processes leaving the system.



When process changes state from new to ready then there is a long term scheduler (LTS). Time Sharing have no LTS.

(2) Short term Scheduler :- It is also called CPU Scheduler. It is the change of ready state to running state of process. CPU scheduler Selects among processes that are ready to execute and allocates CPU to one of them.

Short term Scheduler also Known as dispatcher, which execute most frequently and makes the grained decision of which process to next execute. Short term Scheduler is faster than Long term Scheduler.

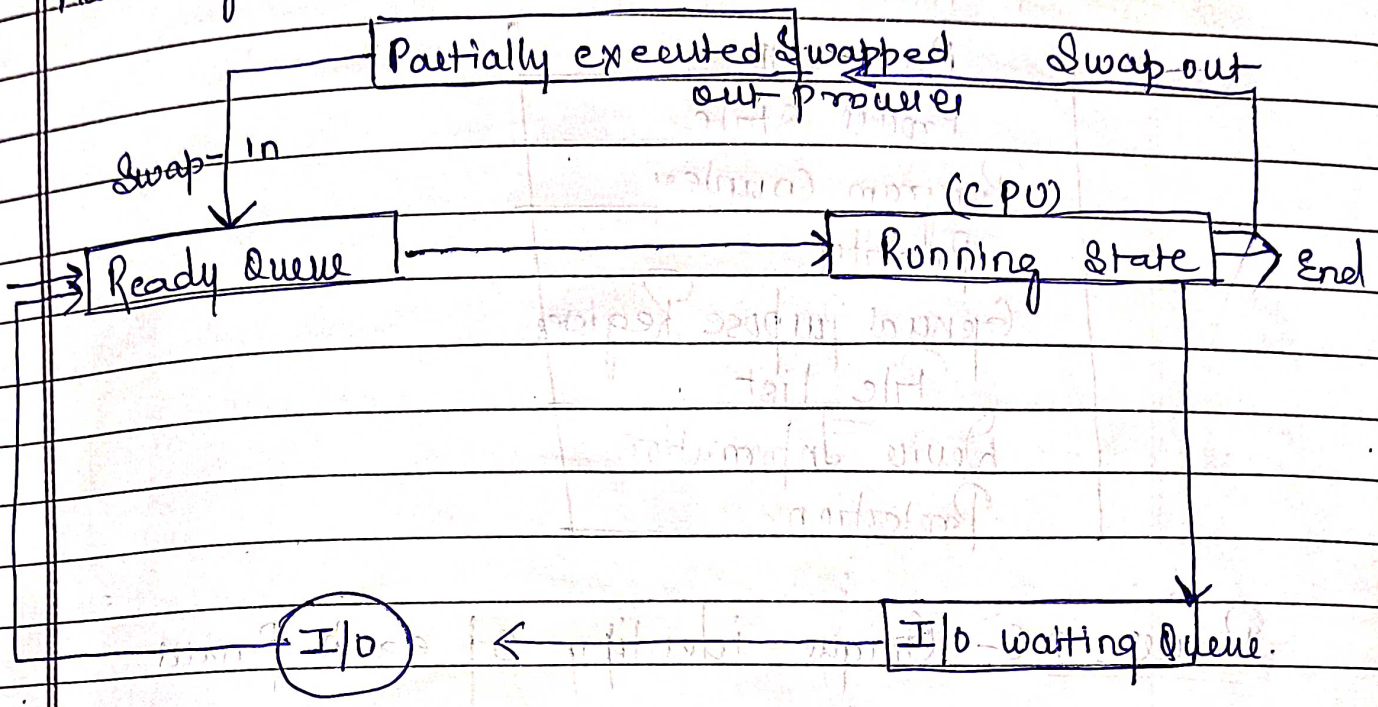


(3) Medium Term Scheduler

MTS is a part of Swapping function. It removes processes from main memory and reduces the degree of

Date : / /
Page No.

multiprogramming. The MTS is in charge of handling the Swapped out processes.



Running process may become Suspended by making an I/o request. Suspended processes cannot make any progress towards completion.

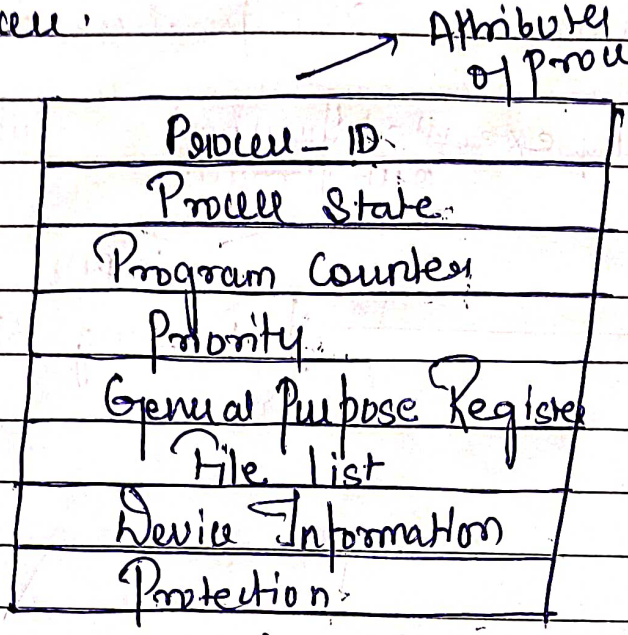
Suspended process is move to Secondary Storage device. Saving the image of a Suspended process in Secondary Storage is called Swapping. and the process is said to be Swapped out or rolled out.

Process Control Block :-

Process control block is a data structure that contains information of the process related to it.

Whenever we create any process then OS create Process control block of each process at

Same time which contains all data related information to the Process.



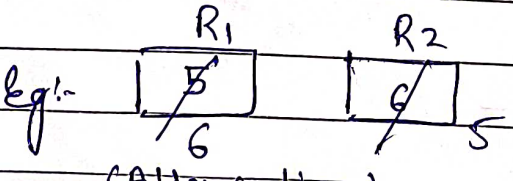
Process_ID :- Unique identity of each Process.

Process States :- Current State of each process

Program Counter :- It is a pointer which points to last instruction

Priority :- Which Process has how much priority is written in this section.

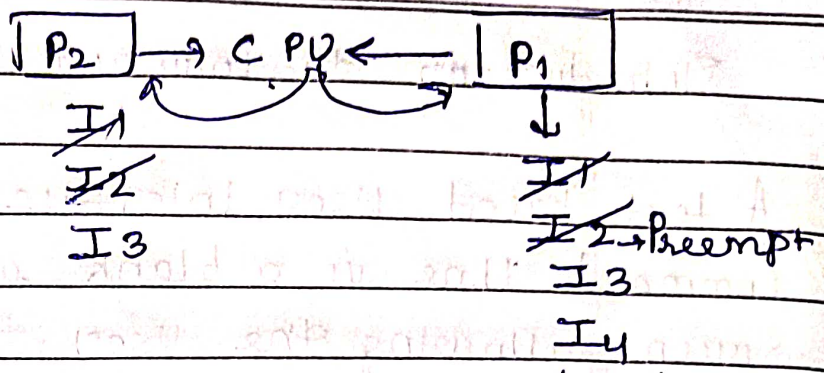
General Purpose Register :- It is used to hold the data that is used during execution.



i++;

File List :- During execution which file is used by any Process.

Date : / /
Page No.



Device Info :- Which device is accessed by any Process.

Protection :- No two Processes can enter in Protective area of other Process.

etc :- It is not mandatory that only this info is available. Vary O.S to O.S.

Date : / /
Page No.

LAB ①

Introduction to Command Line Interface:

A text based User Interface to the computer. The Command Line is a blank and cursor on the screen, allowing the user to type in instructions for immediate execution.

① help : To see all commands that are possible.

② cls : To clear all the commands.

③ cd → change directory (folder)

④ cd.. → To go one step back

⑤ cd Windows → To goto this folder

⑥ Now if we want to go directly to C drive directly by closing all folders write
cd/

⑦ dir → What files and folders are created at what time.

⑧ How to make folders :-
↳ ~~mkdir~~ mkdir → make folder
↳ rmdir → Remove folder.

Eg: mkdir model town → two folders are created.
If we want to create folders with spacing do mkdir "model town"

↳ cut paste :- move diet.docx dod.

↳ Copy paste :- Copy diet.docx dot

Date : / /
Page No.

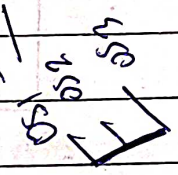
↳ Rename the file :- move dlet.docx folder.docx

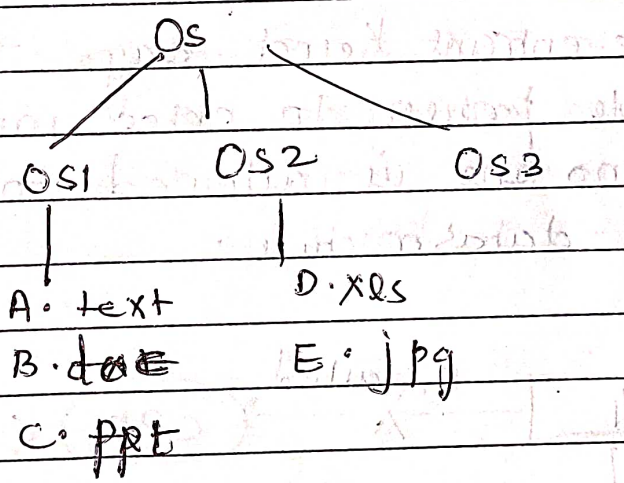
↳ delete → del file.docx

↳ To delete non empty folder → rmdir /s dod

↳ To copy file to some other location.

→ To see battery health
Type :- Powercfg/energy

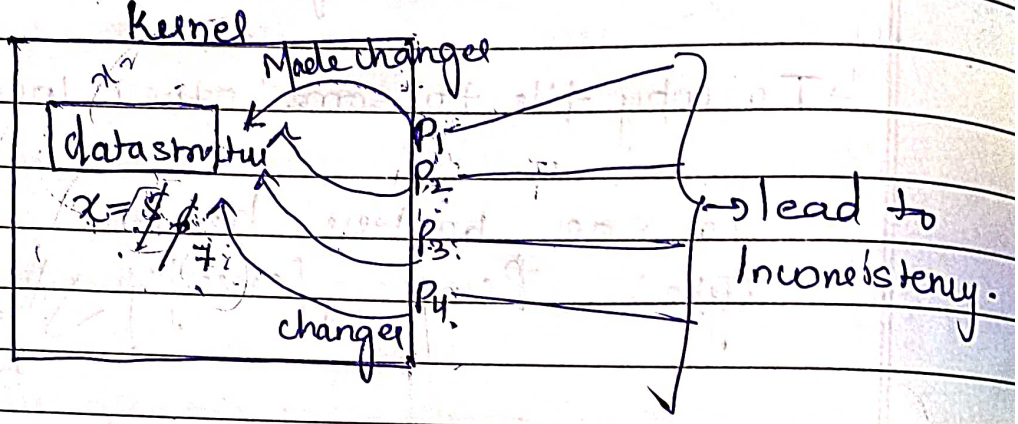




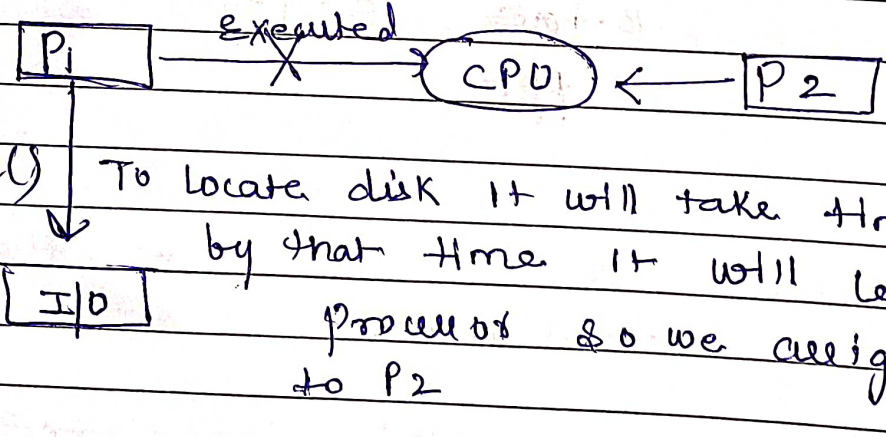
Date : / /
Page No.

Re-entrant Kernel

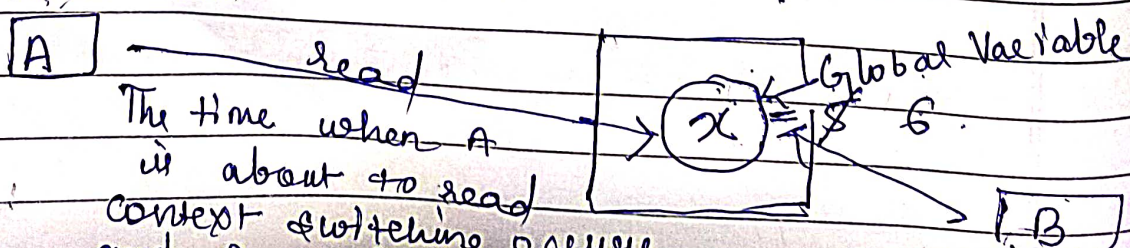
1) It is not any type of OS nor it is part of structure.
2) This we can say is a feature that whether we want to make our Kernel Re-entrant in nature or not.
3)



4) So re-entrant Kernel says I will allow multiple process to enter my Kernel space but no one is allowed to change my Global datastructure.
5)



6) Why we are making Kernel re-entrant? Example:- What happen if we will not make our Kernel re-entrant?
Kernel mode



B changes its value to 6 again Date: Page No.
switching occurs and A find 6 but original
value is 5. So A is reading two diff values
at two diff time.

So If we will not make our Kernel re-entrant
this inconsistency problem will occur. So If
we are allowing Multiple processes then Kernel
need to be re-entrant in nature.

So Now Another Question is How to make Kernel
Re-entrant?

↳ We have to use re-entrant functions
and for those functions only multiple process
are allowed. and re-entrant fun are to
be designed in such a way that process can
modify only local variable not global
variable.

↳ But strictly adhering to re-entrant functio
is not a complete Soln. for this we can use
a locking mechanism. If any non re-entrant
fn is there in code So only one process
is allowed to execute.

Re-entrant fn.
↑ P_2 P_3 P_4
P_1
only permit
to modify
local variable

Non-Re-entrant
↓
They will allow
to modify
global also
↓
So stop
this locking
is used...

Date : / /
Page No.

System call

1) If we want to access any function of OS then we have to enter in Kernel mode. To access the Service of Kernel the concept System call is used.

2) It is one programatic way by which we can shift from user mode to Kernel mode.

3) Like for Eg. I want to write C Program for print the Value of 2+2
So in this Value 2+2 is enter in user mode but to print output on monitor we have to take help from Kernel.
Eg: To access file in order to do any manipulation and file is in Hard disk for that also I have to request Kernel because all these resources are goverd by OS.

4) In LINUX we have 100 of System call and in windows we have approx 700 System call.

5) For Eg. Printf() → It is not a direct System call but a library or func to access write() System call.

WAP to print 2+2

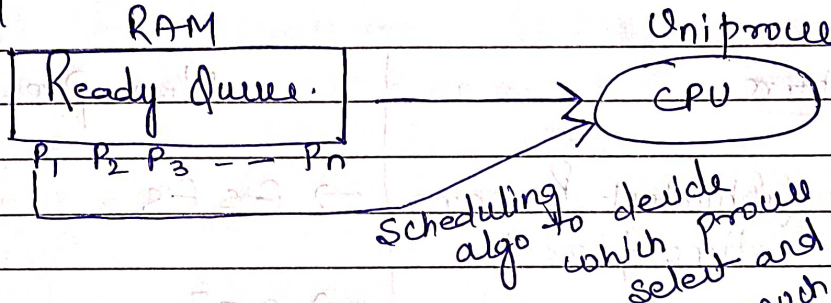
Printf().

Date : / /
Page No.

Unit - (3)

Scheduling Algorithm

Scheduling Algorithm is a way of selecting a process from ready queue and put it in the CPU
* Means according to the degree of multiprogramming we tried to keep multiple processes in the ready queue.

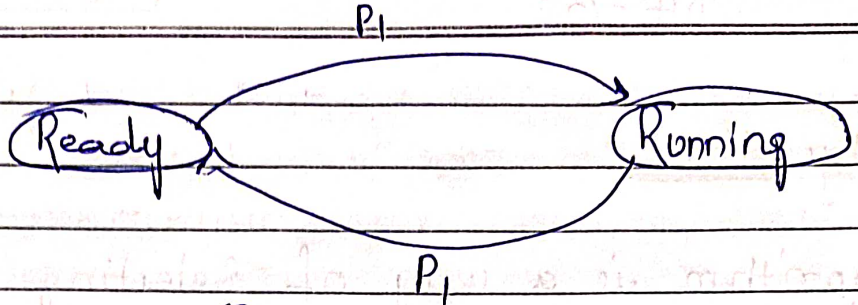


There Scheduling Algorithms are of two types:-

(1) Pre-emptive
This is the way in which for eg if we have send one process from ready state to running state and in between one VVIP process came and raise interrupt for CPU execution so we have send process P1 back to ready state and start executing VVIP process

Non Pre-emptive
This is the way in which once any process is send to running state it cannot be stop in b/w before final completion.

Date : / /
Page No.



① Time Quantum
② Priority
③ less time
} Reason of Pre-emption

Pre-emptive	Non pre-emptive
→ * SRTF (Shortest Remaing time first)	→ FCFS
→ LRTF (Longest Remaining time first)	→ SJF
→ ** Round Robin	→ LJF
→ Priority based (Both Preemptive and Non Preemptive)	→ HRRN (Highest Response Ratio Next)
	→ Multilevel Queue
	→ Priority

Various times in CPU Scheduling

① Arrival time :- It is amount of time The time at which process enters the ready queue or state
② Burst time :- It is duration Time required by a process to get execute on CPU

Date :
Page No.

For eg: If I want to deposit money in bank and for that I enter in bank at 10:AM so 10:AM is my arrival time and time to deposit money depend on the person who is going to update and process my transaction for eg that person takes 10 min to deposit so 10 min is my burst time.

→ It is amount of time.

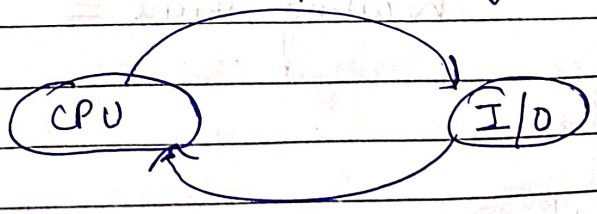
(3) Completition time :- Time at which process complete its execution. (for eg At 12 I came out from bank so completition time is 12.)

(4) Turn around time :- { (Completition time - Arrival time) }
(12 - 10) = 2 hrs.

(5) Waiting time :- { Turn around time - Burst time }
20 - 10 = 9 ✗

(6) Response time :- (The time at which a process gets CPU first time) — (Arrival time).

So Processes are basically of two types:
→ CPU bound
↳ I/O bound.



Various Scheduling Criteria

Date : / /
Page No.

These are the Parameters on which we can judge which Scheduling algorithm is best. These algorithms will help us to determine the efficiency of each algorithm

(1) CPU Utilization :- "We want to keep CPU as busy as possible".

CPU utilization is the average function of time, during which processor is busy. The load on the system affects the level of utilization that can be achieved. CPU utilization may range from 0% to 100%. In time shared system, CPU utilization is primary consideration.

(2) Throughput :- If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per unit time, called throughput.

(3) Waiting time :- The avg period of time a process spends waiting in ready queue.

Waiting time = Turn around time − Execution time or Burst time

(4) Now what is turnaround time ??

~~From~~ The interval from time of submission of a process to time of completion is TAT.

Date : / /
Page No.

TAT is the sum of periods spent waiting to get memory, waiting into ready queue, executing on CPU and doing I/O.

Turn around time = Completion time - Arrival time

(5) Response time :-

Response time is the time from submission of a request until the first response is produced.

(The time at which a process gets CPU first time) - (Arrival time.)

(6) Priority :- Give Preferential treatment to process with higher priorities.

7) Balanced utilization :- Utilization of memory, I/O devices and other system resources are also considered. Not only CPU utilization considered for performance.

(8) Fairness :- Avoid process from starvation. All processes must be given equal opportunity to execute.

Date : / /
Page No.

1) First Come first Serve Scheduling (FCFS)

Criteria is Arrival time and Mode is non-Preemptive.

In FCFS whichever process enters the ready queue first is executed first.

Advantages :-

- It is Simple to Understand
- It can be easily implemented using queue data structure
- ~~It does~~

Disadvantages :-

- It does not consider the priority or burst time of the process
- It suffers from convoy effect.

Convoy Effect :- Consider process with higher burst time arrived before the process with smaller burst time.

Then smaller process have to wait for a long time for longer process to release the CPU.

Que Consider the set of 5 process whose arrival and burst time are given below :

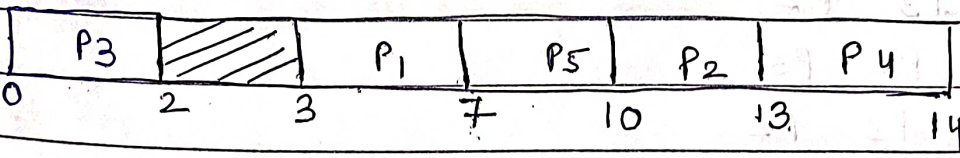
Date : / /
Page No.

Process_ID	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

Step ① Lets first draw table to calculate all the possible times

Process_ID	Arrival time	Burst time	Completion time	TAT	WT	RT
P1	3	4	7	4	0	0
P2	5	3	13	8	5	5
P3	0	2	2	2	0	0
P4	5	1	14	9	8	8
P5	4	3	10	6	3	3

Step ② Before Calculating Completition time, TAT, WT and response time first draw Gantt Chart



Grant chart

AVG Waiting time = (0+5+0+8+3)/5

Now Calculate All the above mentioned times.

Date : / /
Page No.

Turn Around time = Completition time - Arrival time

Waiting time = TAT - Burst time

Response time = (First time process get CPU) - Arrival time

Gate Questions

GATE CS 2012

Que Consider the 3 processes P_1, P_2 P_3 shown in the table

Process	Arrival time	Time Unit req
P1	0	5
P2	1	7
P3	3	4

The Completion time order of the 3 processes under the policies FCFS are :-

1) P1 P2 P3
2) P1 P3 P2
3) P1 P2 P3
4) P1 P3 P2

Correct Ans :- 3)
P1 P2 P3

Date : / /
Page No.

(2) SJF (Shortest Job first) → Non Pre-emptive

Till now we were Scheduling the processes according to their arrival time (In FCFS scheduling). However in SJF scheduling algorithm, we Schedule the process according to their burst time.

In SJF Scheduling, the process with the lowest burst time, among the list of available processes in ready queue is going to be scheduled next.

Advantages

- Maximum throughput
- Minimum average waiting and turn around time

Disadvantages

- May suffer with the problem of starvation
- It is not implementable because the exact burst time for a process can't be known in advance.

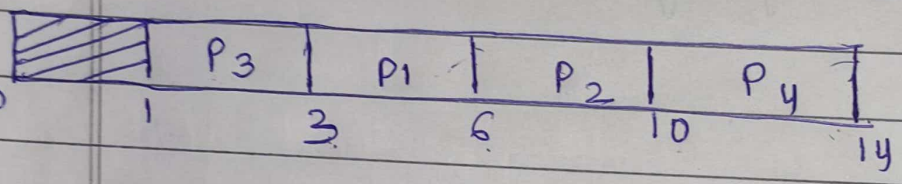
Ques

Process No	Arrival time	Burst time
P_1	1	3
P_2	2	4
P_3	1	2
P_4	4	4

Criteria = Burst time
Mode = Non Preemptive

Process No	Arrival time	BT	CT	TAT	WT	RT
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	2	3	2	0	0
P_4	4	4	14	10	6	6

Gantt chart :-



Ready Queue

→ P_1 ~~P_3~~

→ ~~P_1~~ P_2

→ ~~P_2~~ P_4

Que

Process	AT	BT
P_1	1	7
P_2	2	5
P_3	3	1
P_4	4	2
P_5	5	8

Solve using non preemptive SJF

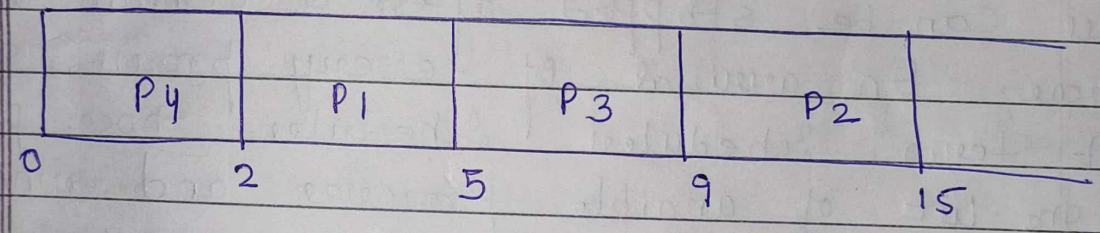
Date : / /
Page No.

Que

Process	Burst time
P_1	3 → 1
P_2	6 → 1
P_3	4
P_4	2

Arrival time is 0 for all processes.

Process	Burst time	AT	CT	TAT	WT	RT
P_1	3	0	5	5	2	2
P_2	6	0	15	15	9	9
P_3	4	0	9	9	5	5
P_4	2	0	2	2	0	0



Ready Queue

→ P1 P2 P3 ~~P4~~

→ ~~P1~~ P2 P3

→ P2 ~~P3~~

→ ~~P2~~

(3) Shortest Remaining time first (SJF with pre-emption)

↳ The preemptive version of SJF scheduling is known as Shortest Remaining time first (SRTF).

↳ With the help of SRTF algorithm, the

Date : / /
Page No.

Process having the smallest amount of remaining burst time will be selected first to execute. So basically in SRTF, the processes are scheduled according to shortest remaining time.

↳ However, the SRTF algorithm involves more overheads than the SJF algorithm; because in SRTF, OS is required frequently in order to ~~maintain~~ monitor the CPU time of the jobs in the READY queue and to perform context switching.

↳ In the SRTF algorithm the execution of any process can be stopped after a certain amount of time. On arrival of every process, the short term scheduler schedules those processes from the list of available processes and running processes that have the least remaining burst time.

↳ After all the processes are available in the ready queue, then no preemption will be done and then the algorithm will work the same as SJF algo.

Advantages

The main advantage of SRTF algorithm is that it makes the processing of the jobs faster than the SJF algorithm, if mentioned it's overhead charges are not counted.

Date : / /
Page No.

Disadvantages

In SRTF, the Context Switching is done a lot more times than in SJF due to more consumption of the CPU valuable time for processing. The consumed time of CPU then adds up to the processing time and which then diminishes the advantage of fast processing of this algorithm.

GATE Question

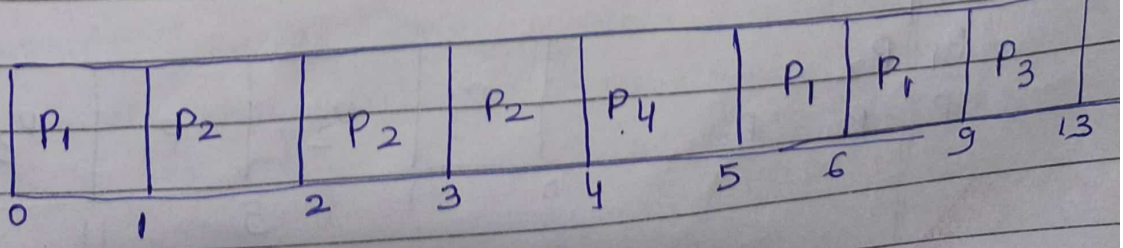
Process_No	Arrival Time	Burst Time
P_1	0	5
P_2	1	3
P_3	2	4
P_4	4	1

Criteria :- Burst Time
Mode :- preemption

Ready Queue
P_1
P_1 P_2
~~P_1 P_2 P_3~~
P_1 P_3 ~~P_4~~

Process_No	AT	BT	CT	TAT	WT	RT
P_1	0	~~5 4 3~~ 0	9	9	4	0
P_2	1	~~3 2 1~~ 0	4	3	0	0
P_3	2	~~4~~ 0	13	11	7	7
P_4	4	~~1~~ 0	5	1	0	0

Gantt Chart :-



UGC NET July-18

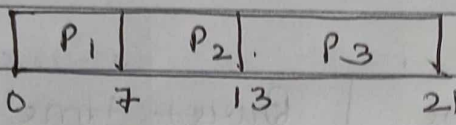
Date : / /
Page No.

Que 1

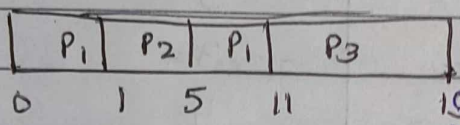
Process	Arrival time	Burst time
P_1	0	~~7~~ 6
P_2	1	~~4~~ ~~3~~
P_3	2	8

The Gantt chart for preemptive SJF Scheduling algorithm is ?

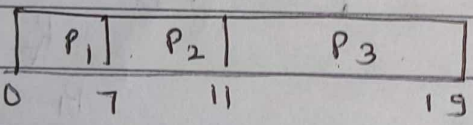
1)



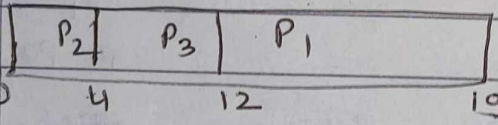
2)

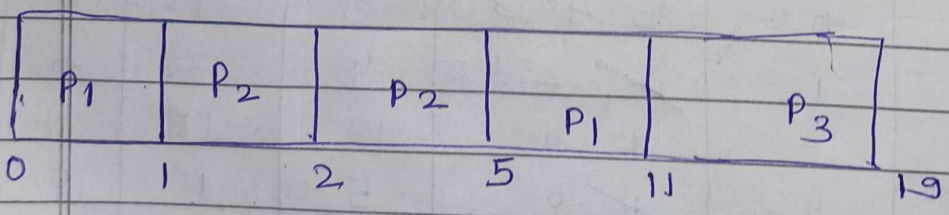


3)



4)



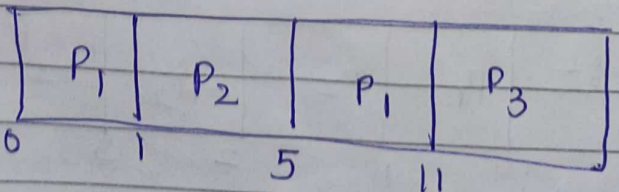


Ready Queue.

P_1

P_1 P_2

P_1 P_2 P_3



So option 2 is correct

Date : / /
Page No.

(4) Round Robin Scheduling

- Round Robin CPU Scheduling is the most important CPU Scheduling Algorithm which is ever used in the history of CPU Scheduling Algorithms.
- Round Robin CPU Scheduling Uses Time Quantum (TQ).
- The time quantum is something which is removed from the burst time and it's the chunk of process to be completed.
- Time Sharing is the main emphasis of the algorithm. Each step of this algorithm is carried out cyclically. The System defines a Specific time slice, known as time quantum.

Working :-

- First the processes which are eligible to enter the ready queue enter the ready queue.
- After entering, the first process present in ready queue is executed for a time quantum.
- After execution is complete, the process is removed from ready queue.
- Even now the process requires some time to complete its execution, the the process is added to ready queue at last position.

Date : / /
Page No.

Advantages :-

↳ A fair amount of CPU is allocated to each J
↳ As it doesn't depend on burst time, it ca
truly implemented in the System.

↳ It is not affected by the convoy effect or
starvation problem as occurred in FCFS.

Disadvantages :

↳ Round Robin CPU Scheduling approach take
longer to Swap contexts.

↳ Time quantum has a significant impact on
its performance.

↳ The Procedure cannot have priorities establi

Ques.

Process No	Arrival time	Burst time
P1	0	5
P2	1	4
P3	2	2
P4	4	1

Solve Using RR CPU Scheduling with TQ = 2

Sol"

Criteria :- Time Quantum

Mode :- Preemptive

TQ = 2

Date : / /
Page No.

Process NO	Arrival time	Burst time	CT	TAT	WT	RT
P1	0	~~5~~ ~~3~~ 1	12	12	7	0
P2	1	~~4~~ ~~2~~ 0	11	10	6	1
P3	2	~~2~~ 0	6	4	2	2
P4	4	~~1~~ 0	9	5	4	4

Ready Queue | ~~P1~~ | ~~P2~~ | ~~P3~~ | ~~P1~~ | ~~P4~~ | ~~P2~~ | P1 |

Running Queue | P1 | P2 | P3 | P1 | P4 | P2 | P1 |
0 2 4 6 8 9 11 12

Gate CS 2020

Ques Consider the following set of processes, assumed to have arrived at time 0. Consider the CPU scheduling algorithms SJF and Round Robin (RR). For RR assume that the processes are scheduled in the order P1 P2 P3 P4

Process	Burst time
P1	8
P2	7
P3	2
P4	4

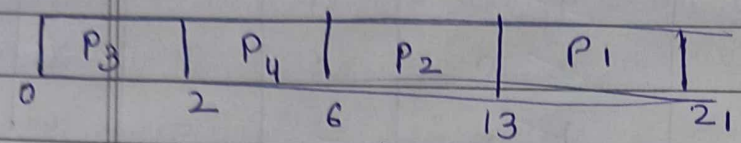
If the time quantum for RR is 4 ms, then the absolute value of difference b/w avg TAT of SJF and RR (round off to 2 decimal places) is

A) 5.0 B) 5.25
C) 5.50 D) 5.75

Ans (B)

Date : / /
Page No.

Acc to SJF



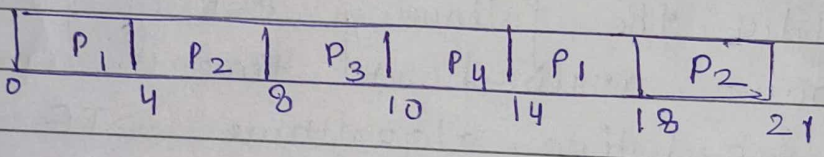
Avg TAT $((21-0)+(13-0)+(2-0)+(6-0))/4$

$= 10.5$

Acc to RR

Ready Queue ~~P1 P2 P3 P4 P1 P2~~

Running Queue



Avg TAT is

$((18-0)+(21-0)+(10-0)+(14-0))/4$

$= 15.75$

Hence

$|10.5 - 15.75|$

$= 5.25$ Ans

GATE

Date : / /
Page No.

Que How many context switch will take place in Round Robin Scheduling method with TQ = 3 for the data given below.

Process	AT	BT
P1	5	6
P2	8	7
P3	3	8
P4	6	3
P5	2	2
P6	4	4

Sol" TQ = 3

Process	AT	BT
P1	5	6 3 0
P2	8	7 4 1 0
P3	3	8 5 2 0
P4	6	3 0
P5	2	2 0
P6	4	4 1 0

Ready Queue

P5	P3	P6	P1	P4	P3	P2	P6	P1	P3	P2	P2

Running Queue

P5	P3	P6	P1	P4	P3	P2	P6	P1	P3	P2	P2

2 4 7 10 13 16 19 22 23 26 29 32 33

Total 11 Context Switching.

Date : / /
Page No.

(5) Priority Scheduling

Priority Scheduling is a method of Scheduling proc that is based on priority. In this algorithm, the Scheduler selects the proces to work as per the priority no. assigned.

The Process with higher priority should be carried out first, whereas Jobs with equal priorities are carried out on a FCFS basis

Types of priority Scheduling

(a) Pre-emptive Scheduling

In pre-emptive Scheduling the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

b) Non- Pre-emptive Scheduling

In this type of Scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various

Date : / /
Page No.

hardware platforms.

Characteristics of Priority Scheduling

* A CPU algorithm that schedules process based on priority.
* It used in OS for performing batch process.
* If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED Basis.
* In priority scheduling, a number is assigned to each process that indicate its priority level.
* In this type of Scheduling algorithm if a newer process arrives, that is having a higher priority than the currently running process, then currently running process is preempted

Que (Peemptive priority Scheduling)

Priority	Process_NO	AT	BT
10	P1	0	5
20	P2	1	4
30	P3	2	2
40	P4	4	1

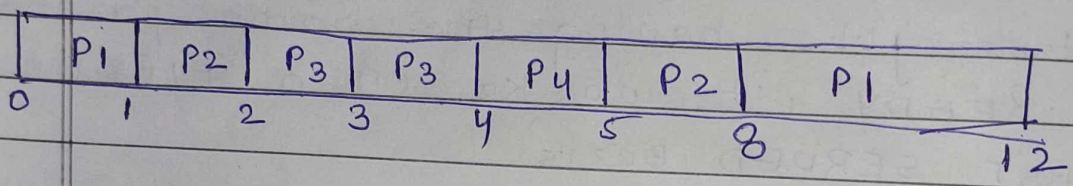
NOTE: Higher the No. higher the priority

Mode :- Pre-emptive

Criteria :- Priority

Priority	Process_No	AT	BT	CT	TAT	WT
10	P1 ✓	0	~~5~~ 4 0	12	12	7
20	P2 ✓	1	~~4~~ 3 0	8	7	3
30	P3 ✓	2	~~2~~ 1 0	4	2	0
40	P4 ✓	4	1 0	5	1	0

Gantt chart



$$\text{AVG TAT} = \frac{22}{4} = 5.5$$

$$\text{AVG WT} = \frac{10}{4} = 2.5$$

Ques Non Pre-emptive Priority Scheduling

PID	Priority	AT	BT
P1	2	0	3
P2	6	2	5
P3	3	1	4
P4	5	4	2
P5	7	6	9
P6	4	5	4
P7	10	7	10

Criteria: Priority

Mode: Non preemptive

[Here the number higher the Priority]

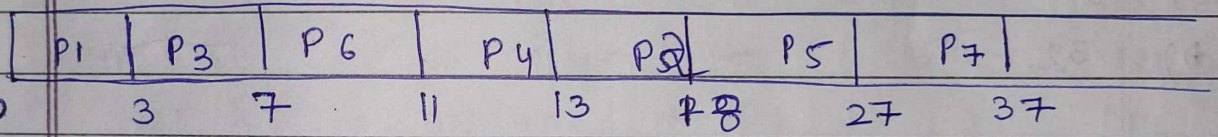
Date : / /
Page No.

WT
7
3
0
0

Soln

PID	Priority	AT	BT	CT	TAT	RT	WT
P1	2	0	3	3	3	0	0
P2	6	2	5	18	16	11	11
P3	3	1	4	7	6	2	2
P4	5	4	2	13	9	7	7
P5	7	6	9	27	21	12	12
P6	4	5	4	11	6	2	2
P7	10	7	10	37	30	20	20

Gantt chart



Ready Queue.

P1

P2 P3

P2 P4 P5 P6 P7

y

emptive

Gate CS-2017

Ques Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds) and priority (0 is highest priority) shown below. None of the process processes have I/O burst time.

Date : / /
Page No.

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

The average waiting time (in milliseconds) of all the process using pre-emptive priority scheduling algorithm is :

A) 29
B) 30
C) 31
D) 32

Ans : A

Process	AT	BT	Priority
~~P1~~	0	~~11~~ ~~9~~ 0	2
~~P2~~	5	~~28~~ ~~24~~ ~~21~~ 0	0
~~P3~~	12	2	3
~~P4~~	2	~~10~~ ~~7~~ ~~3~~ 0	1
~~P5~~	9	16	4

P1	P4	P2	P2	P4	P1	P3	P5
0	2	5	9	12	33	40	49	51	67

Ready Queue :
P1
P1 P4 P2

Deadlock

1) Deadlock is a situation where a process is waiting for the resource held by some other waiting process and there is no progress for watting process.

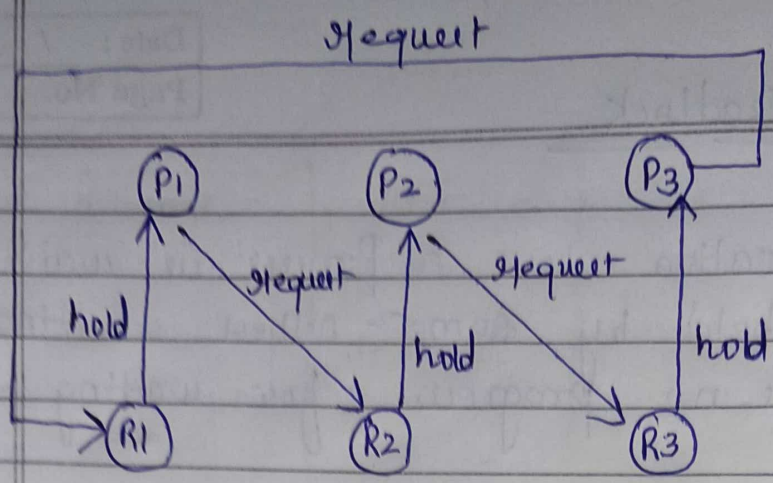
2) A deadlock is a situation where each of the computer process waits for a resource which is being assigned to some other process. In this situation, none of the process gets executed since the resource it need is held by some other process which is also waiting for some other resource to be released.

3) For Eg:- Let us assume there are three processes P1 P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.

- After Sometimes P1 demands for R2 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2 also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.

- In this scenario, a cycle is formed among three processes. None of the process is progressing and they all are waiting. The computer becomes Unresponsive since all the processes got blocked.

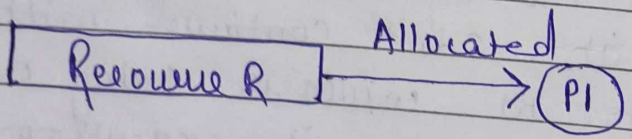
4) If a Process is in waiting state and is unable to change its state because the resource req by the Process is held by some other waiting process.



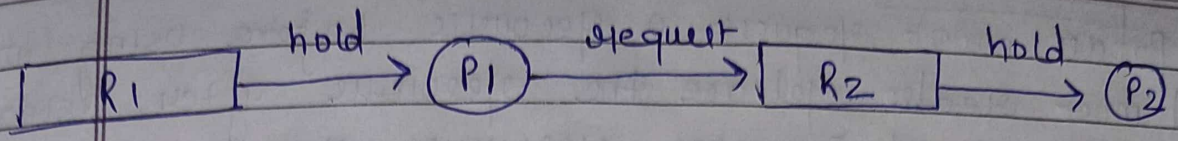
Necessary Conditions of Deadlock :-

There are four different conditions that result in deadlock. These four conditions are also known as coffman conditions.

1) Mutual Exclusion :- A resource can be held by only one process at a time. In other words, if a process P1 is using some resource R at a particular unit of time ; then some other process P2 can't hold or use the same resource R at that time. The process P2 can make a request for that resource R but it can't use that resource simultaneously with process P1.

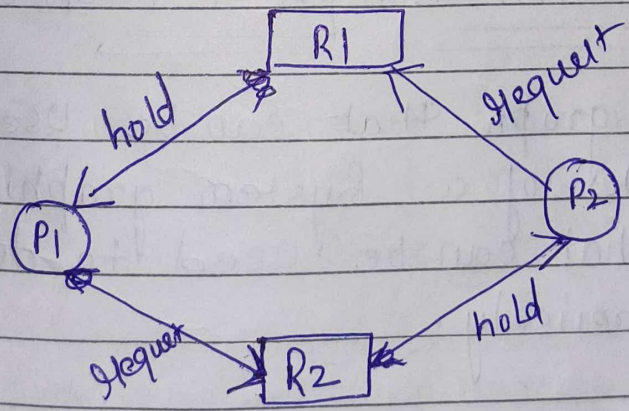


2) Hold and wait :- A process can hold a resource and at the same time can request for other resources that are being held by some other process. For eg :- P1 can hold Resource R1 and at the same time, it can request some other Resource R2 currently hold by P2



3) No Pre-emption :- A resource can't be preempted from the process by another process, forcefully. For example if a process P1 is using some resource R, then some other process P2 can't forcefully take that resource. If it is so then what's the need of various CPU scheduling algorithms. The process P2 can request for the resource R and can wait for that resource to be free by the process P1.

4) Circular Wait :- A situation can arise in which process P1 holds resource R1 while it request resource R2 and P2 holds R2 while it request R1. This is called circular wait.



Deadlock will happen if all the above conditions happen simultaneously.

Date : / /
Page No.

System Model :-

A process must request a resource before using it and must release the resource after using it. The no. of ~~process~~ resources requested may not exceed total no. of resources available in the system.

A process may utilize a resource in following sequence

1) Request :- The process request the resource. If the request cannot be granted immediately, then requesting process must wait until it can acquire the resource.

2) Use :- The process can operate the resource.

3) Release :- The process releases the resource.

Resource Allocation Graph (RAG) :-

1) RAG is a directed graph that can be used to illustrate the state of a system graphically. A directed graph that can be used to describe deadlock more precisely.

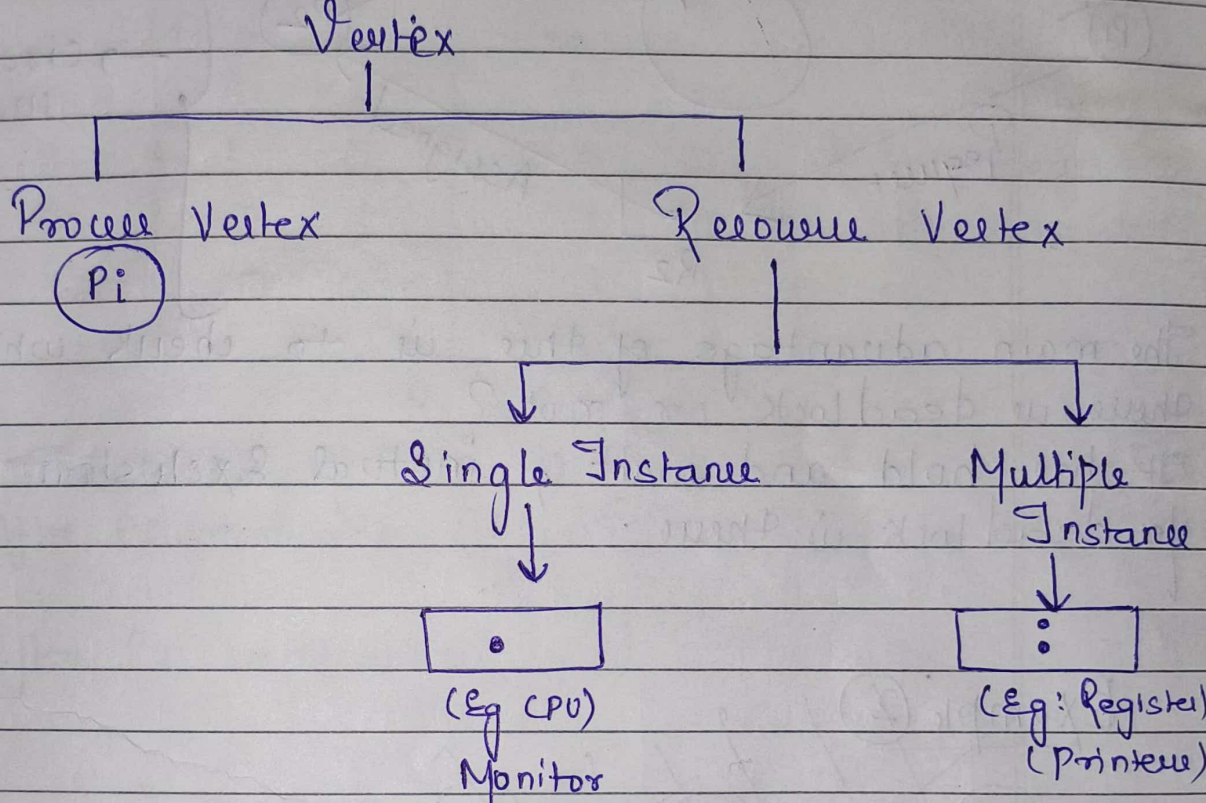
2) It is the most convienent and efficient way to represent the state of the system.

3) How the resources are allocated to the processes can be graphically observed with the help of RAG.

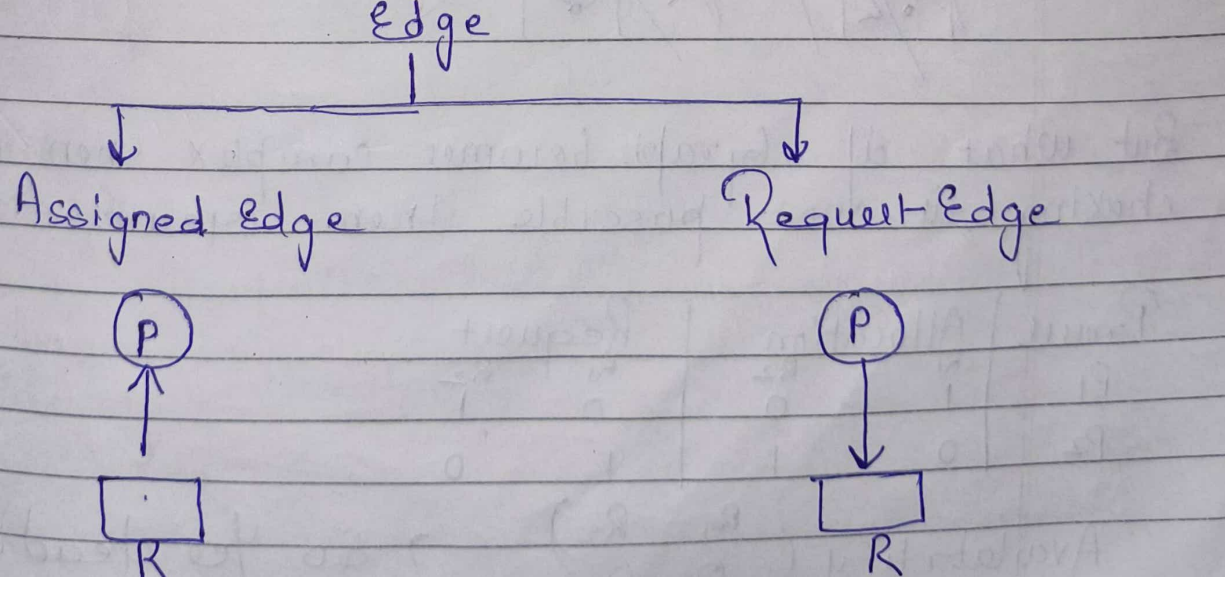
Date : / /
Page No.

4) Like a Normal Graph it is having two things
   a) Vertex
   b) Edges

5) Vertices are of two types

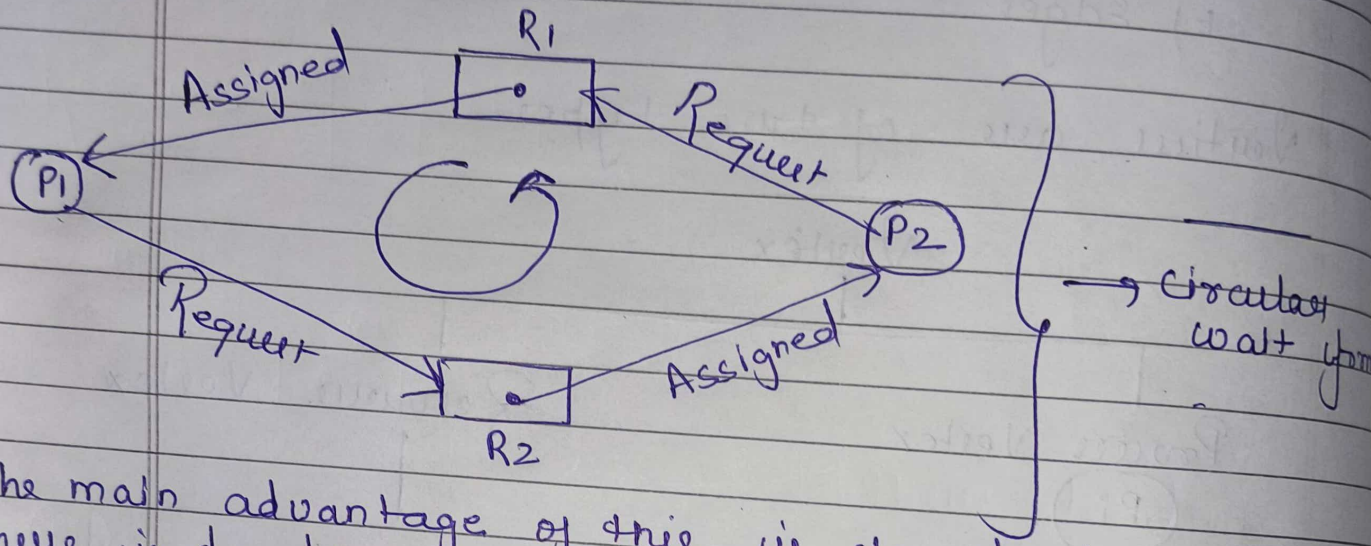


6) Edges are also of two types which represen



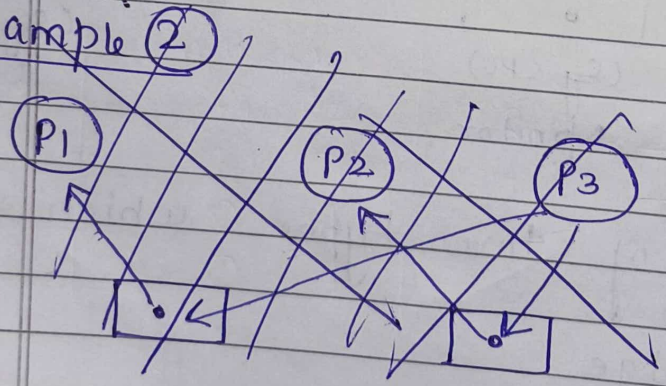
Now let's Solve one Example :-

Resource of Single Instance



The main advantage of this is to check whether there is deadlock or not?
It has hold and wait, mutual Exclusion so Yes deadlock is there.

Example (2)



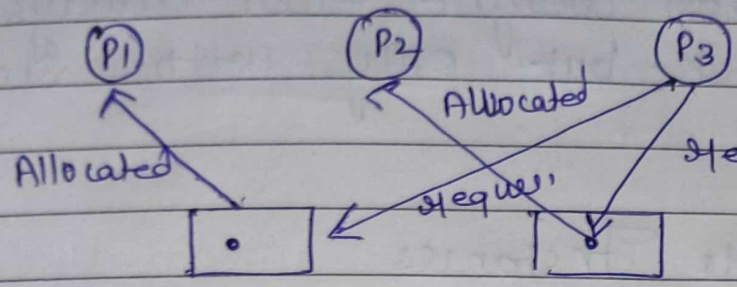
But what if Graph becomes complex then manually checking is not possible then create Matrix

Process	Allocation		Request	
	R1	R2	R1	R2
P1	1	0	0	1
P2	0	1	1	0

Availability (R1 R2) → 0 0 → So Yes deadlock.

Date : / /
Page No.

Example (2)



Process	Allocate R1	Allocate R2	Request R1	Request R2	
P1	1	0	0	0	→ No request
P2	0	1	0	0	→ No request
P3	0	0	1	1	

terminated

Availability (R1 R2) = (0 0)

After P1 termination (1 , 0)

After P2 termination (1 , 1)

Current Availability (1,1) → 2 Resources
1 process

No deadlock only starvation

If waiting

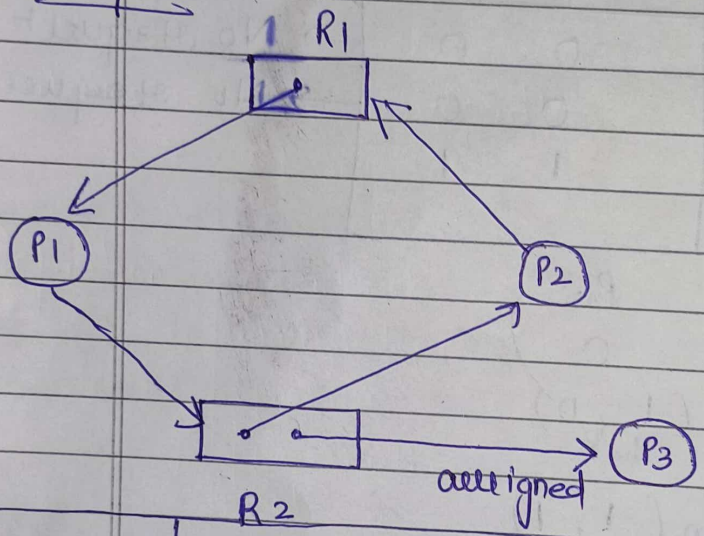
finite (Starvation)

infinite (Deadlock)

From all previous example we conclude that if RAG has circular weight then always there will be a deadlock. but only with single Instance resource.

Resource Of Multiple Instances

Example :-



Is there deadlock ?

R2

Process	Allocate R1	Allocate R2	Request R1	Request R2
X P1	1	0	0	1
X P2	0	1	1	0
X P3	0	1	0	0

Availability : { R1 R2
(0 0)
+ 0 1 → After P3 terminate
Availability 0 1
+ 0 0 Give R2 to P1
1 1 → After P1

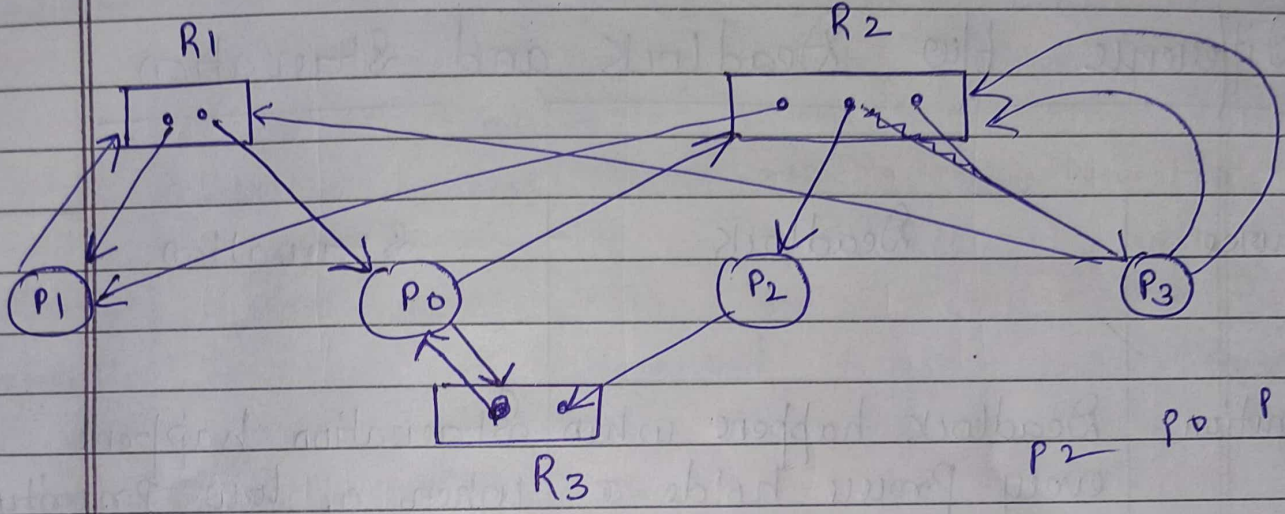
No deadlock (There is cycle then there is

Date : / /
Page No.

deadlock having Single Instance) but with cycle having multiple instance there is chance of deadlock but not always.

Gate Exam: 2018

Example:-



P2 P0 P1

Process	Allocation			Request		
	R1	R2	R3	R1	R2	R3
X P0	1	0	1	0	1	1
X P1	1	1	0	1	0	0
X P2	0	1	0	0	0	1
X P3	0	1	0	1	2	0

	R1	R2	R3	
Current Availability	0	0	1	request fulfilled
	0	1	0	
Current Availability	0	1	1	→ P0 request
	1	0	1	
Current Availability	1	1	2	→ P1 request
	1	0	0	
	2	1	2	→ P3 request

⇒ Current available (2 3 2) Ans.

Date : / /
Page No.

2 1 3
0 1 0
2 2 3 Ans

Current Availability

No deadlock

Difference b/w Deadlock and Starvation

Features	Deadlock	Starvation
Definition	Deadlock happens when every process holds a resource and waits for another process to hold another resource	Starvation happens when a low priority program request a system resource but cannot run because a higher priority program has been employing that resource for a long time.
Basic	A deadlock occurs when no process can proceed and becomes blocked.	Starvation occurs when low priority procedures are blocked while high priority operations proceed.
Other name	Deadlock is also known as circular wait	Known as a lived lock.

Date : / /
Page No.

Resource	Other process block requested resource while a process is deadlocked.	High Priority process continue to use the requested resources.
Arising Condition	Mutual Exclusion, Hold and wait, No pre-emption and circular wait all happen simultaneously.	Uncontrolled resource management enforcement of priorities.
Prevention	It can be prevented by avoiding the situations that lead to deadlock.	Aging may prevent it

Methods to Handle Deadlock

① Deadlock Ignorance :- (Ostrich Method)

We are ignoring bcoz we don't want to affect the performance or functionality inc.

It simply says ignore the deadlock. It is widely used technique. The user avoid to write patches to handle deadlock. Because writing of these patches of program will surely affect the functionality performance of os. It is also assumed that deadlock occurs rarely.

Date : / /
Page No.

We get this idea from Ostrich as when Sand Storm come ostrich simply stick one's head in the sand and pretend there is no problem

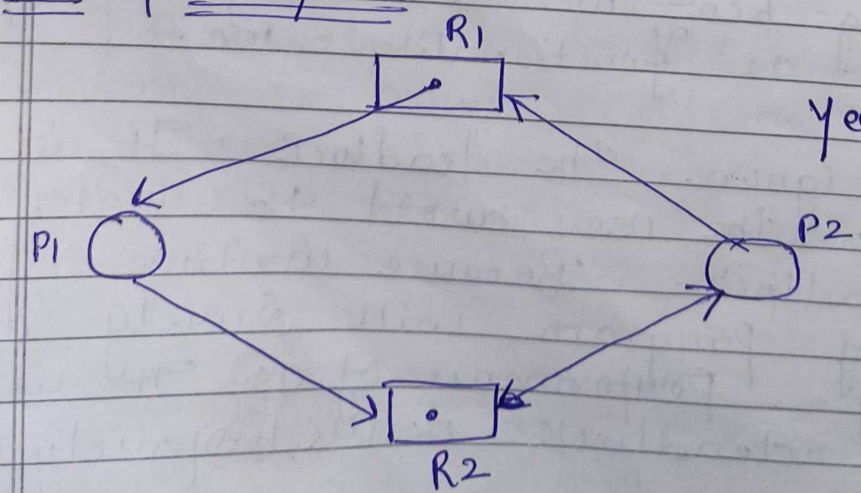
(2) Deadlock Prevention :-

"Prevention is better than Cure"

Deadlock prevention is a set of methods for ensuring that atleast one of the necessary condition can't hold. These methods prevent deadlocks by constraining how request for resources can be made.

a) Mutual Exclusion :- This condition must hold for non-shareable resources like printer. Some resources, such as files may allow multiple access for reads but only exclusive access for write. Not every resource in System can be used in shareable fashion to take the condition NO Eg: printer, tape drives. If all the resources in a System can be made shareable, then deadlock can be prevented.

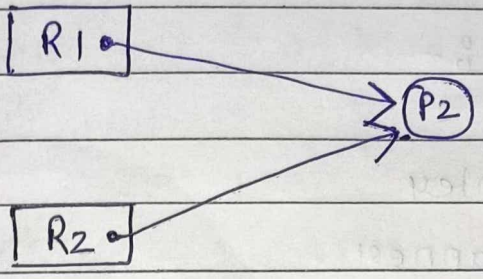
b) Non Preemption



Yes it has deadlock

Date : / /
Page No.

if Preemption is allowed, then we have preempted P1 Process. (running to ready) & resource R1 will be free now.



If a Process request a resource that is currently held by another process, OS may preempt the second process and require it to release its resource.

Preemption is possible for certain types of resources such as CPU and main memory. Preemption is done on basis of time stamp or time quantum.

So we are trying to prevent deadlock by making no Preemption to preemption.

No

c) Hold and Wait :-

Here rather than a process is holding some resources and waiting for others that are currently not available. All resources needed to it should be allocated to it in beginning before its execution.

For eg. Process copies data from a floppy disk to a hard disk, sort a disk file and then prints results to a printer. If all the resources are requested at beginning

Then process will hold printer for its entire execution, even though it need it only at end. Resource utilization is poor here.

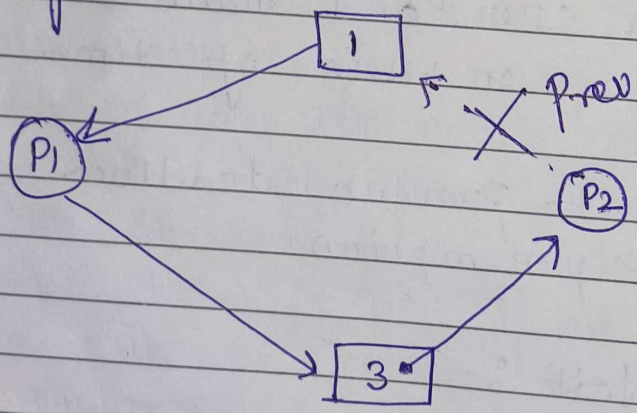
No

d) Circular Wait :-

1 Printer
2 Scanner
3 CPU
4 Register

If P1 need printer (1) it can request ~~in~~ for 2, 3, 4 (in increasing order)

Eg:



Now if (P2) come and request for 3 i.e. CPU He can give but if it now request for (1) & (2) then this request can't be fulfilled.

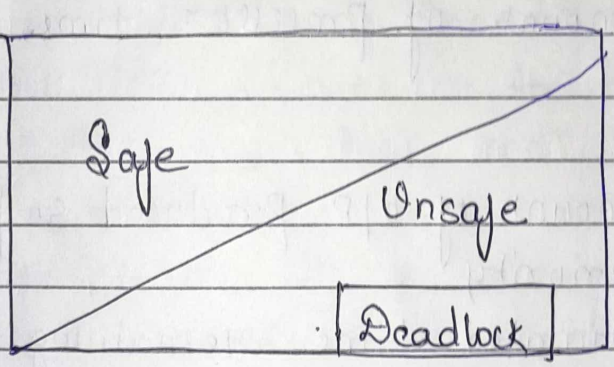
One way to prevent circular wait condition is by linear ordering of diff types of System resources

Date : / /
Page No.

(3) Deadlock Avoidance :- (Banker's Algorithm)

Deadlock avoidance therefore allows more concurrency than prevention does. Deadlock avoidance requires additional information about how resources are to be requested.

With deadlock avoidance, a decision is made dynamically whether current resource allocation request could be granted, doesn't lead to a deadlock.



A safe state is a state in which there is atleast one order in which all the process can be run for completion without resulting in deadlock.

(4) Deadlock detection & Recovery :-

If the system is not using any deadlock avoidance and deadlock prevention then a deadlock situation may occur. Deadlock detection approach do not limit resource access or restrict process actions.

deadlock recovery technique

Date : / /
Page No.

Once deadlock has been detected, some strategy is needed for recovery. Two solutions are there :-

a) Process Termination :-

Abort one process at a time until the deadlock cycle is terminated. The order in which process are selected for abortion should be on basis of some factors.

→ Least amount of processor time consumed so far
→ least amount of o/p produced so far.
→ Lowest priority
→ Most estimated time remaining
→ Least total resources allocated so far.

b) Resource Preemption :-

If preemption is required to deadlock then three issues need to be addressed.

→ Selecting a victim :- Which resource and which process are to be preempted.

→ Rollback :- Backup each deadlocked process to some previously defined checkpoint and restart all process.

→ Starvation :- How do we ensure that

Date : / /
Page No.

Starvation will not occur.

Deadlock Avoidance Technique : Banker's Algo

1) Bankers algorithm is a deadlock avoidance algorithm. It is named so because this algorithm is used in banking systems to determine whether a loan can be granted or not.

2) Suppose there are 'n' account holders in a bank and the total sum of their money is 's'. if a person applies for a loan then bank first substracts the loan amount from the total money that the bank has and if the remaining amount is greater than s then only bank approved the loan. It is done because if all the account holders come to withdraw their money bank can operate all the things without any restriction in the functionality of banking system.

3) Similarly it works in operating system. When a new process is created in a computer system, the process must provide all types of information to the OS like upcoming processes, request for their resources, count, etc. Based on this info OS decides which process sequence should be executed or waited so that no deadlock occurs in a system.

4) Therefore it is also known as deadlock avoidance and deadlock detection algorithm.

Date : / /
Page No.

5) Several data structures must be maintained to implement the banker's algorithm.

6) Let 'n' be the no. of processes in system and 'm' be the no. of resource types.

↳ Available[j] = K denotes that there are K instances of resource type Rj available. A Vector of length 'm' indicates no. of available resources of each type.

↳ max[i,j] = K, then process Pi may request atmost K instances of resource type Rj. An n x m matrix defines max demand of each process.

↳ Allocation[i,j] = K, then process i is currently allocated K instances of resource type Rj

A n x m matrix defines no. of resources of each type currently allocated to each process.

↳ Need[i,j] = K, then Process Pi need K more instances of resource type Rj to complete the task.

An n x m matrix indicate remaining resource need of each process.

The Bankers algorithm is the combination of the Safety Algorithm and the resource request

Date : / /
Page No.

algorithm to control the processes and avoid deadlock in a System.

Safety Algorithm :→

Safety Algorithm is used to find State of the System. That is System may be in Safe state or Unsafe state.

Safety Algorithm

1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4....n
2) Find an i such that both
a) Finish[i] = false
b) $Need_i$ <= Work
if no such i exists goto step (4)
3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)
4) if Finish [i] = true for all i
then the system is in a safe state

Resource Request Algorithm

Date : / /
Page No.

1

Resource-Request Algorithm

1) If Request$_i$ <= Need$_i$
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2) If Request$_i$ <= Available
Goto step (3); otherwise, P$_i$ must wait, since the resources are not available.
3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows:
Available = Available – Requesti
Allocation$_i$ = Allocation$_i$ + Request$_i$
Need$_i$ = Need$_i$– Request$_i$

Ques Consider a System that contains 5 processes P1, P2, P3, P4 and P5 and three Resource types A, B and C. Following are the resource types: A has 10 instances, B has 5 and C has 7 Instances. (AKTU Ques)

CPU Memory Printer

Process	Allocation (A B C)	Max Need (A B C)	Current Available (A B C)
P1	0 1 0	7 5 3	3 3 2
P2	2 0 0	3 2 2	
P3	3 0 2	9 0 2	
P4	2 1 1	4 2 2	
P5	0 0 2	5 3 3	
	7 2 5		

Answer the following questions? Total – Allocated = Available

1) What is the reference of the need matrix
2) Determine if the system is safe or not
3) What will happen if the resource request

Date : / /
Page No.

(1, 0, 2) for process P1, can the system accept this request immediately?

Soln

A = 10, B = 5, C = 7

Max - Allocated

Process	Allocation A	B	C	Max Need A	B	C	Current Available A	B	C	Remaining Need A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2	5	3	2	1	2	2
P3	3	0	2	9	0	2	7	4	3	6	0	0
P4	2	1	1	4	2	2	7	4	5	2	1	1
P5	0	0	2	5	3	3	7	5	5	5	3	1
							10	5	7			

(Curr avail) = 10 5 7

Safe Sequence :- (P2) P4 P5 P1 P3

b) Yes System is Safe

c) Part :- For granting the request we have to check Request <= Available that is (1,0,2) <= (3,3,2) since the condition is true so request is grant immediately

Ques (2) Gate2020

Process	Allocation E	F	G	Max Need E	F	G	Available E	F	G
P0	1	0	1	4	3	1	3	3	0
P1	1	1	2	2	1	4			
P2	1	0	3	1	3	3			
P3	2	0	0	5	4	1			

m → j

n

i

Soln

	Allocation			Max Need			Max Available			Max Remaining Need		
Process	E	F	G	E	F	G	E	F	G	E	F	G
P0	1	0	1	4	3	1	3	3	0	3	3	0
P1	1	1	2	2	1	4	4	3	1	1	0	2
P2	1	0	3	1	3	3	5	3	4	0	3	0
P3	2	0	0	5	4	1	6	4	6	3	4	1
							8	4	6			

Safe Sequence :- (P0 → P2 → P1 → P3)

Yes System is Safe and there is no deadlock.

Ques (3) A System is having 3 process each require 2 units of resource R. The minimum no. of units of R such that no deadlock will occur ____ ?

Gate Ques

a) 3 b) 5 c) 6 d) 4

Ans If we have 3 process

P1 P2 P3

~~then~~ all process require 2 instance of resource

So if we have 6 resource then no deadlock will occur.

P1 P2 P3

2 2 2 = 6

but ques is asking minimum.

So let's try for

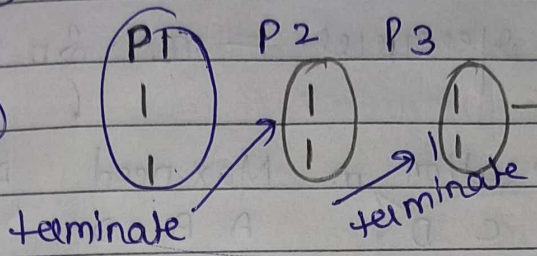
Date : / /
Page No.

R = 2

P1 P2 P3

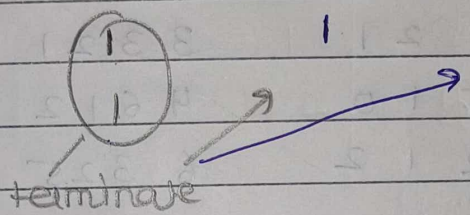
Case ① 1 1 → deadlock

Case ② P1 P2 P3 → terminate



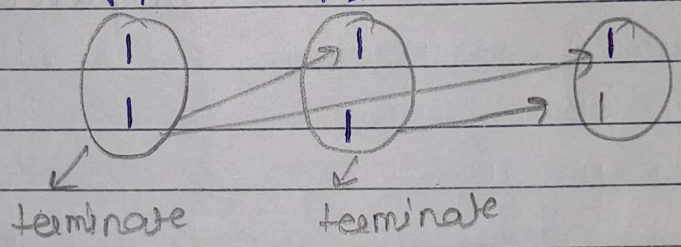
Case ③ P1 P2 P3

1 1 → Again deadlock



R = 3

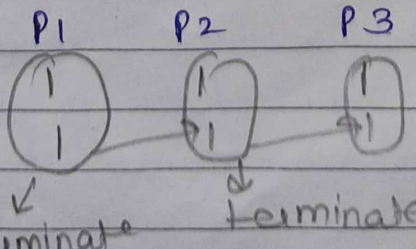
Case ① P1 P2 P3 → terminate



Case ② P1 P2 P3

1 1 1 → Again deadlock

R = 4



So yes with R=4 No deadlock will occur

Date : / /
Page No.

So min no. of resources to avoid deadlock
$R = \boxed{n+1}$

max no. of resources = n So that deadlock will occur.

Ques

Process	Allocation (A B C D)	Max need (A B C D)	Available (A B C D)	Remaining (A B C D)
P0	3 0 1 4	5 1 1 7	0 3 0 1	2 1 0 3
P1	2 2 1 0	3 2 1 1	+ 3 1 2 1 = 3 4 2 2	1 0 0 1
P2	3 1 2 1	3 3 2 1	+ 2 2 1 0 = 5 6 3 2	0 2 0 0
P3	0 5 1 0	4 6 1 2	+ 0 5 1 0 = 5 11 4 2	4 1 0 2
P4	4 2 1 2	6 3 2 5		2 1 1 3

n

Safe Sequence: P2 → P1 → P3

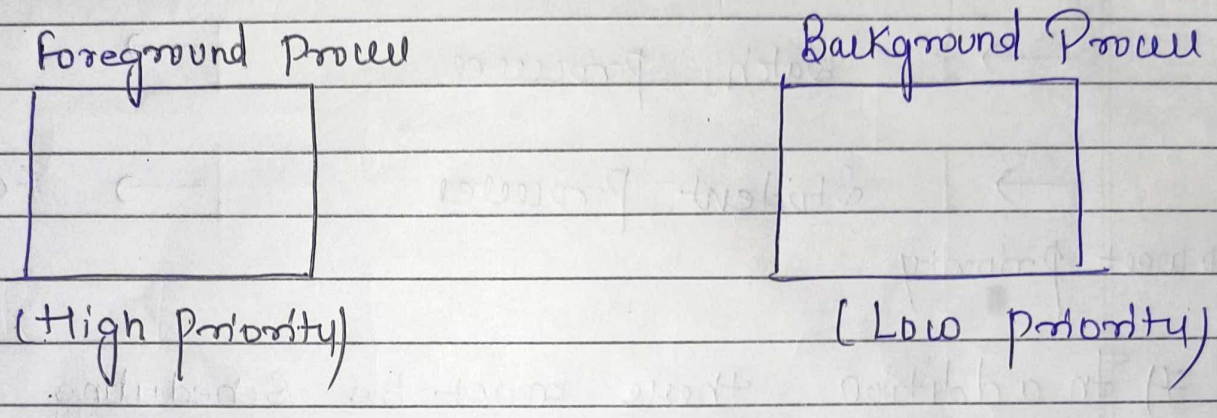
Both P0 & P4 can't be completed.

Date : / /
Page No.

Multilevel Queue Scheduling in OS

1) It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs.

2) For Example, a common division is a foreground (Interactive) process and a background (batch) process.



3) These two types of processes have different response time requirements and so might have different scheduling needs. In addition foreground process may have priority over background process.

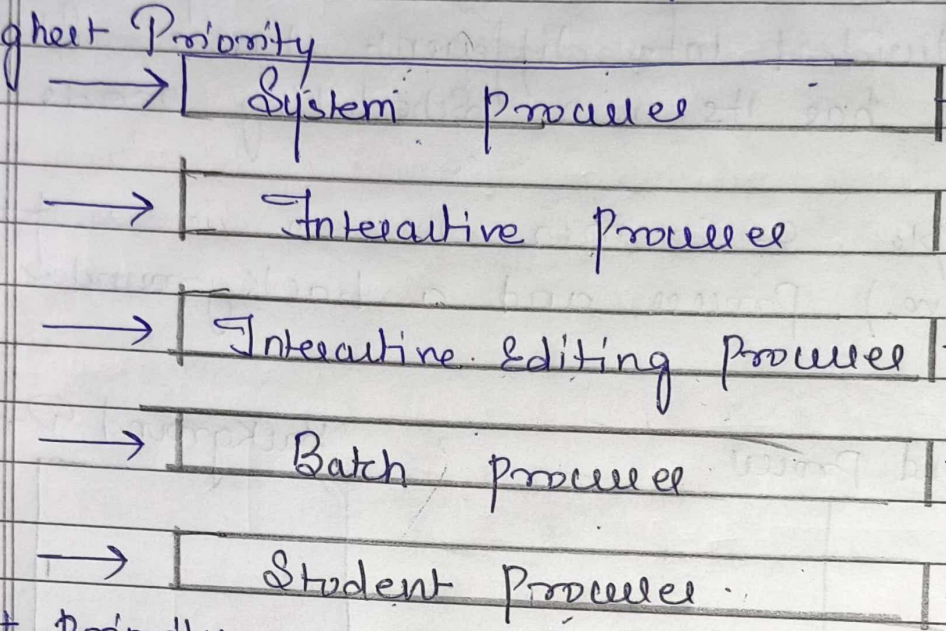
4) A multilevel queue scheduling algorithm partitions the ready queue into several seperate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority or process type.

5) Each queue has its own scheduling algorithm

6) For example Foreground queue might be scheduled

Date : / /
Page No.

by round robin algorithm, while the background queue is scheduled by an FCFS algorithm.



7) In addition there must be scheduling among queues, which is commonly implemented as fixed priority pre-emptive scheduling.

Multilevel feedback queue Scheduling Algorithm

1) The multilevel feedback queue scheduling algorithm (MFQ) is an extension of (MLQ) which allows a process to move between queues.

2) The idea is to seperate processes according to the characteristic of their CPU bursts.

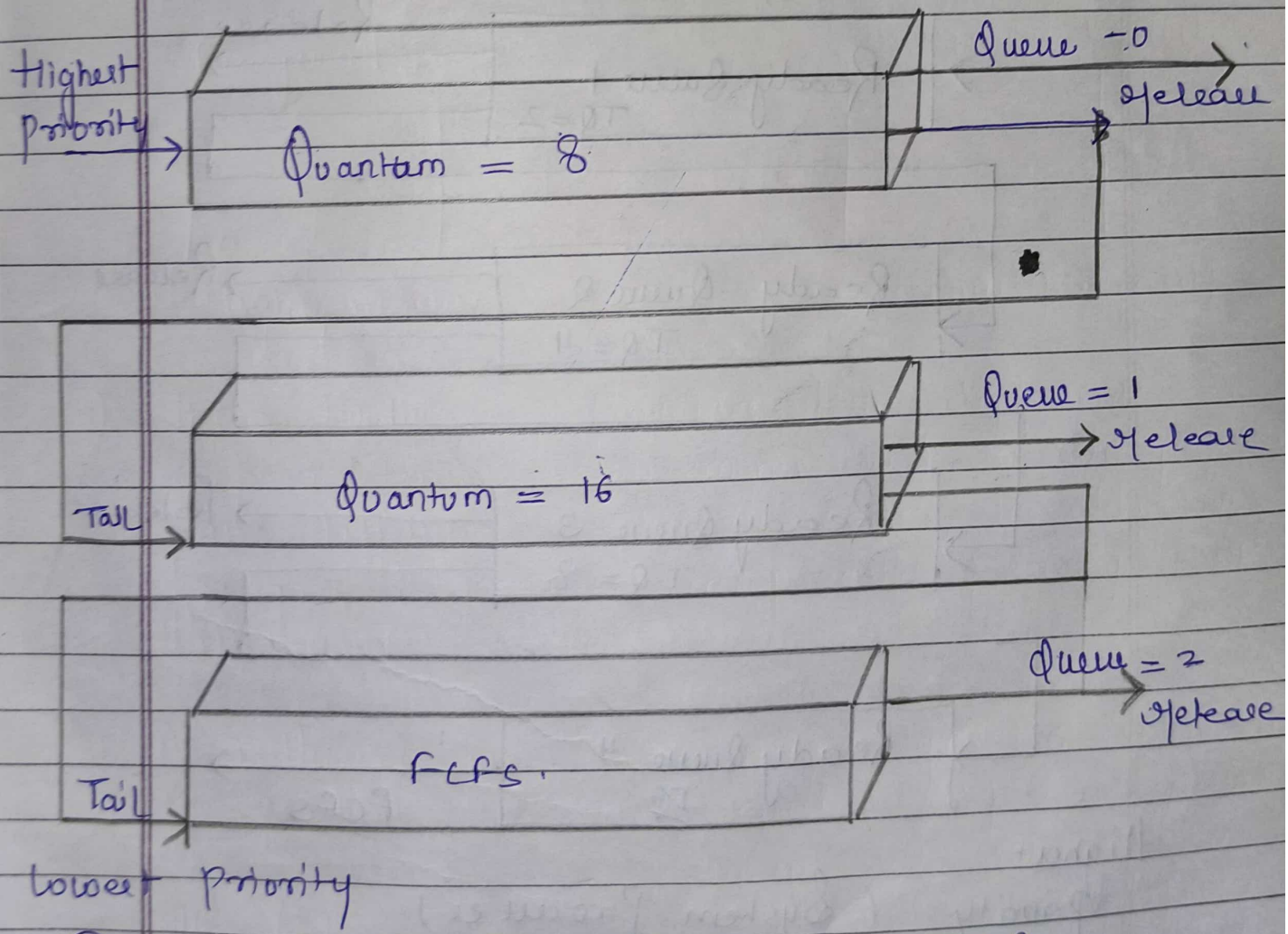
3) If a process uses too much CPU time, it will be moved to a lower priority queue.

Date : / /
Page No.

4) This Scheme leaves I/O bound and interactive processes in the higher priority queues.

5) In addition, a process that waits too long in a lower priority queue may be moved to a higher priority queue.

6) This form of aging prevents starvation.



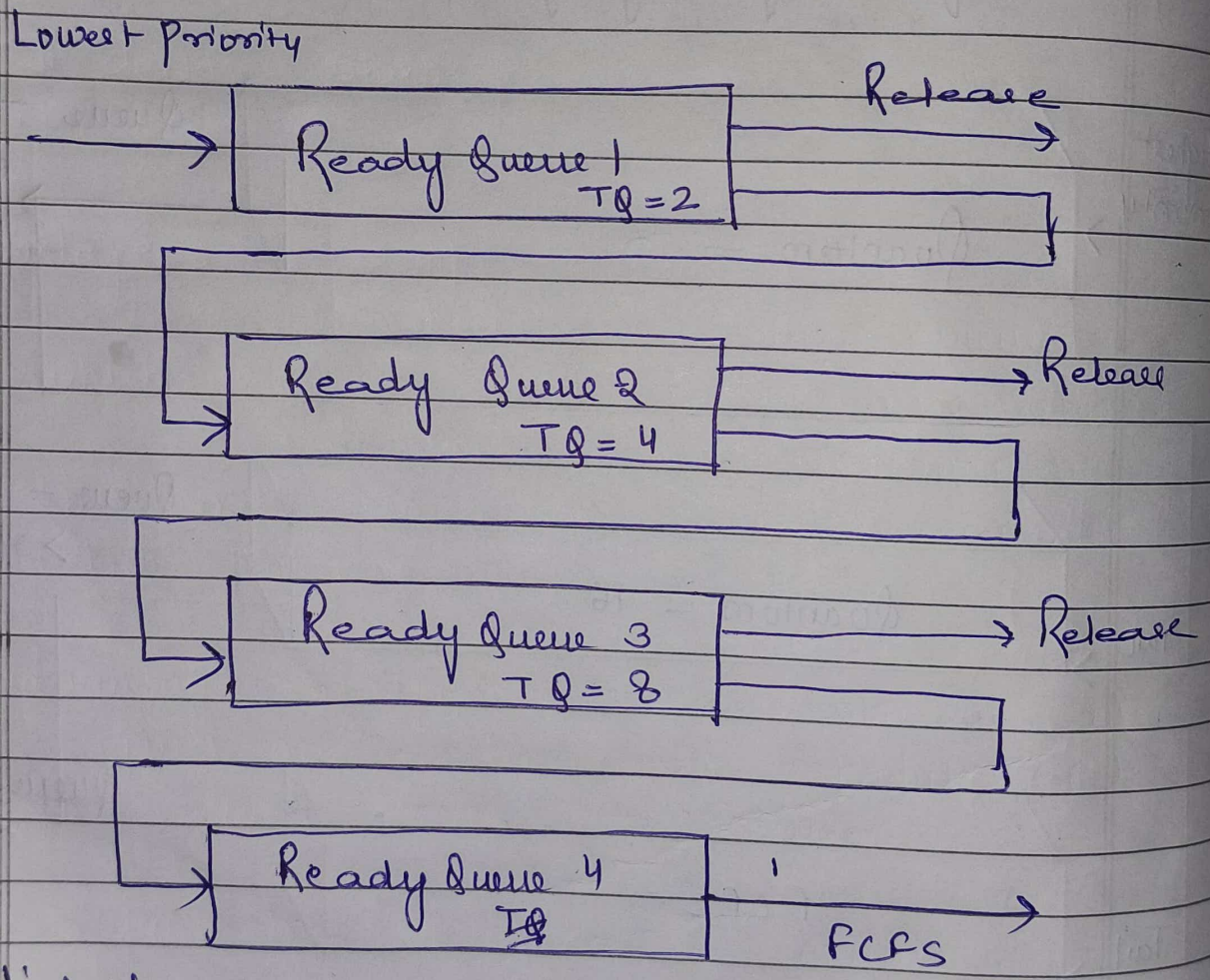
Parameters by which MFQ is defined :-

1) The number of queues
2) The scheduling algorithm for each queue
3) The method used to determine when to

upgrade a process to a higher priority queue.

4) The method used to determine when to demote a process to a lower priority queue

5) The method used to determine which queue a process will enter when that process needs service.



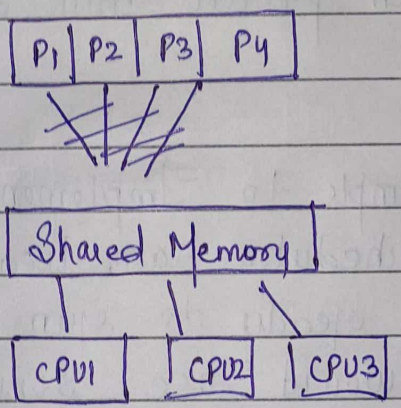
Date : / /
Page No.

Multiple Processor Scheduling Algorithm

↳ In a multiple processor system there are multiple CPU's

↳ The processing load is distributed among there available processes.

↳ But in multiprocessor system, scheduling multiple processors is quite complex.

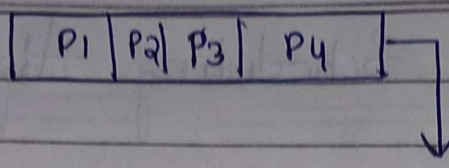


↳ We have multiple processors in the system so we have to decide how scheduling algorithm allocate the processes to multiple CPUs

So MultiProcessor Scheduling can be Implemented using 2 approaches:-

① Asymmetric Multi Processor Scheduling:-

In asymmetric approach one processor is made the master processor that handles all the scheduling policies, I/O operation, resource allocation and other system actions. Other processors works like a slave that are responsible to execute only user code.



(master CPU)

CPU3 CPU2 CPU2 CPU1 (This CPU has local Scheduler)

↳ Here there is a dedicated CPU to run the Scheduler

↳ This Scheduler decides for everyone.

↳ It decides which process will execute on which CPU

Advantage :- • Simple to Implement
• Scheduler have local queue in which processes ready to run are stored.
• It would use some mechanism to decide which process will execute on which processor.

Disadvantage :- • Performance degradation as all CPU(s) are waiting for Scheduler CPU for which process to execute.

(2) Symmetric Multiprocessor Scheduling

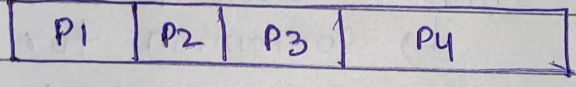
↳ In Symmetric multiprocessing a single copy of OS is maintained which is used by all processor

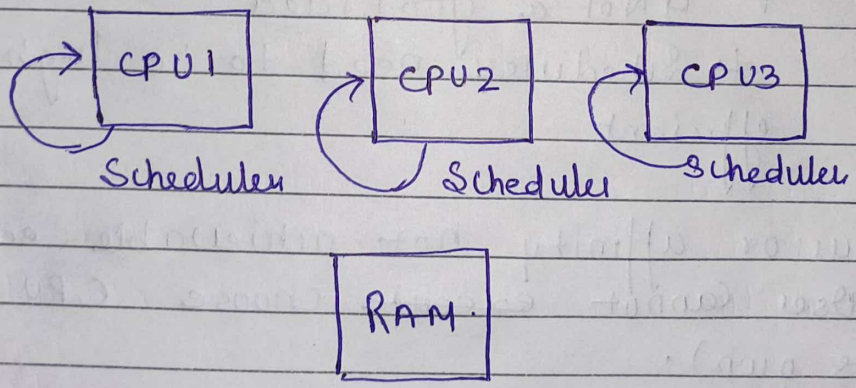
↳ All processors share I/O bus and memory

↳ Here the process waiting for CPU to get executed are maintained in global queue.

Date : / /
Page No.

- Here each process can do the process scheduling on its own and select a process to execute from the global ready queue.
- In order to have efficient use of processors a local ready queue associated with each processor is also maintained.
- Each processor runs a scheduler independently to select the process to execute.

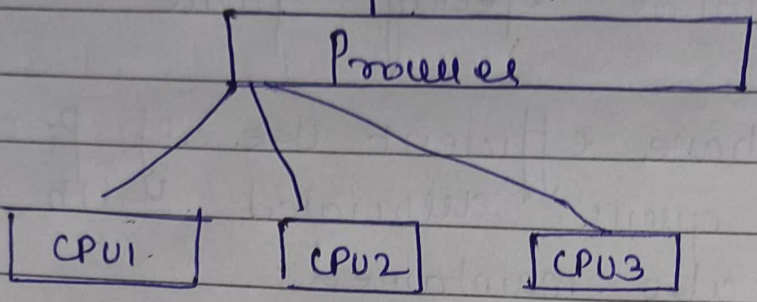




- There are three methods to implement this
  - Using Global queue
  - Using ~~local~~ Seperate queue for each CPU
  - Hybrid approach

Date : / /
Page No.

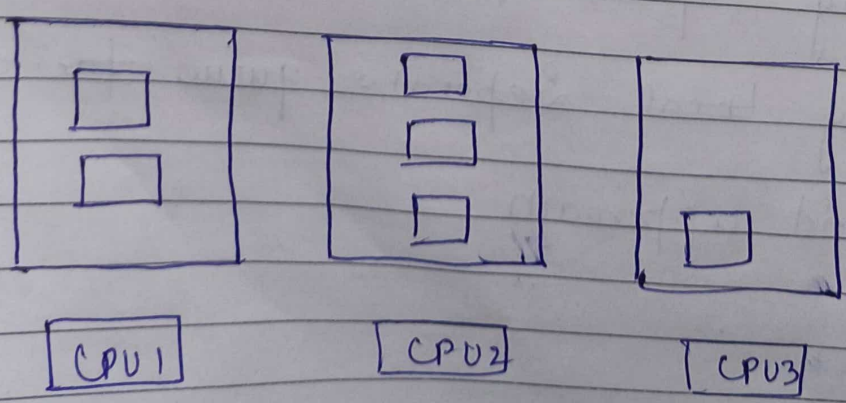
a) Symmetric Scheduling with Global queue :-



Adv :- ↳ Good CPU utilization
↳ Fair to all process.

Disadvantages :- 1) Not Scalable
↳ contention for Global queue
2) Locking mechanism is needed in Scheduler
↳ Not a good idea
↳ Schedulers need to be highly efficient
3) Processor affinity not achievable easily (User cannot ~~execute~~ choose CPU on its own).

b) ~~Locking~~ Symmetric Scheduling with Seperate queue for each CPU.



Date : / /
Page No.

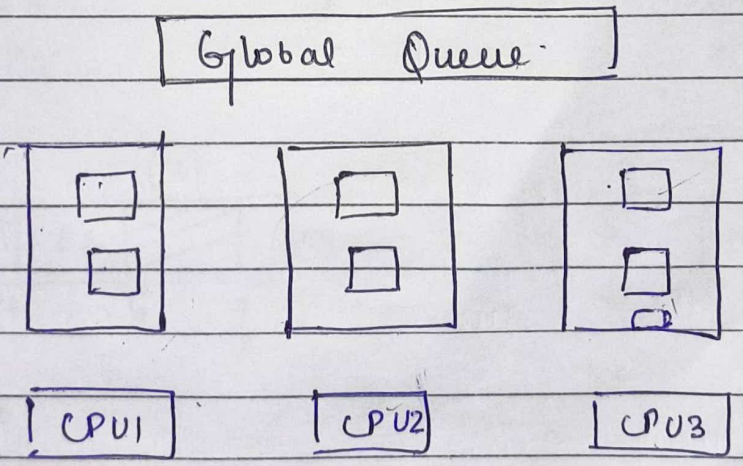
b) Each CPU execute process from its own local ready queue of process.

b) It uses static partition of process across CPU. At the start of the execution of process either the user / OS decides in which CPU process will execute then the process will be placed on corresponding CPU(s) queue.

Advantage :- • Easy to Implement
• No locking mechanism needed
• Scalable.

Disadvantage :- • If some CPU(s) have lot of process in ready queue and some have less process in ready queue

c) Hybrid approach :-



• It uses both local and global queues
• local queues are associated with each CPU
• Global queue is shared among CPUs
• Global queue is used to maintain

Date :
Page No.

Load balancing

- Used in Linux 2.6

Adv :- Load balancing across queues is feasible.

Gate Ques

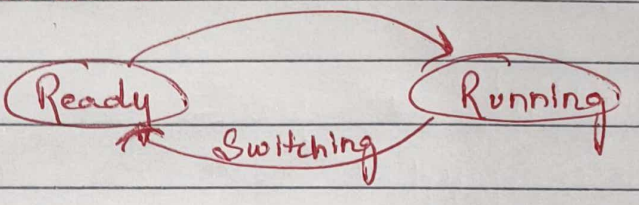
Date : / /
Page No.

Imp Ques

Process	Burst time	AT
P1	8	0
P2	2	0
P3	7	0
P4	3	0
P5	5	0

Solve using Round Robin Scheduling.

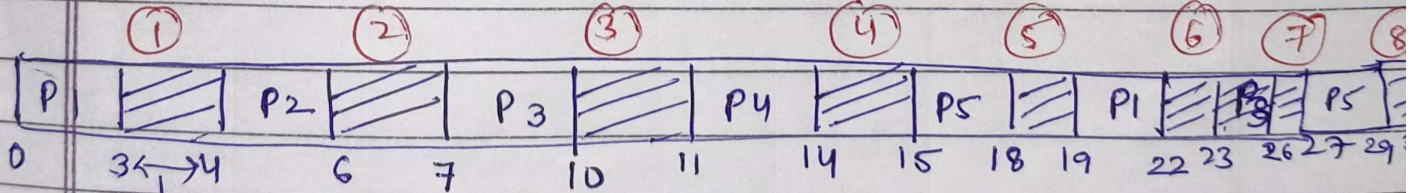
Context Switch :→



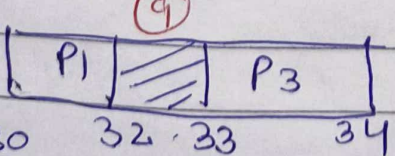
Let Context Switch time is 1unit and TQ = 3
Then calculate its WT, RT, CPU utilization%

Sol^n

Process	BT	AT	CT	TAT	WT	RT
P1	8	0	32	32	24	0
P2	2	0	6	6	4	4
P3	7	0	34	34	27	7
P4	3	0	14	14	11	11
P5	5	0	29	29	24	15



Context Switch



Total No of Context Switches = 9

Date : / /
Page No.

CPU utilization = $\frac{\text{Expected time}}{\text{Actual time}}$

= Expected time is equal to total burst time

= $\frac{\text{Total Burst time} \times 100}{34}$

= $\frac{25}{34} \times 100 = 73.52\%$ Ans