Date 10/06.

Operating System

1) Intro & Background

2) Process Management.
   - Process concept.
   - cpu scheduling.
   - I.P.C. & synchronization (Imp) (I)
     (Interprocess comm.)
   - concurrency
   - dead locks.
   - Threads.

3) Memory Management:
   - RAM chips.
   - Loaders vs Linkers
   - Techniques
     - Partitions
     - Paging
     - segmentation
   - Virtual memory. (Imp) (II)

4) File system & Device mgmt.
   - Interface
   - Disk characteristics.
   - Impl. Issues. (Imp) (III)

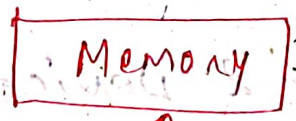
5) Protection.

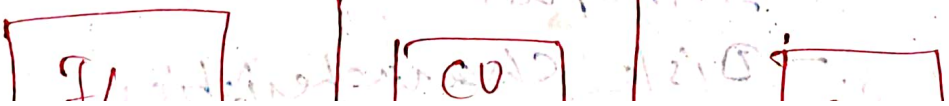
Sys. S/w
- Resource Manager.
Set of utilities to simplify appl. development.
Ctrl program(s)
Like a Govt (process Mgmt, memo. mgm
= CA vs C.O.

CA → This is giving a outline about the comp. like the block d/a of the H/w.

CO → In which way we manage the comp. H/w

→ Block d/a of digital comp-





2) M-O
(lower level
out the ins
on the regis
out during si

eg of M-operat
C = a+b;
Load R1, M
Load R2, M
ADD R1, R2
Store M[C]

ALU → Addn, ..
operands
relational ope

Primary M.

func of ctr Unit →

1) Timing & ctrl sys.(

2) M-operations. (which is the lower level operatn carried out the inside the processor on the register, data is carried out during single cycle.

eg of M-operatns

```
C = a+b;
Load R1, M[a]
Load R2, M[b]
ADD R1, R2
Store M[C], R1
```

ALU → - Addn, mult, sub. etc. operatns of logical & relational operatns.

Primary Memory → It is volatile

- RAM, ROM. Cache, Registers.

RAM, cache, Reg. are volatile.
But - ROM is not volatile.

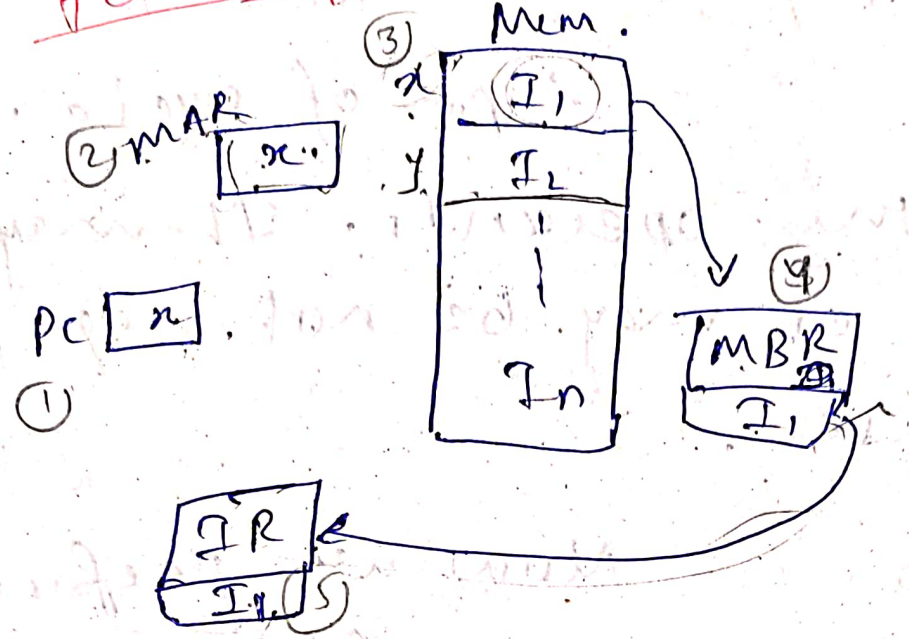
→ The mem. is presented in the block dia of the comp. is a primary mem. The seco. mem is presented as the I/O devices

Bootstrap → It is a loader & it loads the O.S. from the Sec. mem. to the primary mem. In the initially the Primary mem is empty & the O.S. is in the sec. mem & we can't directly access any things from the sec. mem. It containing the BIOS. It is k/a BIOS ROM or ROM BIOS,

BIOS → Basic Input/output system. This is performed by the ROM & these r having like PROM, EPROM, EEPROM.

features	Primary	Secondary
Access time	high	less
volatile	volatile	Non volatile
larger	less	high.
size	high	less
Ex	RAM, ROM, Cache, Reg.	CDROM, Floppy,

Fetch Cycle →

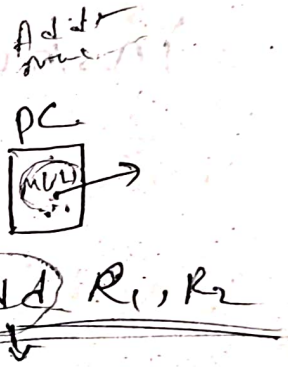


PC → Pgm counter.

MAR → Mem. Addr Reg.

IR → Instruction Reg.

MBR → mem. buffer Reg.



Even the instruction is ~~code~~ loaded in the mem. we don't know about the type of the cycle. So, the various type of cycles are there -

- Relational cycle
-

Decode cycle - Here we find out the type of the cycle.

Operand fetch :- Acco. to the type of cycle we fetch the operands. It may be req or may be not req : for some cases.

Execution — Now we perform the execution of the instruct^n.

These all the above 4 cycles combinely make a cycle k/a Instruction cycle or comp. cycle.

Interrupt cycle → If the processor is capable of supporting interrupt mechanism then it is said to be interrupt cycle.

If process executing instruct^n if the device gets interrupt. it can interrupt only after execution cycle.

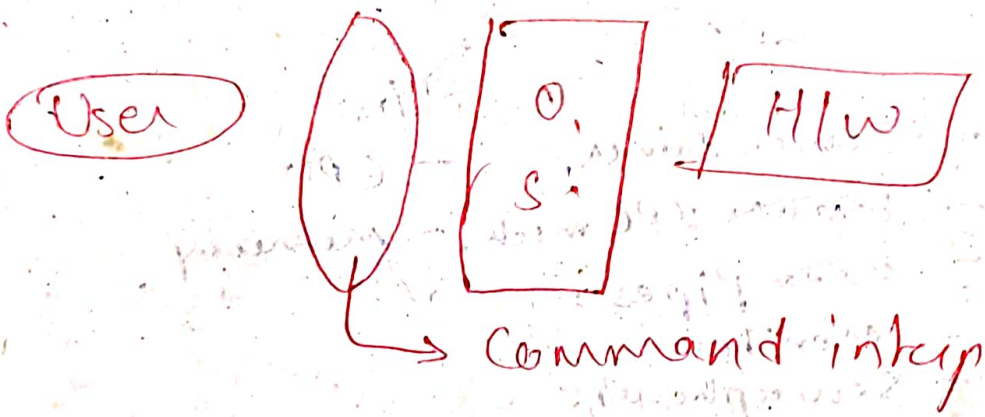
So after every execution cycle it polls whether there are only interrupt cycles.

1) Stored Prog. concept —

Only when the instruction is executed when the instruction is stored in the primary mem. otherwise we can't execute the instruction.

2) Sequential flow of ctr.

Only one instru. is executed at a time.



Os. → It is only the convienient environment ~~b/w the H/w~~ for & user. It is interface betn the user & H/w.

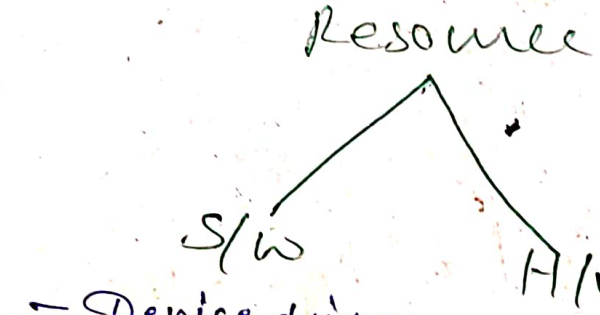
Command Interpreter — It is the interface b/w the O.S. & User.

→ Shell → In the DOS & UNIX. (Text Based)

→ GUI. (Desktop).

Resource Manager →

Resources → It is classified in two way S/W & H/W.



- Device driver
- ~~Loader~~ IPC mech.
- ~~Loader~~ Pipes.
- Monitors.
- Semaphores.

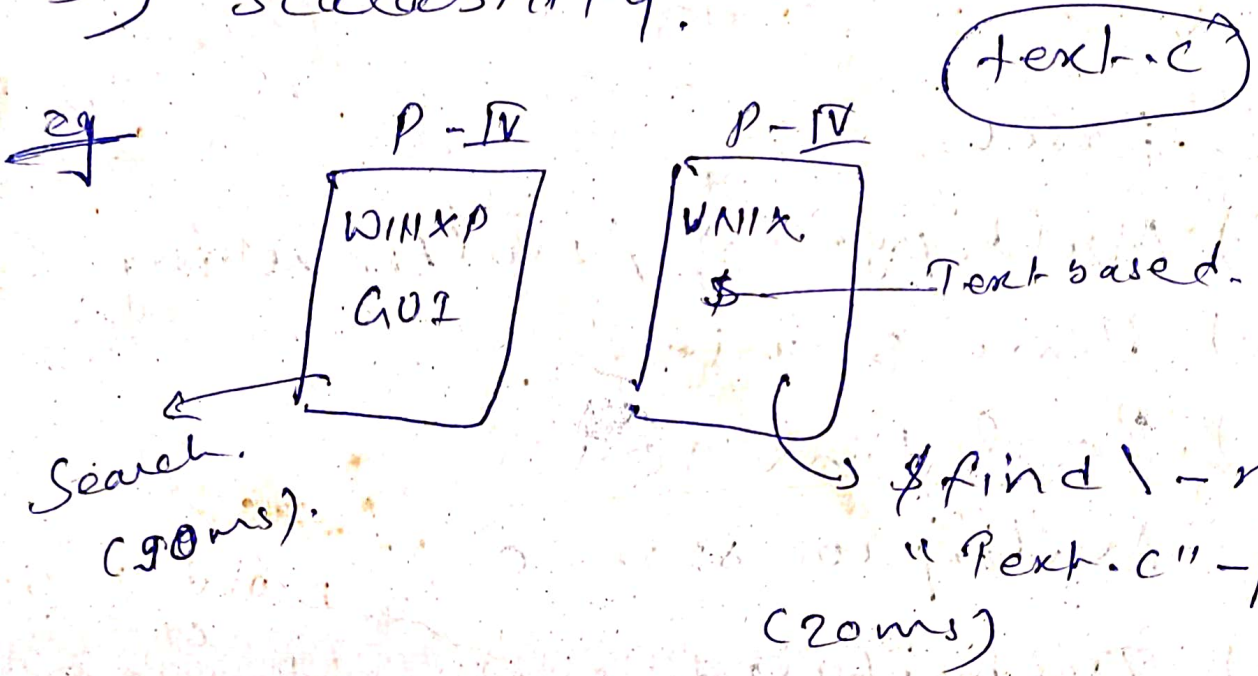
- CPU
- Memory.

* A resource is one which is directly under the ctrl of O.S. or may be a part of O.S.

~~Set of Utilities to simplify.~~

Goals of an O.S. →

1) Efficiency.
2) Convenience
3) Reliable & Robust.
4) Portable.
5) Scalability.



* Convenience is the primary goal of the O.S. But in such environment the O.S. is required the efficiency like - Military sys, Rtos sys. real time environment,

* Portable means the O.S. is not binded for any particular m/c.

* Scalability means ability to adding of new pgms or ability to evolve.

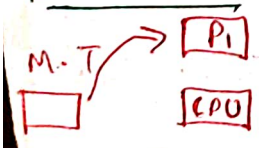
Types of O.S →

1). Batch.
2) Multipgming / multitasking
3) Time sharing
4) Real time O.S
5) Multiprocessor O.S
6) Distributed O.S.
7) Embedded O.S.

Conventional O.S

Modern O.S or Advanced O.S

Batch → Group of similar task to perform any job.

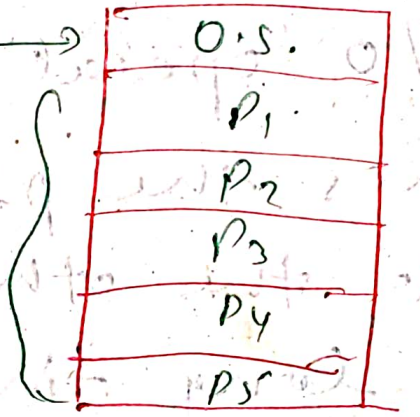


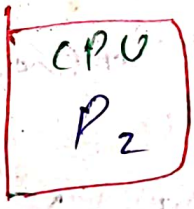
like – payroll. It is unipgming concept. When an error occur the CPU becomes idle.

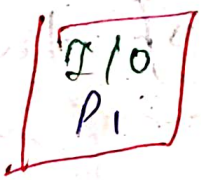
Multiprogramming → Ability to have multiple ready pgm for the pgm in the main memory so that if the running pgm required any I/O operation then CPU can switched to any ready pgm from the main memory

Multiplexing of M.M.

Multiplexing of the CPU within diff. pgms is kla M.P.







MP

Preemptive.

Non preemptive

Non preemptive → The CPU is not release the process under any ~~pressure~~ pressure. It release under its own choice. It is release the process only when two conditions satisfy -

- Complete the process.
- I/O operation required.

* But in this the prob. is arises that the other process waiting a long time until the completion of that process.

Preemptive — It is of two type.

- Priority.
- Time.

Priority - When the CPU release the process beco of the priority basis..

Time — ~~When~~ There the times are alloted for the every processes. & when the CPU.

4) Real time O.S. → strict ^ Deadlines.

eg Nuclear c.r; Mobile Sys. stealite launching.

5) Distributed O.S. → N/w + transparency

6) Multiprocessor O.S → Managing the sys. more than 1 CPU.

7) Embedded O.S. →

eg. - cell phone.
- palm top.

The development of new small hand device like palmtop with limited func.

* for O.S. process is the central locus of the control. b'coz process is not there the O.S. becomes idle.

Date
8/10/06

Process Mgmt

(i) Process Concept →

- Program under execution.
- Unit of execution
- Schedulable / Dispatchable unit.
- Instance of pgm.
- animated spirit
- Locus of ctrl.

Pgm Vs Process

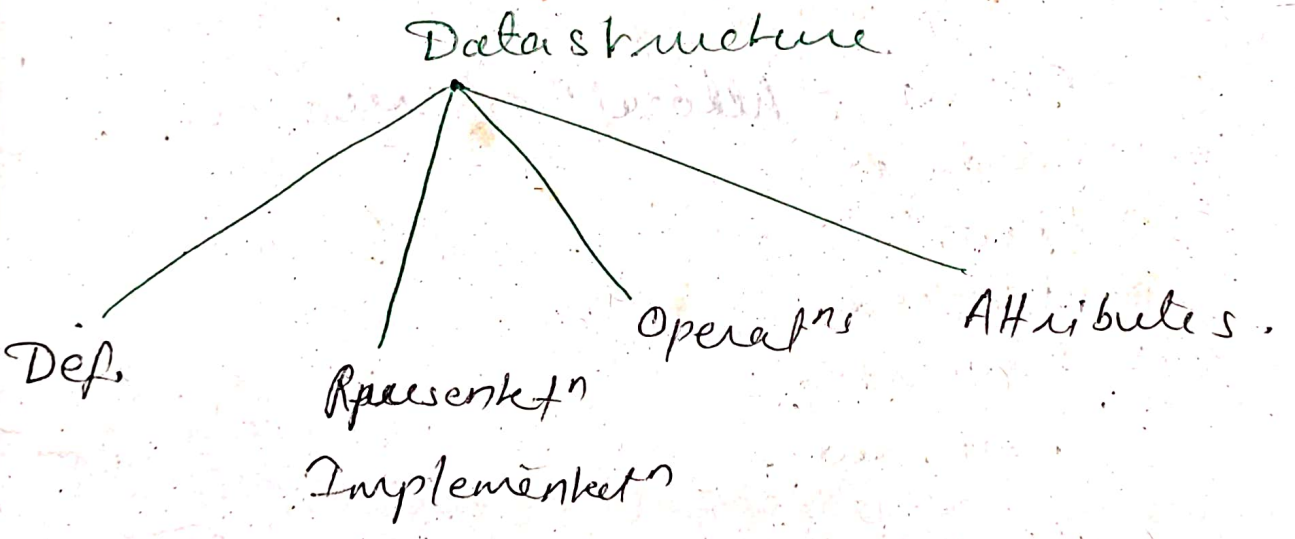
Pgm	Process
- Passive Entity	- active Entity
- Without Internal Resources	- With resources
- Disk	- Main mem.

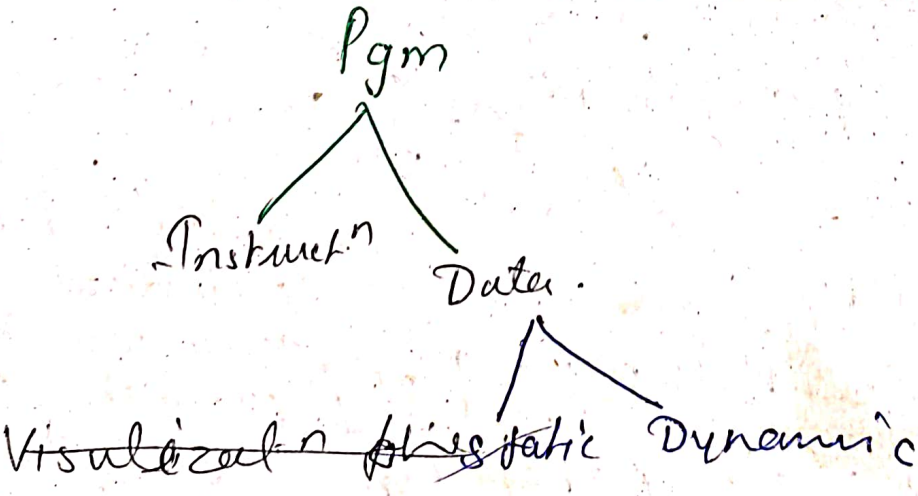
* Without pgm there is no process.

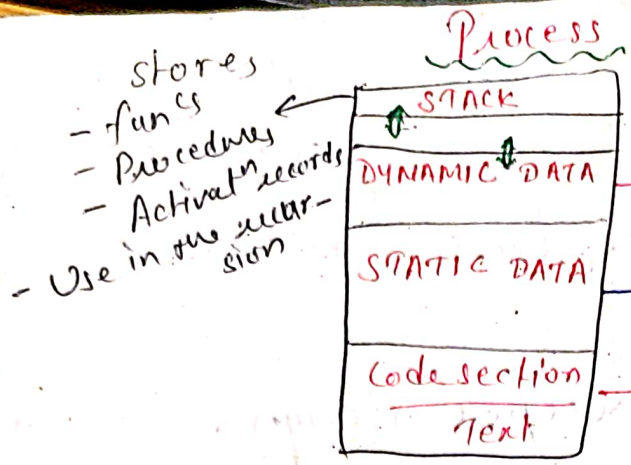
* Execution → When any process is utilize any one or more resources of the comp. like - mem, CPU, I/O.

* Designer or developer designs the O.S who estimulates the process.

Designers' view of process — Process is nothing but the Data structure.







↕ → It is varying.

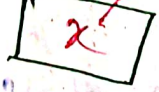
The data which are allocated during load time is falls into the static data.

Operatns of a process →

- Create – Allocatn of resources.
- Shedule
- Run
- blocked
- Suspended
- Resumed

Process Attributes →

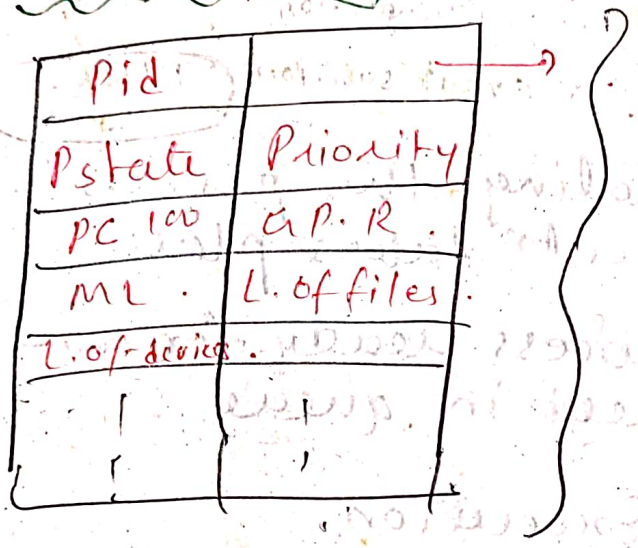
- id (process id)
- state
- Priority
- Pgm Counter (PC)
- GPR (general purpose Reg.)
- Mem limits
- list of open files
- List of open Devices.



The value of the PC indicates that which instructn is going to be execute.

- Protection info.

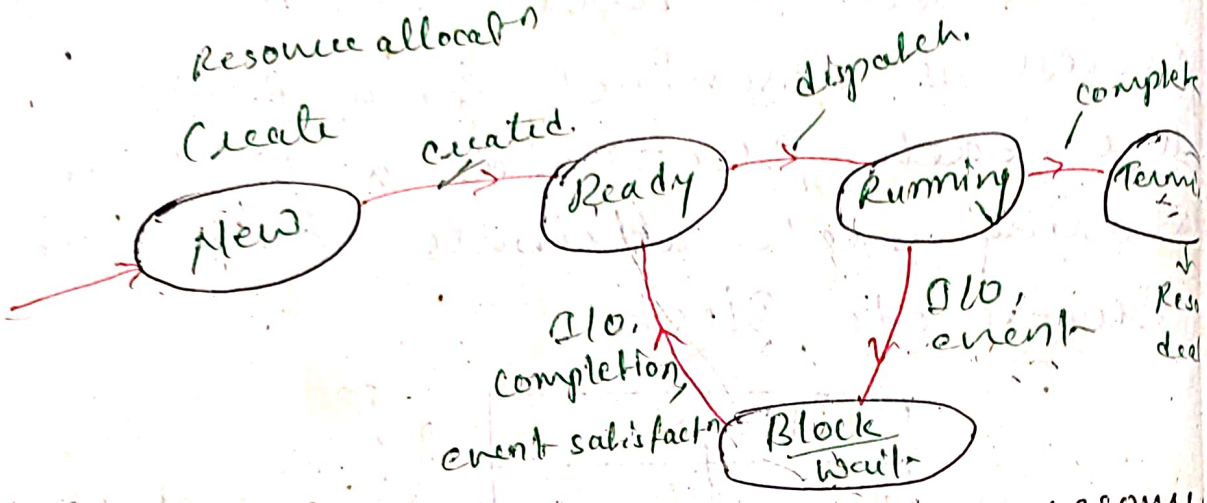
These all the above attributes of the process is essential for a process So, the each process has a block for storing such attributes k/a the Process ctr Block.



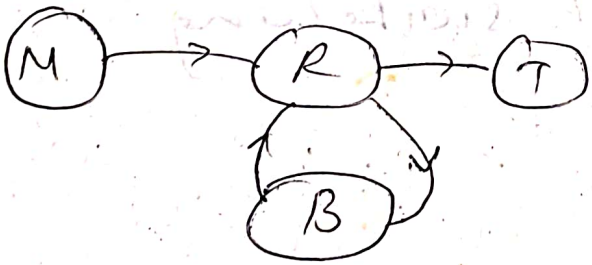
Set of the values of the attributes are combinely said as the content of a process.

When we switch from the one process to the another then it is k/a content switching.

Process state & state transition Dia-



* New - Creating the process i.e. resource allocatn takes place
* Ready - Process ready for execut^n are placed in queue.
* Running - Execution
* Terminated - resource deallocatn.
* Block/wait - Not terminated but needs some I/O operatn or some event.
* In the Batch O.S. there is only one process in the mem.



* Multiprocess O.S. means more than one running states.

Ans The given transition diagram is referring to the which O.S —

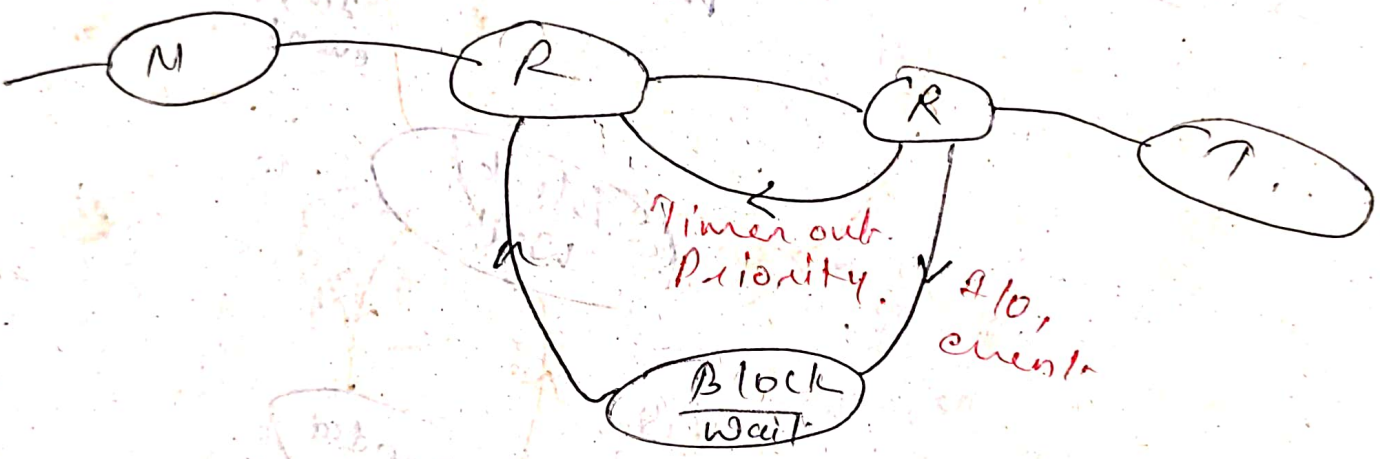
a) Batch O.S.

b) Preemptive M.P. O.S.

c) M.P. —"— O.S.

d) Multiprocessor O.S.

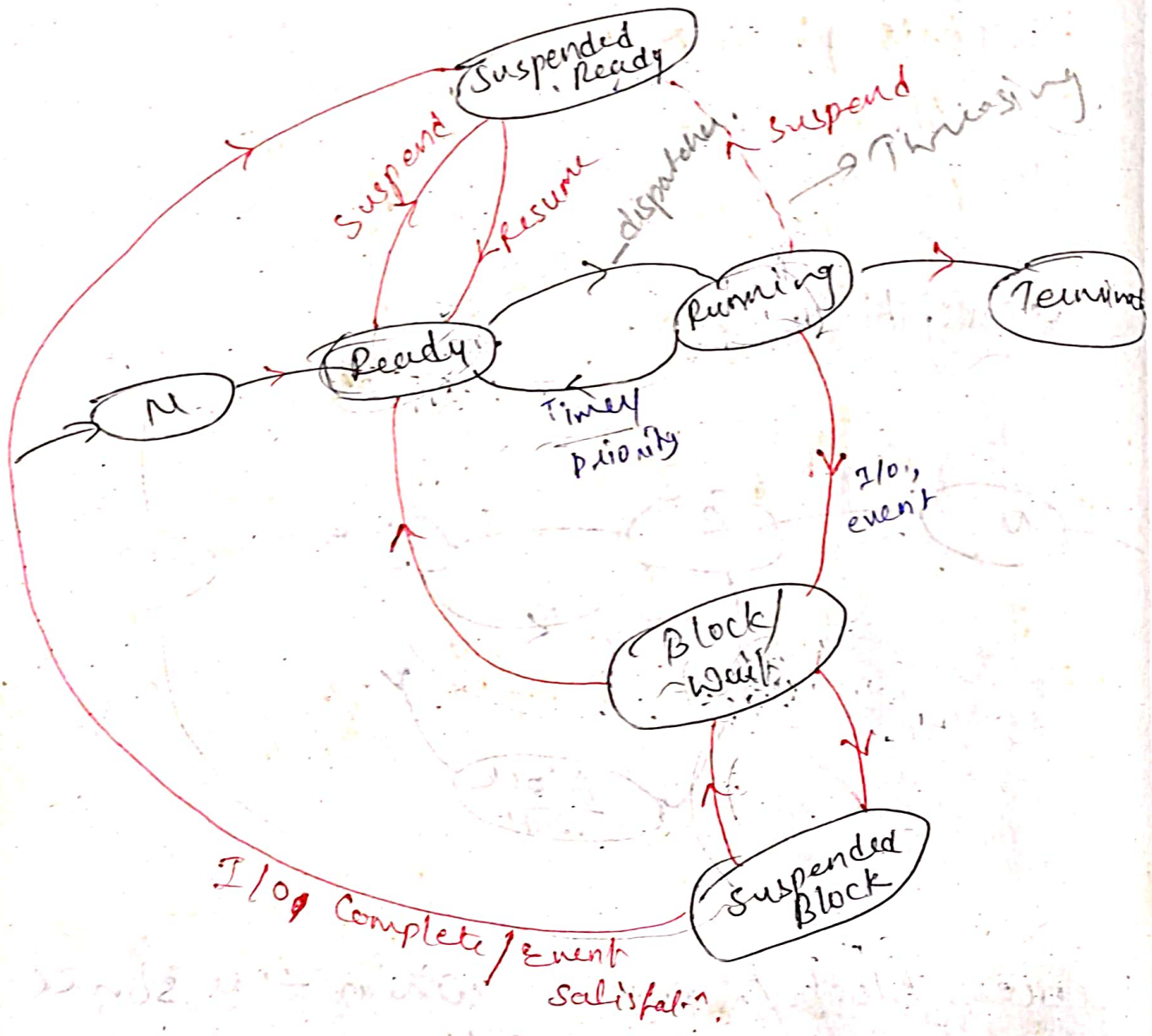
Preemptive O.S.



Process Block/wait → when the some I/O, event. requied then the process is block.

Suspended → When the resources are not sufficient for the process & degrade the efficiency of the comp. Then we have to suspend some process. Means it. through the process from

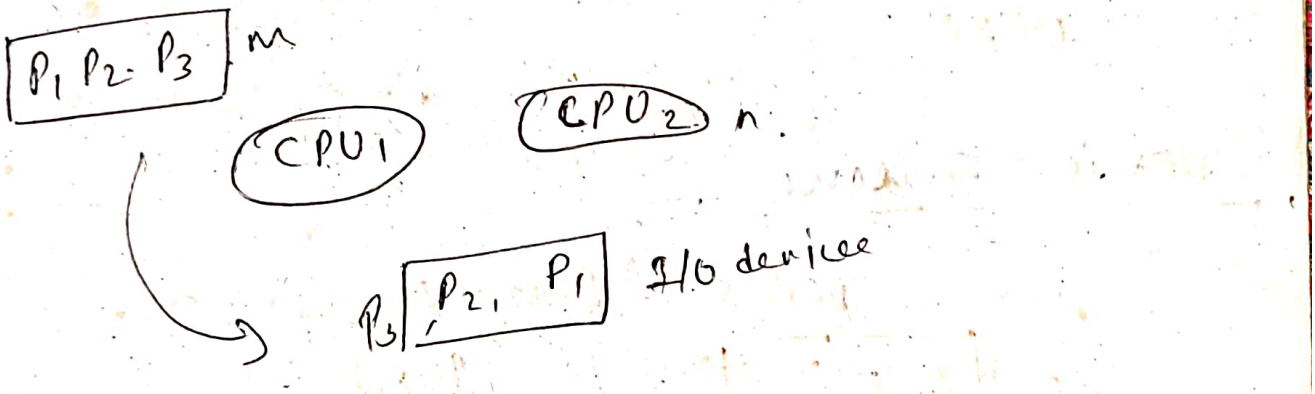
the main mem. to the second mem. eg: when the not sufficient mem. is available.



This is the state transition dia.
of Preemptive, multipgming O.S
& this is a deterministic finite
state automata. (DFA).

Q. Consider a CPU with n CPUs with m processors. where m > n, what is the lower bound & upper bound on the no. of process that can be in ready, running & block state.

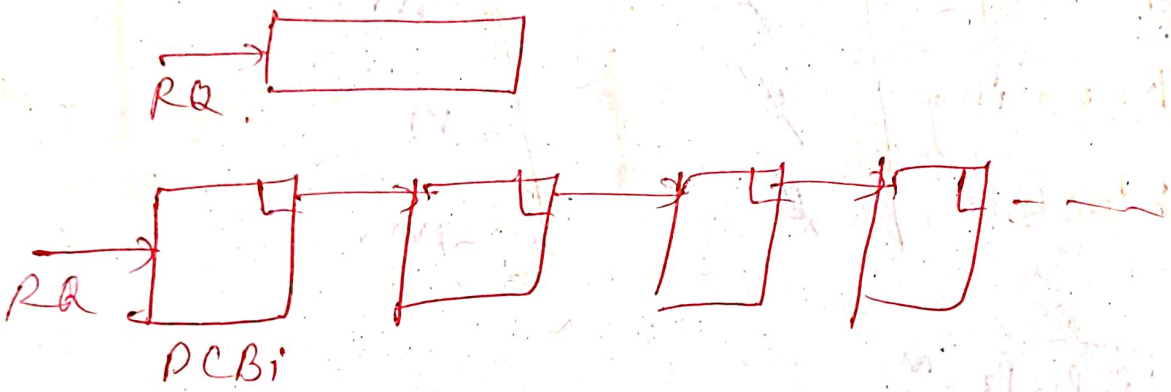
	Min	max.
Ready	∅	m.
Running	∅	n
block	∅	m.



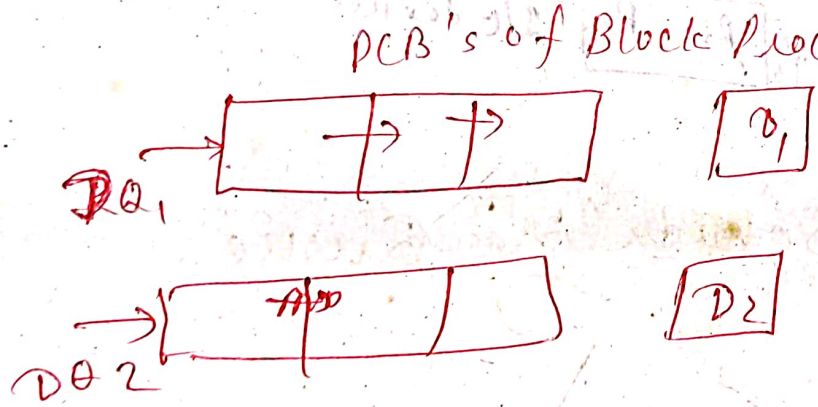
Scheduling Queue

- Ready Queue (Ready to run process)
- Block Queue [I/O, event]
  - Device Queue
  - Event Queue.

Ready Queue It is simply a linked list to point the another pcbi. Its implementation mainly in the doubly linked list. B'coz the deletion & insertion is easy.

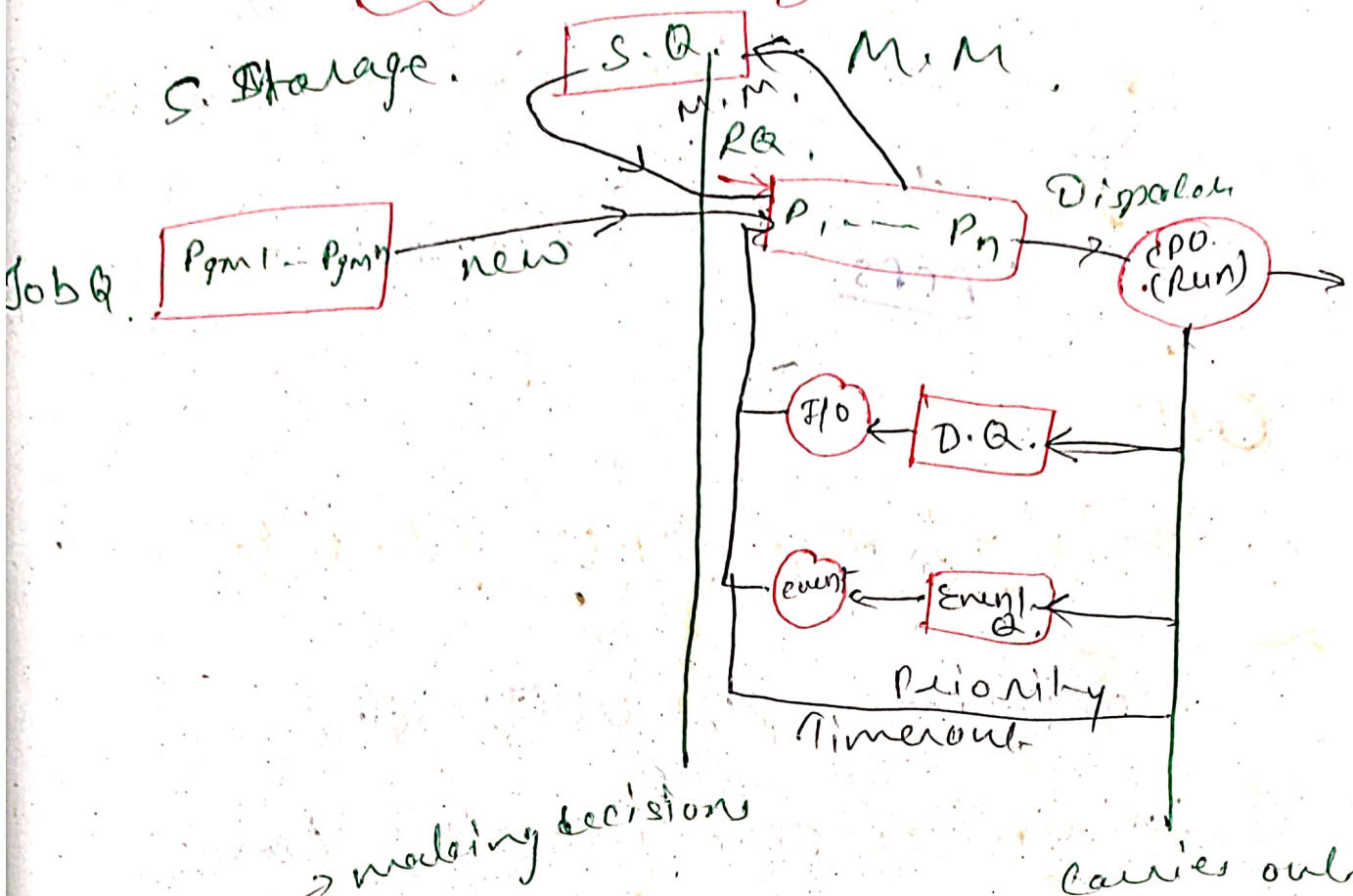


Device Queue —



Event Queue —

State Queuing Diq



Schedulers & Dispatchers

Job scheduler.

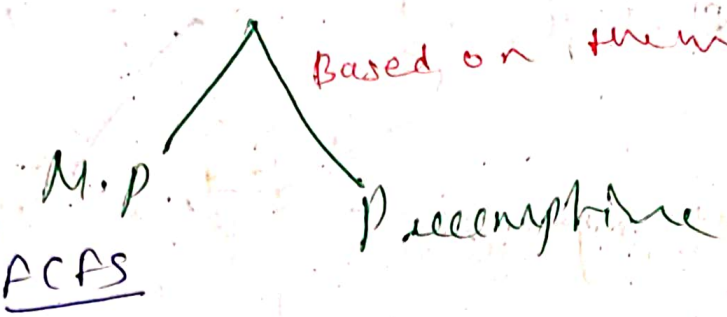
1) Long term. It takes process fm the pool & assigns into the mem. It is responsible for the degree of multiprgming.

2) Short term sche. also k/a the CPU scheduler.

It may use the some criteria if uses the scheduling to making decisions. Dispatcher is works with the CPU scheduler.

3) Medium term sched. — ~~Which term is doing the~~ is part of the swapping It removes the processes from the It reduces the degree of MP. It is inchange handling swapping

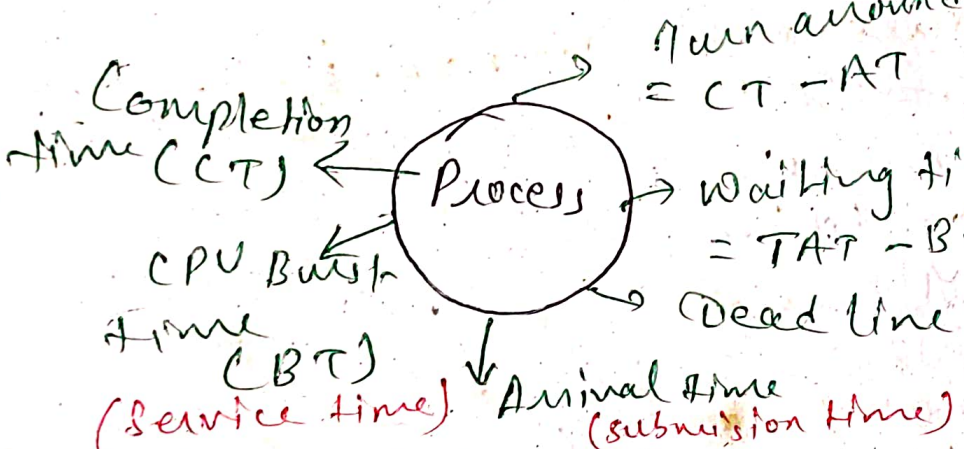
2. CPU scheduling -



~~CPU~~

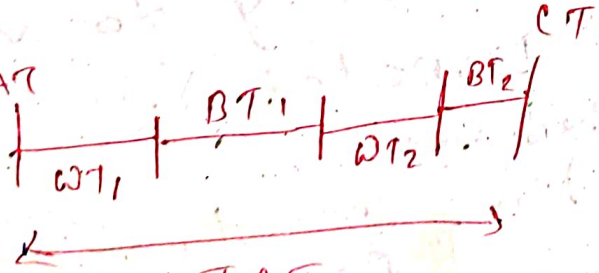
Goals of the CPU scheduler →

1) Maximize CPU utilization
2) Minimize the waiting time.



* The arival of the process in the ready queue is b/c Arival time

*

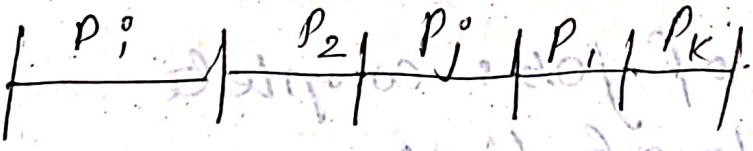


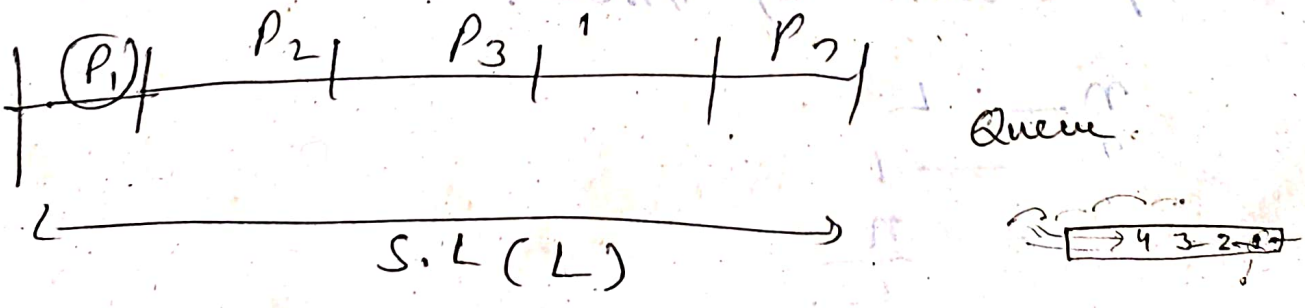
Let 'n' process (P1 - - - Pn)

- A.T P_i → Ai
- B.T P_i → Xi
- C T_{Pi} → Ci
- TAT P_i → (Ci − Ai)
- Wt. TAT P_i → $\frac{(Ci - Ai)}{Xi}$
- Avg. TAT (n process) → $\frac{1}{n} \sum_{i=1}^{n} (Ci - Ai)$
- Avg. Wt. TAT → $\frac{1}{n} \sum_{i=1}^{n} \frac{(Ci - Ai)}{Xi}$

Let schedule length of 'n' processes (L) →

n process schedules ⇒ n!





schedule length = max(Ci) − min(Li)

Deadline overrun = $\frac{C_i - D_i}{P_i}$ (D = Dead

Over deadline = when the ans. is in '+ve' then it is kla .. over dead line. ~~overrun~~

Under dead line = when the ans. is '0' or '-ve' then it is kla Under dead line.

Throughput of the Sys. = $\frac{n}{L}$

(efficiency)

↳ The no. of jobs complete per unit of time.

n — L
1 — $\frac{L}{n}$

Response Time → The time taken by process to generate a initial response from the time of submission of request of

Goals of a CPU scheduler

1) Max. CPU utilization
2) minimize the
   - Waiting time
   - TAT
   - Response time
3) fair. → The CPU must be fair enough to save all the process means unbias

→ If CPU is not fair, the process are going into starvation.

2. CPU scheduling Tech.

1) FCFS (first come & first serve)

Criteria – A.T.

Mode – N.P.

$$TAT = CT - AT$$

$$WT = T.A.T - BT$$

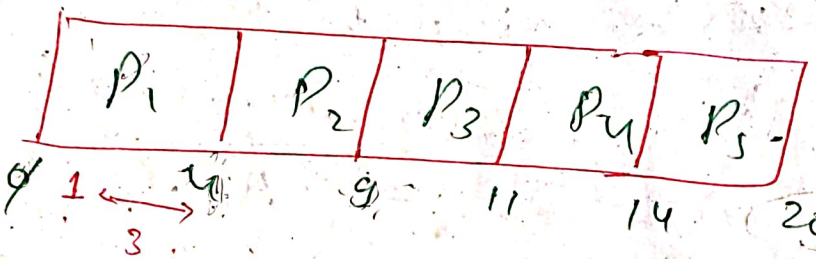
eg

P. No	A.T	B.T
1	0	4
2	1	5
3	2	2
4	3	3
5	4	6

Calculate C.T, T.A.T, W.T. ?

Soln

Gantt chart.



C.T	T.A.T	W.T
4	4	0
9	8	3
11	9	7
14	11	8
20	16	10
	x	y

TAT = CT − A

WT = TAT −

$$\text{Avg. TAT} = \frac{x}{n} = \frac{x}{5}$$

$$\text{Avg. WT} = \frac{y}{n} = \frac{y}{5}$$

$L = 20 - 0$

$= 20$

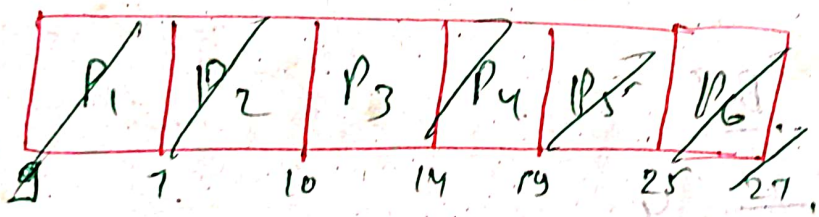
$\mu = \frac{5}{20} = \frac{1}{4} = 25\%$

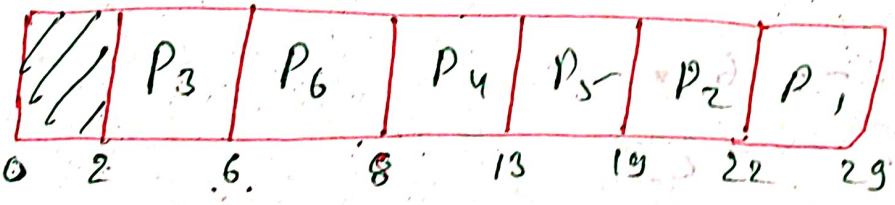
Que

Pid	AT	BT	CT	TAT	W.T
1	0	7			
2	1	3			
3	2	4			
4	4	5			
5	5	6			
6	3	2			

TAT = CT - AT.

WT = TAT - BT





TAT(P_2) = 22 - 8 = 14.

WT(P_2) = 14 - 3 = 11

Schedule length L = 29 - 2 = 27.

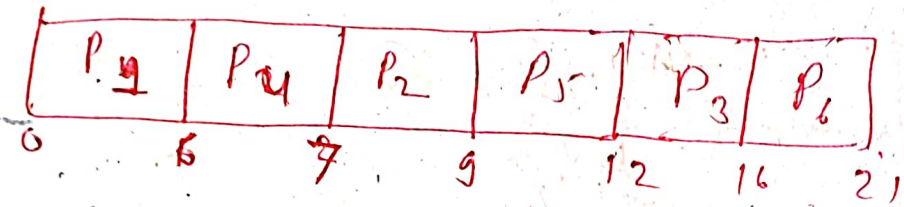
2) { Shortest-job-first (SJF)
   { shortest-process-Next (SPN)

Criteria : BT.

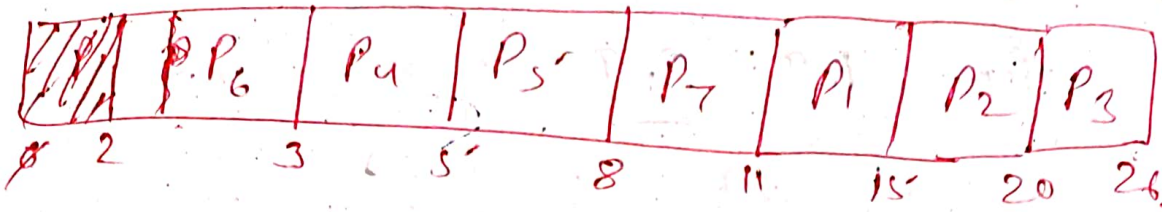
Mode : NP (By default)

Pid	AT	B.T
1	0	6
2	1	2
3	2	4
4	3	1
5	4	3
6	6	5

Gantt Chart -



Pid	AT	BT
1	3	4
2	3	5
3	4	6
4	2	2
5	5	3
6	2	1
7	6	3



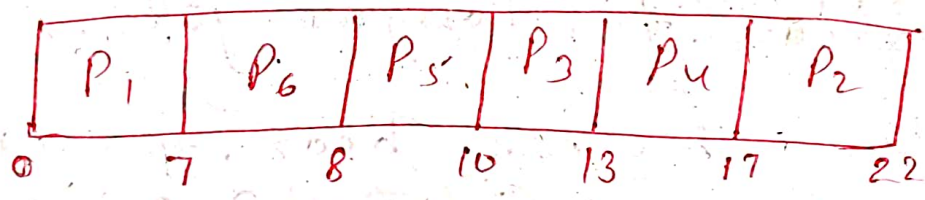
* Shortest-remaining Time first (SRTF).

Criteria - B.T.

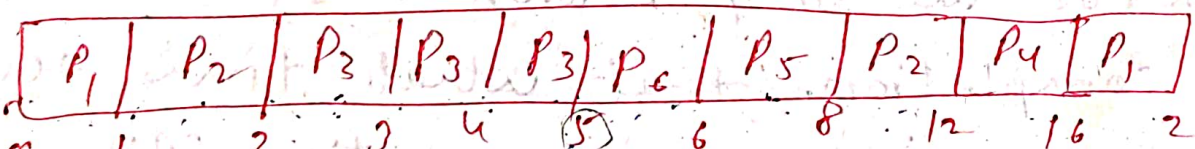
Mode : Preemptive.

P.No	AT	B.T
1	0	7
2	1	5
3	2	3
4	3	4
5	4	2
6	5	1

for NP (SJF)



for SRTF (P-SJF).

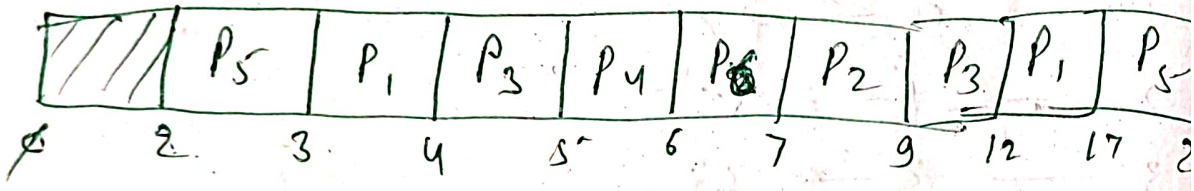


TAT (P5) = 10 - 4 = 6

TAT (P5) = 8 - 4 = 4

eg

PN	AT	B.T	
1	3	6	5
2	6	2	2
3	4	4	3
4	5	1	x
5	2	8	7
6	5	1	x



Performance Analysis →

A 1) Optimal — max. Thruput.

2) Starvation to longer jobs.

3) Implementation b'coz. the criteria is B.T.

* B.T. is the not known in the advance b'coz we only known this after the completion. that what time the process is running in the CPU.

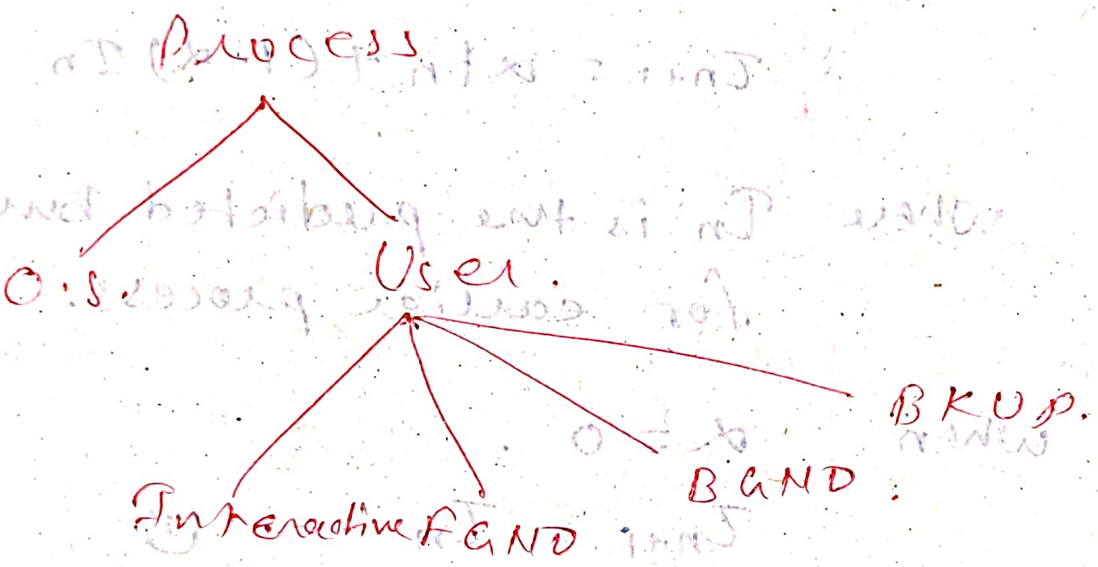
* But we predicting the Burst time. So that we can apply the shortest path first.

Prediction →

P1 → (12KB) → 5 units

① Size → P1 → (10KB) → 5 units.

② Type →



③ Avg. of the previous completed B.T.

$P_1 - 5$

$P_2 - 6$

$P_4 - 4$

$P_5 \rightarrow ?$

$? = (x/4)$

④ Weighted Avg. → (Exponential Averaging)

Date 1/10/06

Shortest Job First

Exponential Averaging Algo. for predicting the Next CPU Burst...

Let $\tau+1$ be the predicted burst time of the arrived new process, let t_n is the burst time of just completed process.

then $0 \leq \alpha \leq 1$

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

where τ_n is the predicted burst time for earlier process.

when $\alpha = 0$.

$$\tau_{n+1} = \tau_n \quad \text{---} (1)$$

when $\alpha = 1$

$$\tau_{n+1} = t_n \quad \text{---} (2)$$

when ~~$\tau_n = t_n$ then the~~ we gives the eqn (1) & (2) the equal weightage then

$$\tau_n = t_n$$

then $\alpha = 0.5$.

$$\tau_{n+1} = \frac{1}{2} t_n + \frac{1}{2} \tau_n$$

$$= \frac{1}{2} (t_n + \tau_n)$$

$$\tau_{n-1} = \alpha t_{n-2} + (1-\alpha) \tau_{n-2}$$

$$\tau_n = \alpha t_{n-1} + (1-\alpha) \tau_{n-1}$$

$$\tau_{n+1} = \alpha t_n + (1-\alpha) [\alpha t_{n-1} + (1-\alpha) \tau_{n-1}]$$

$$= \alpha t_n + \alpha(1-\alpha) t_{n-1} + (1-\alpha)^2 \tau_{n-1}$$

$$= \alpha t_n + \alpha(1-\alpha) t_{n-1} + (1-\alpha)^2 [\alpha t_{n-2} + (1-\alpha) \tau_{n-2}]$$

$$= \alpha t_n + \alpha(1-\alpha) t_{n-1} + \alpha(1-\alpha)^2 t_{n-2} + (1-\alpha)^3 \tau_{n-2}$$

So called as the Exponential Averaging algo.

Eg.

$\alpha = 0.5$

$\tau_4 = 10$ $\qquad \tau_5 = ?$

t_1, t_2, t_3, t_4

$4, 8, 3, 5$

Soln

$$\tau_5 = \alpha t_4 + (1-\alpha) \tau_4$$

$$= 0.5 * 5 + 0.5 \tau_4$$

$$\tau_4 = 0.5\,r\,(3+\tau_3)$$

$$\tau_3 = 0.5\,r\,(8+\tau_2)$$

$$\tau_2 = 0.5\,r\,(4+10)$$

$$= (7)$$

Highest Response Ratio Next (HRRN)

Mode – N.P.

Criteria – Reponse Ratio (Rr)

$$= \frac{w+s}{s}$$

w = waiting time of process so far.

$$s = \frac{\text{Service time}}{\text{B.T.}}$$

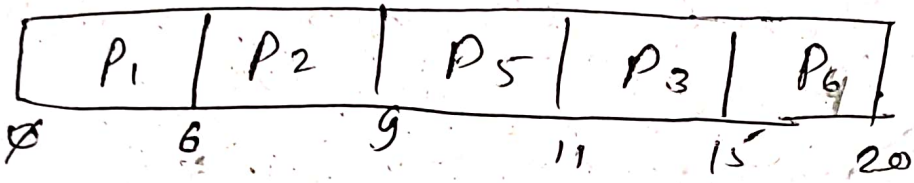
→ waiting time never '-ve'.

→ It favours longer jobs & shorter job.

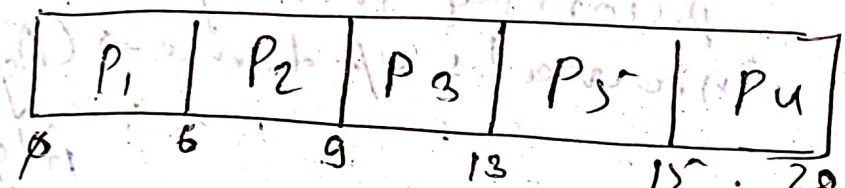
→ The minimum response time is '1' when the waiting time is '0'.

P. No.	A.T.	B.T
1	0	6
2	2	3
3	4	4
4	6	1
5	8	2

SJF – NP.



HRRN – NP.



(I) At = 6

$RR_2 = \frac{4+3}{3} = \frac{7}{3}$ ✓ (where $w_2 = 6-2 = 4$.)

$RR_3 = \frac{2+4}{4} = \frac{6}{4}$ $w_3 = 6-4 = 2$

$RR_4 = \frac{0+5}{5} = 1$ $w_3 = 6-5 = 0$

(II) At = 9

$RR_3 = \frac{5+4}{4} = \frac{9}{4}$ ✓ $w_3 = 9-4 = 5$

$RR_4 = \frac{3+5}{5} = \frac{8}{5}$

$RR_5 = \frac{1+2}{2} = \frac{3}{2}$

AI = 13

$$RR_4 = \frac{7+5}{5} = \frac{12}{5}^{\wedge}$$

$$RR_5 = \frac{5+2}{2} = \frac{7}{2} \checkmark$$

Priority Based Scheduling -

Static

Untill end of execution it will be same

Dynamic

It will change the priority in regular intervals of time → (Aging Algo)

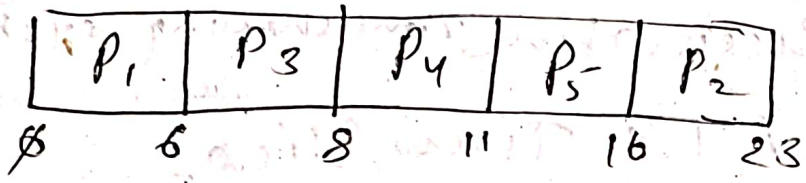
→ But we are using only static method.

(i) N.P

Criteria - Priority.

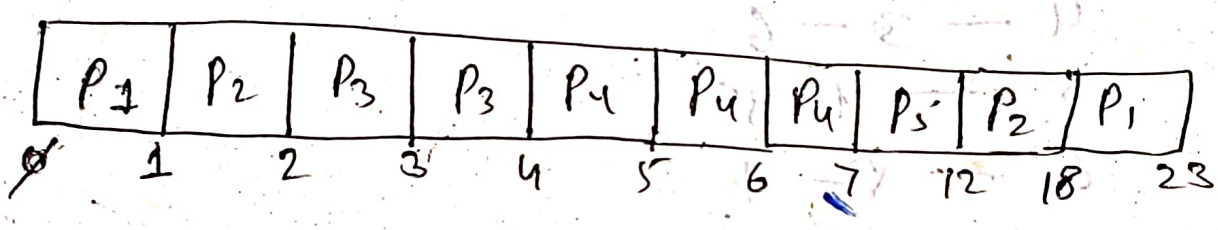
Pid	PNo	A.T.	B.T.
4	1	0	6
5	2	1	7
8	3	2	2
7	4	3	3
6	5	5	5

NP - Priority —



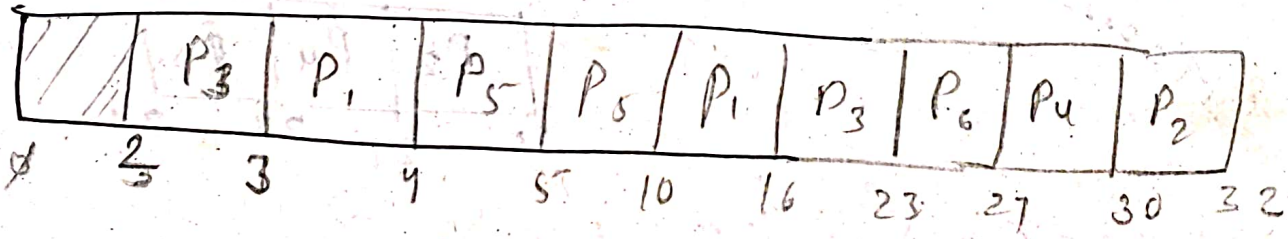
(ii) Preemptive Priority → The process which is running with a lower priority is discarded when the new process is arrived with high priority.

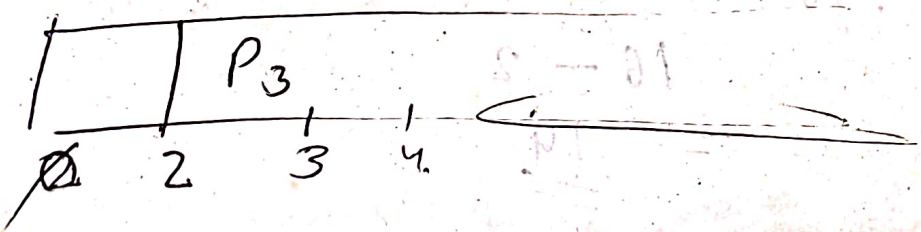
Prep - Priority



Prio	PNo	AT	BT	
8	1	3	7	✓
2	2	7	2	✓
6	3	2	8	✓
4	4	6	3	✓
9	5	4	6	✓
5	6	5	4	✓

Pre - Priority





Round Robin (Preemptive FCFS) [Fi

→ Multiuser, Multipgm

Mode : Preemptive Timesharing system

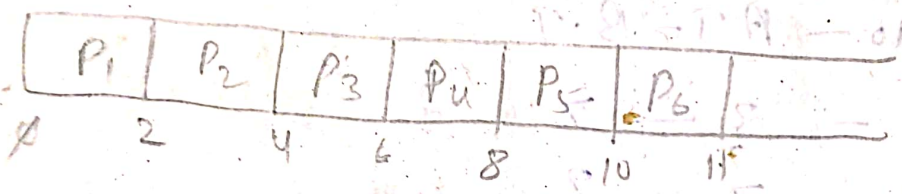
multiProcessor

Criteria : AT + Time Slice.

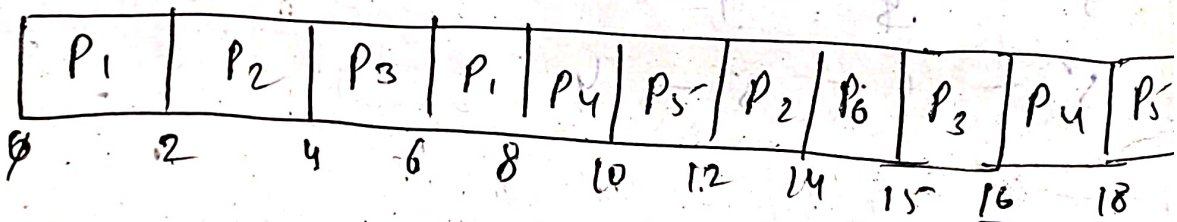
PNo.	AT.	B.T.	
1	0	4	2 ✓
2	1	5	1 ✓
3	2	3	1 ✓
4	3	6	2 ✓
5	4	3	1 ✓
6	5	1	✓

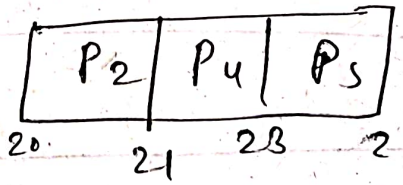
TQ = 2

X



Ready Queue → P1, P2, P3, P1, P4, P5, P2, P6, P3, P4, P5, P2, P4, P5





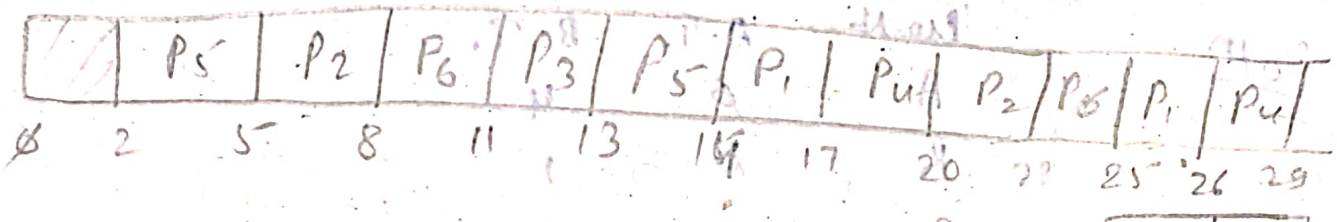
TAT(P3) = CT − AT

= 16 − 2

= 14

P No	A.T	B.T.
1	6	4 1 X
2	3	5 2 X
3	5	2 X
4	7	8 5 2 1
5	2	4 1 X
6	4	7 4 1 X

TQ = 3.



R.Q = P4, P2, P6, P3, P5, P1, P4, P2, P6, P1, P4, P6, P4

Performance — (R.R) depends on the Time Quantum.

Time Quantum

Too Large (FCFS)

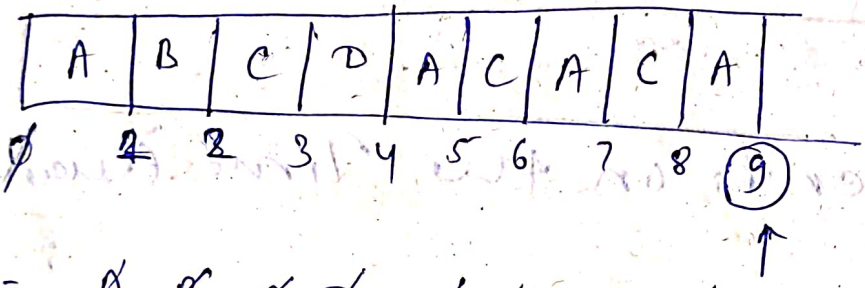
Small	Large
1. More context-Switches	1. less context-Switches
2. More interactive	2. Less Interactive.
3.	3.

XSRV.

Que Consider a set of four jobs A, B, C, D arriving at the time ∅ with predicted burst times of (4, 1, 8, 1) Cal. the CT of job A under the R.R. sched with the Timeslice of 1 un

Soln

Pro. No.	A.T.	B.T.
A	∅	4
B	∅	1
C	∅	8
D	∅	1



R.Q = A, B, C, D, A, C, A, C

Ans 9

Que Consider a set of n processes sharing the CPU in the RR fashion the context switch over or context switching time is gi as 'S'. what must be the TQ 'Q' such that not only the

context switch overhead is minimized but also insured that each process is guaranteed to get its turn at the CPU atleast after every 't' sec.

Soln

a) $q \leq t - ns$

b) $q \leq \frac{t - ns}{n-1}$

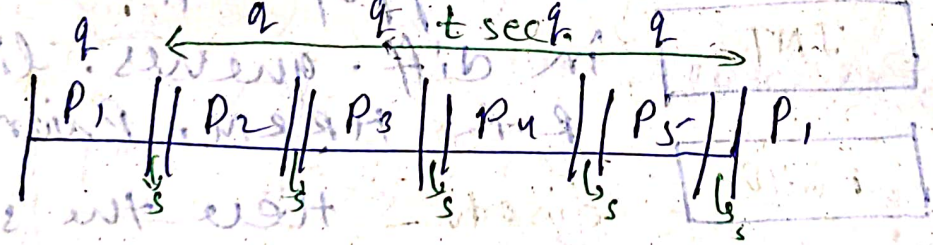
c) $q \geq \frac{t - ns}{n-1}$

d) $q \geq \frac{t - ns}{n+1}$

'n' - processes
's' - context switch Time
'q' - Time quantum
't' - sec.

Ex RQ : (P_1 - - - P_5)

$n = 5$, (P_1 - P_5)



→ the we are spending time for computation & context switching.

→ Each 's' is run for n time. means ns

→ So $q = \frac{t - ns}{n-1}$

but atleast after each t sec.

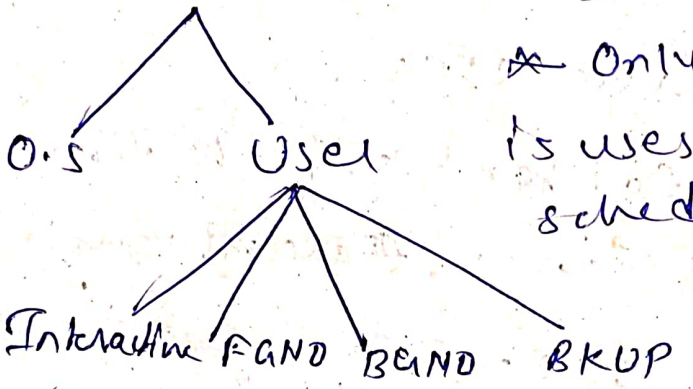
So $q \leq \frac{t - ns}{n-1}$

Multilevel feedback Queue scheduling

Single Queue →

RQ [ ]

All the Processes (PCB's)



* Only one queue is uses only one scheduling

Multilevel Queue —

→

highest priority

lowest

O.S

INT

FGND

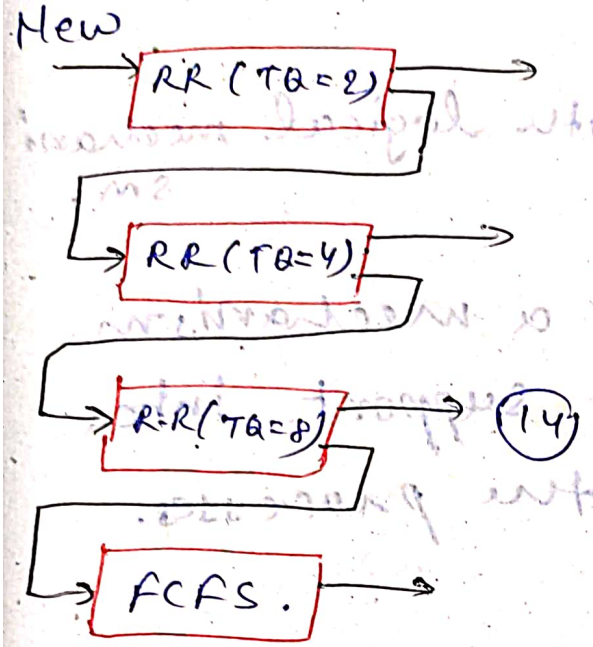
BGND

BKUP

Multiple Queue — we can apply diff. schedul in diff. queues. like R.R, HRRN, Priority disadv — Here the star vation is the disadva bcoz the O.s has the most priority & priorit is in order in the queue, O.s, INT, FGND, BGND, BKUP.

Multilevel feedback Queue →

This is with the feedback.

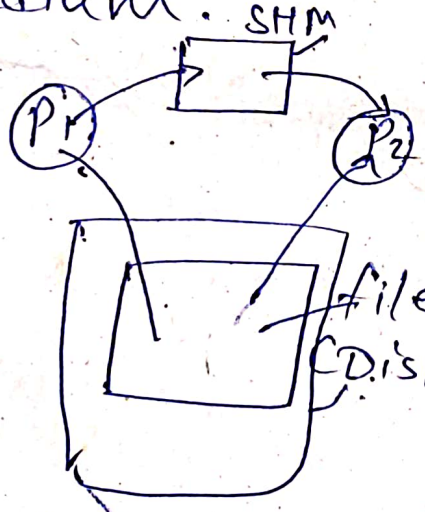


→ starvation is also present here.

→ But in this process we are providing a chance to get execution.

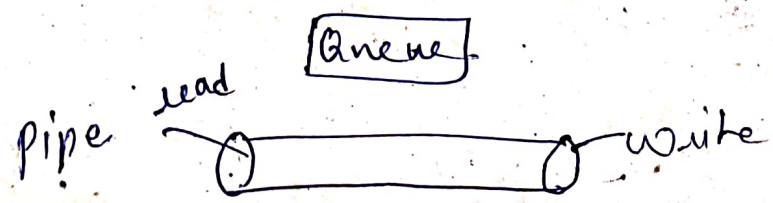
I.P.C & Synchronization ⟹

Process must communicate each other for this the basic requirement is Medium.



→ Generally we don't prefer a file b'coz it required much time for the access. But file is good medium.

→ Apart from local address space of the process we are providing shared memory.



→ Pipe is logical file in main memory. It has two for read & write. There are n mechanism queue - monitor, file, pipe, etc.

→ Hence these an the diff. IPC mechanism are provided.

→ We are going to deal with logical mechanism

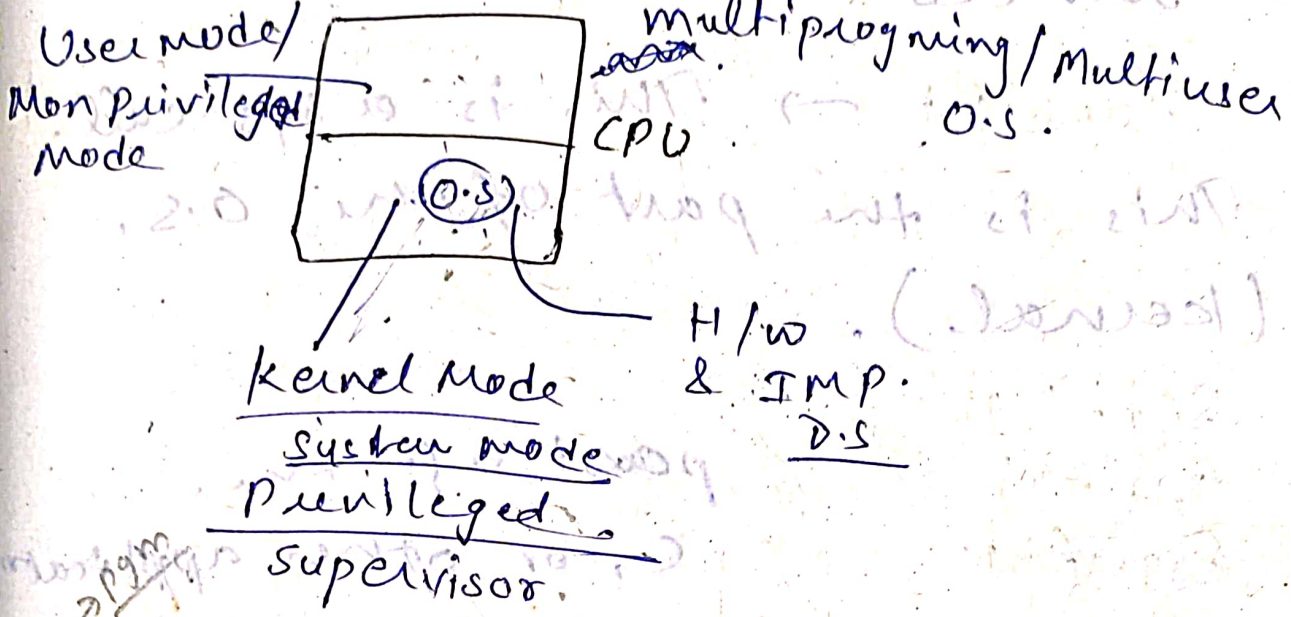
IPC mechanism → Is a mechanism that support inter process comm. b/w the processes.

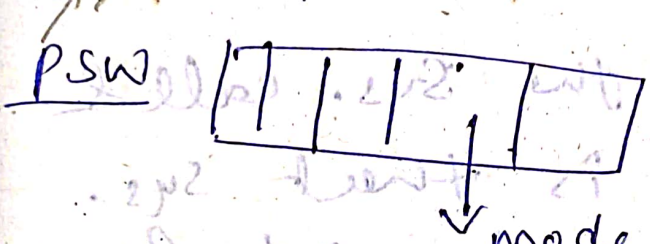
Synchronization Problems →

When we are communicating, the problem which we are facing in communication is known as synchronization problems.

System call Vs Library call

These two both are basically a fun^{cs}.





User Mode (0)

Kernel Mode (1) — mode bit

→ In kernel certain Priveleges will only run & no other pgms will run.

→ The pgms that runs on kernel mode coill give out H/w & implements of O.S. will be given.

→ So certain pgms must be run.

→ The multiuser / multiuser O.S provide the sys. call Interface which provides the list of all the services.

→ fork() → This is a sys-call This is the part of the O.S. (kernal).

→ Sqrt();
sin();
Scanf(); } part of the C, or other appl'n

* The diff. b/w. the sys. call & the Lib. call is that sys. call run in kernal mode & Lib. call run in user mode

→ main()
{ fork();
printf("Hello.");
}

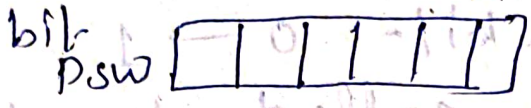
when fork(); It gives it prints "Hello." "Hello."

two times Hellow

Hello. Hello

Pgm status word →

PSW has several field and ith node



User mode – 0
Kernal mode-1 } mode bit

→ In the user pgm if we want some O.S. services then to access the O.S. service is said to be sys call.

→ It is a high level pgm and a call to the operating system for service.

→ It provides a interface.
i.e. API
SCI – List of all the services.

→ The implementation of system call is in operating sys. but a library call a call to the func which is implemented in user library in user mode.

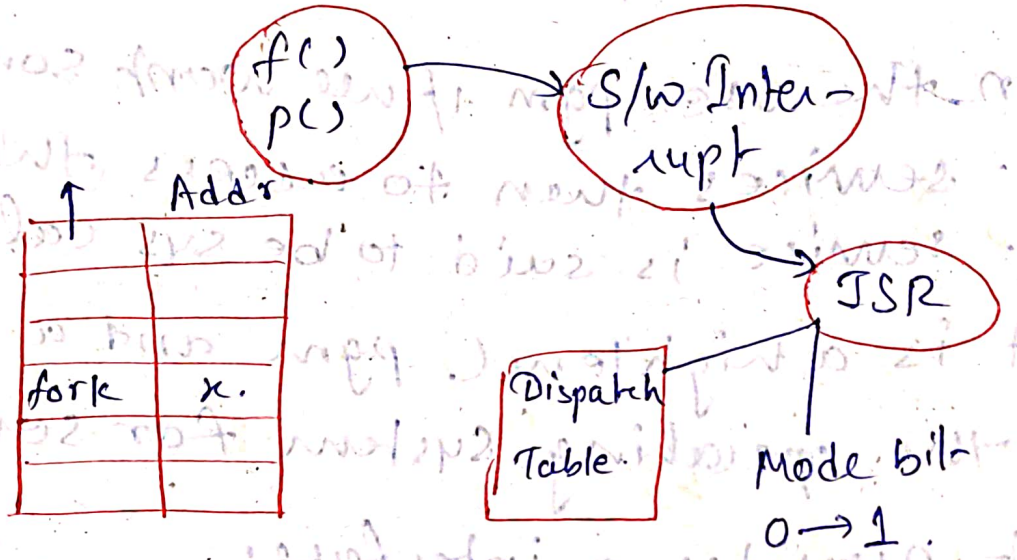
fork(); → O.S. system call

Sqrt()
sin()
Scanf() → User library (or) user mode library call.

S/W Interrupt → An Interrupt which is generated to execute an instruction is called s/w interrupt.

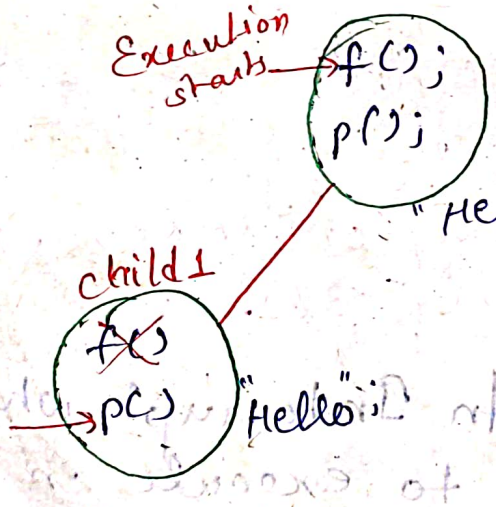
→ Every interrupt has its service interrupt routine (ISR).
It will change mode bit- 0 → 1.

→ It consults a table called dispatch table will have no. of entries.



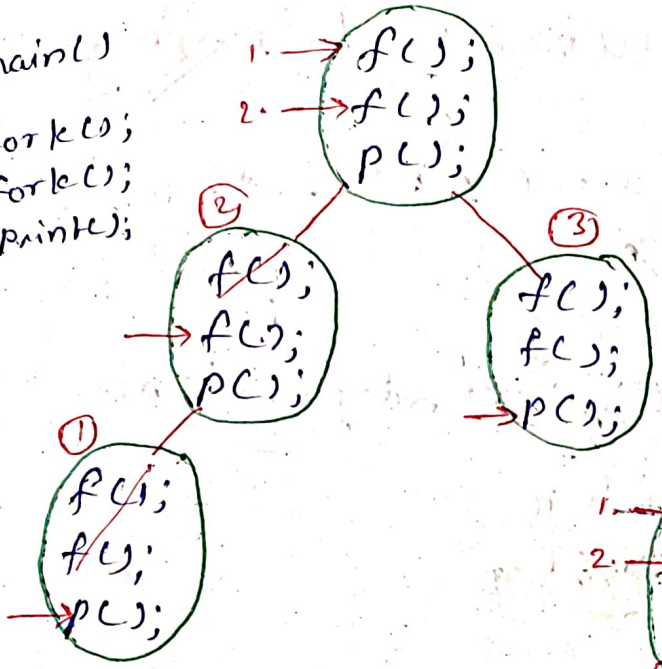
→ When fes, fork function is called it creates child process ch1 another pro The code of the child is same as parent process.

→ In the child process execution sta from instruction next to the fork But the code is same as parent



→ A single pgm is associate with 2 processes i.e. instance of the pgm.

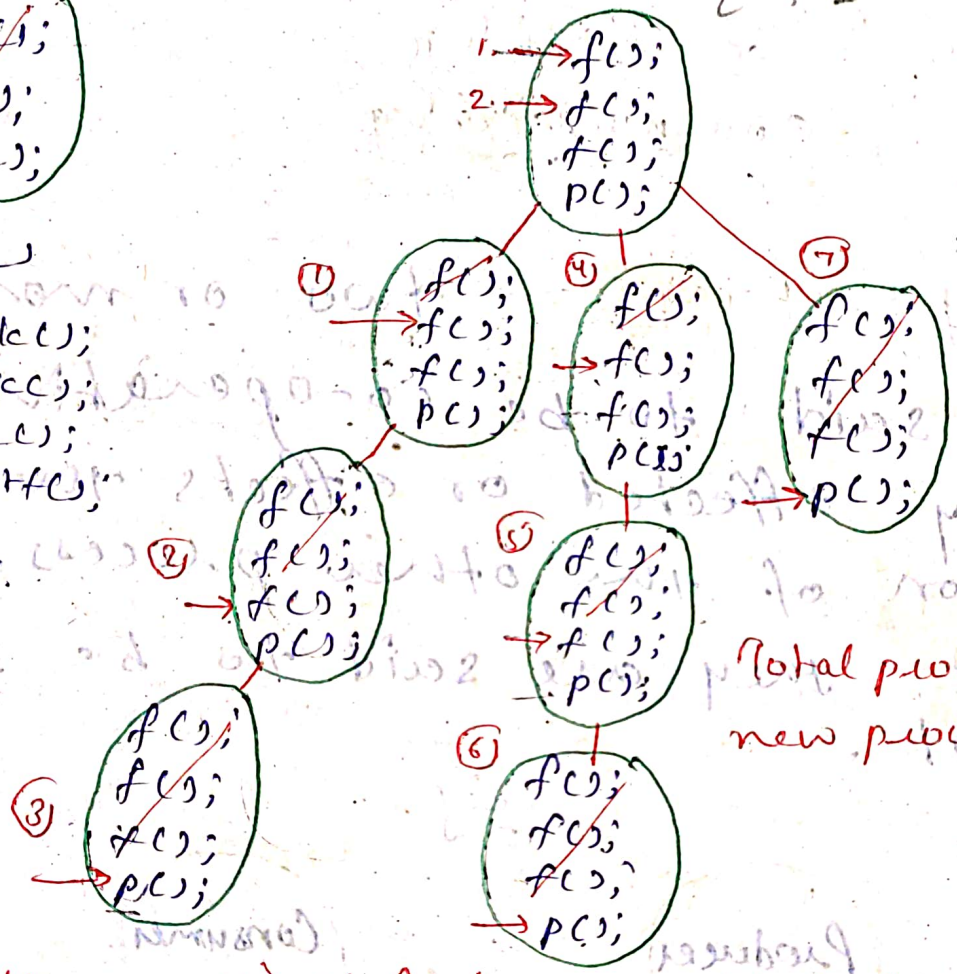
1) main()
{ fork();
fork();
printf();
}



Total processes = 4
new processes = 3.

2^n = 4 total
$2^n - 1$ child

2) main()
{ fork();
fork();
fork();
printf();
}



Total processes = 8
new processes = 7

* If there are 'n' forks, then total no. of processes $= 2^n$.

new no. of processes = $2^n - 1$

Que
main
{ int i, n;
for(i=1; i<=n; ++i) n+1
fork();
}

new process

a) $n!_0 - 1$ b) $2n-1$ c) $2^n - 1$ d) $n^2 - 1$