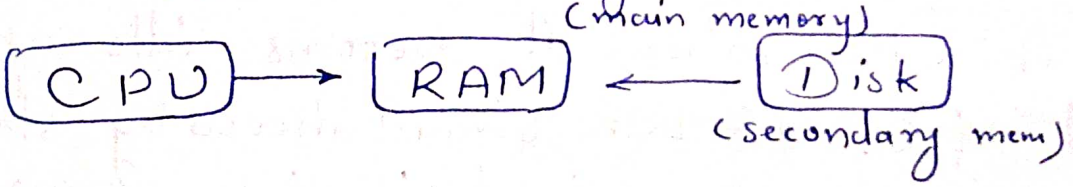
Operating System

Goals

- Easy to use / Good UI. → Primary goal
- Efficiency – Utilising max. resources. → Secondary

Function of Operating System

i) Process management.



Process vs Program.

Program in execution | → Passive set of instruction

ii) Memory Management (Main memory)

Allocations & deletion of diff^n tasks on main memory. & the address.

iii) File Management

iv) Secondary Memory / storage managent.

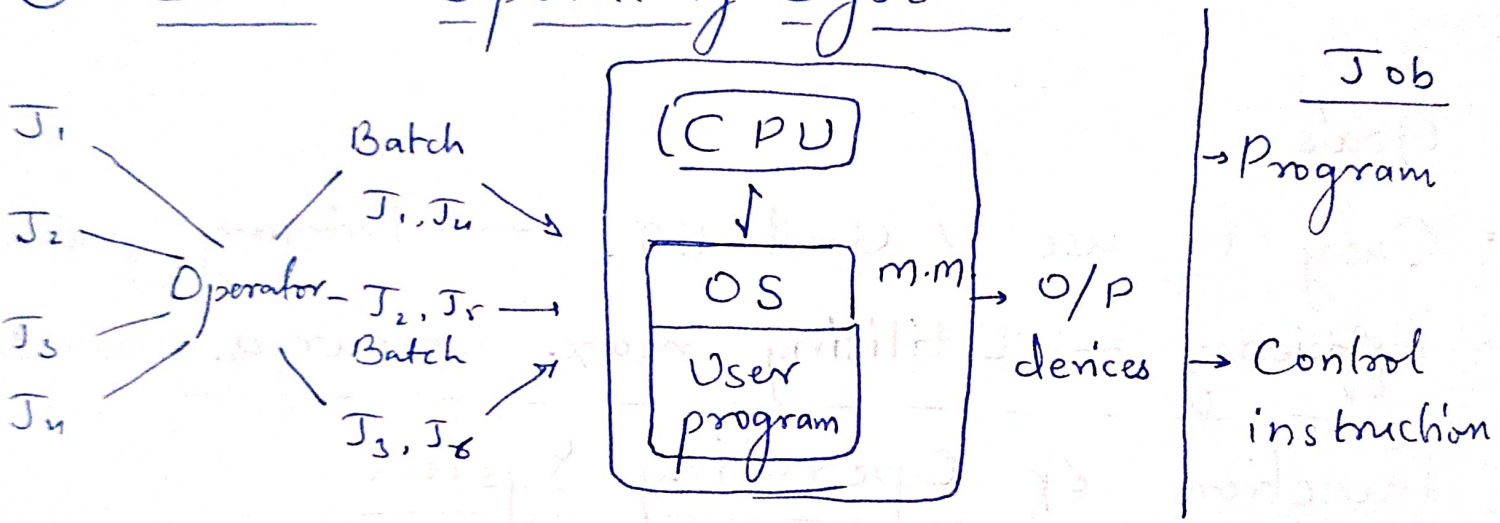
v) I/O Management

vi) Security & Protection

vii) Network Management

Evolution of Operating System.

① Batch Operating System



② Multiprograming OS

Utilization of CPU when it becoms idle after I/O bound job. ~~It~~ We simultaneously store CPU bound job & then load it into CPU after saving its state.

③ Time sharing O.S. (Multitasking OS)

All the multiple jobs are stored & given specific time limit to complete its task, if time limit exceedce next task is executed & this job is paused. ~~I/O~~ Multiprogramming is still applied.

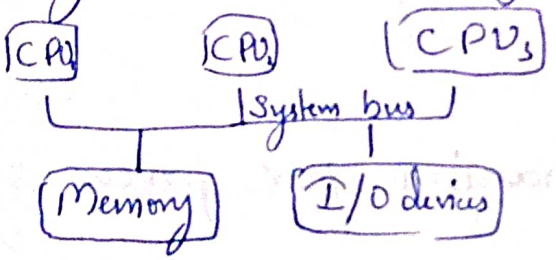
Throughput : No. of programs completed or executed per unit time.

Spooling : Arranging request for I/O or memory resources, then executing it one by one [$P_1, P_2 \ldots$]

~~Tight~~

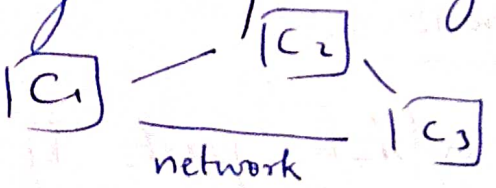
④ Multi CPU System (Multiprocessing O.S.)

i) Tightly Coupled System.



Access to I/O & memory resources is shared, it reduces time to access ridsour

ii) Loosely Coupled System



Spooling is decreased

Advantages of Multi CPU system

↳ Increased throughput
↳ Increased Reliability
↳ Cost efficient (Tightly is cheaper than loo

⑤ Network & Distributed OS

Network OS - Loosely Coupled System

Distributed OS - When a big job is given to Network OS & they are free to distribute the job

⑥ Clustered OS

When distributed OS only shares data (no I/O) It may be a network of long distance system.

⑦ Real time OS

System Calls

① Process Management

- Creation & deletion of user & system process
  → createprocess()
- Suspend & resume of process
  → wait()
- It provide mechanism for process synchronisation
  → shared memory()

② Memory Management

- Keeping track of which parts of memory (main) are currently being used & by whom.
- Deciding which process are to be loaded into memory when memory space becomes available
- Allocation & deallocation of memory spaces.

③ File Management

- Creation & deletion of directories.

- Supporting primitives for manipulating files & data
- Mapping files into secondary storage.
- Backing of files in stable storage media.

④ I/O device Management

- Spooling - Storing I/O in disk till its completion in swap space (Multiprog.. OS)
- A memory management component that include buffering, caching, spooling.
- A general devices driver interface.
- ~~Divers~~ Drivers for specific hardware devices.

⑤ Secondary Storage Management.

- Free space management
- Disk scheduling.
- Storage Allocation

⑥ Networking

⑦ Protection & security.

- Authentication - . . .
- Authorization - Giving or removing permission to user to access certain resources.

⑧ Command Line Interface.

↳ Graphical User Interface (GUI).

Operating System Services

◆ User services: Authorisation given to user & program to increase convinance, etc.

↳ Program execution.

↳ I/O access.

↳ File system management

↳ Communication

↳ Shared memory format – two or more program share memory & ~~the~~ communicate / transfer data through that space.

↳ Message passing – A packet is made with address of sender/reciver & data to comm

↳ Error detection.

◆ System Services:

↳ Resources Allocation.

↳ Protection & Security.

↳ Accounting.

◆ System Calls

↳ Interface bet" user & operating system.

↳ Assembly language was used initially

① Process Control
- Load, execute.
- abort, end
- Create, delete.
- get / set process attribute.
- wait event, signal event.

② File management

③ Device management.
- Read, write, reposition
- Request, release device.
- Information maintainance
- get/set system data.

⑤ Communication
- Open & closed connection.
- Create / Delete connection
- Send message & receive message.
- Transfer ~~user~~ status info.

UNIT - II Process Management

Program : Pasive entity which contains set of text section.

Program Counter : A register that contains the address of the instruction being executed at the current time & it stores next instruction as well.

Process : ..... processor register program counter, stack - local variable, func parameter, data data-section - global variable..

Attributes

1. Process ID → Initialisation etc..
2. Program counter
3. Process State
4. Pointers
5. CPU Registers → stores data when program I/O.. are not in cpu execution.
6. Memory ~~location~~ management -
7. I/O status information : list of I/O devices allocated to proc, list of open files
8. Accounting.

- Attributes are stored in PCB / TCB

Process Control Block.

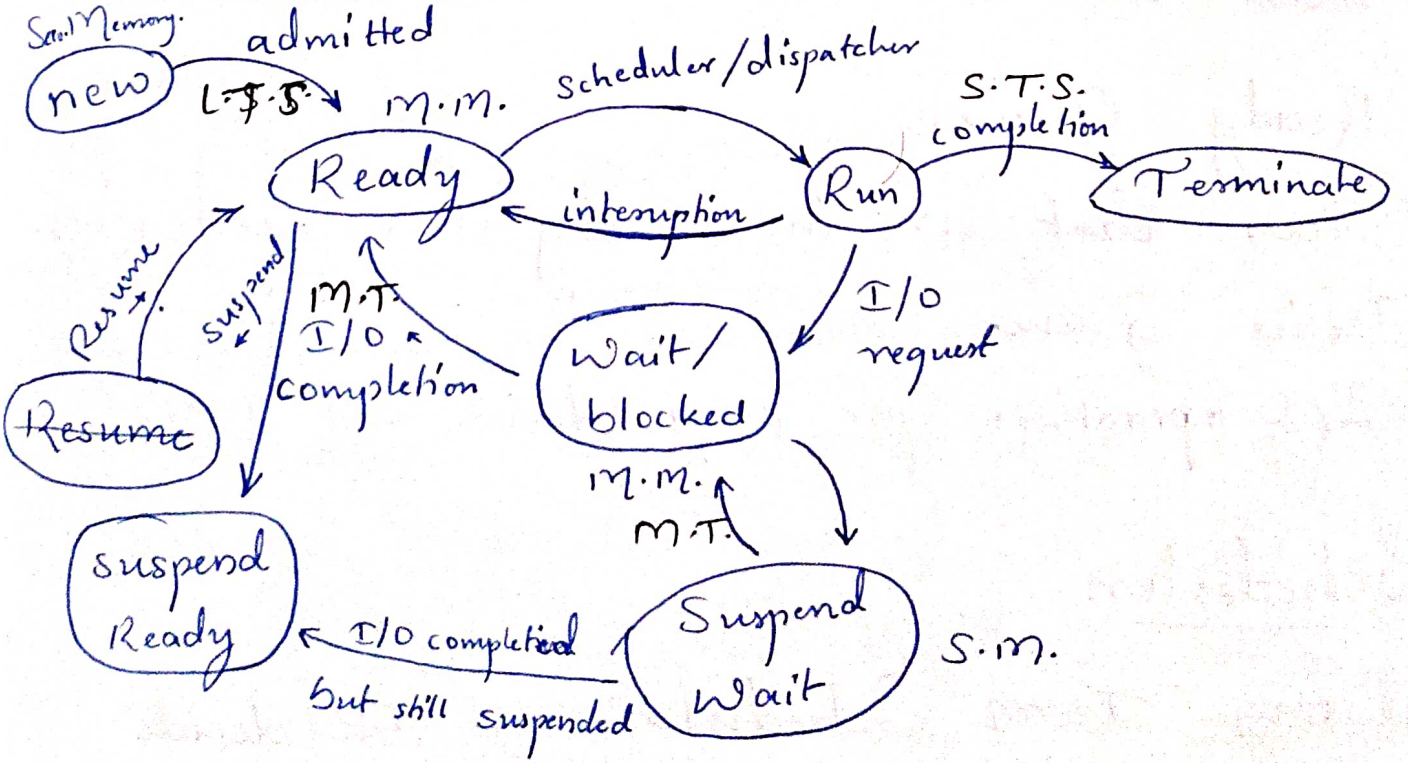
P.C.B / T.C.B. (Context of the process)

Process ID	Process State.
Program Counter	
CPU Registers	
Memory limits	
⋮	
List of Open files.	

Context Swithing.

When CPU ~~sti~~ switches from P_1 process to P_2 it stores data / context of P_1 in PCB_1 & loads data/ context of P_2 from PCB_2.

Process State Transition Diagram.



I/O operation can be done from Secondary Mem

When some important process that has to be performed.

1) To make space wait process for I/O operation is suspended.
2) Else Ready ~~wait~~ process is suspended
3) wait process perform I/O operation from sm it gets completed & st resume from suspend ready to ready state

Scheduling Queue

① Job Queue (Queue is implemented as linked list)

Process that are in new state. need J.Q.

② Ready Queue.

Process that are in ready state need R.Q.

③ Device Queue.

I/O operations for every devices need D.Q.

◈ Schedulers

① Long Term Scheduler - It decide degree of Multi-programming (no. of programmes in Memory)

② Short Term Scheduler - Decides which progr to load from ready to run state.

S.T.S. takes upto 10 ms of time to decide.

Dispatcher - loads & save PCB from Ready state. (Also switches User mode to Kernel mod vice versa)

If 100 ms is time quantum.

$\left(\frac{10}{100+10}\right) \times 100 \longrightarrow <10\%$ of time is extra requ

Operations on Process

① Process Creation.
② Process Termination.

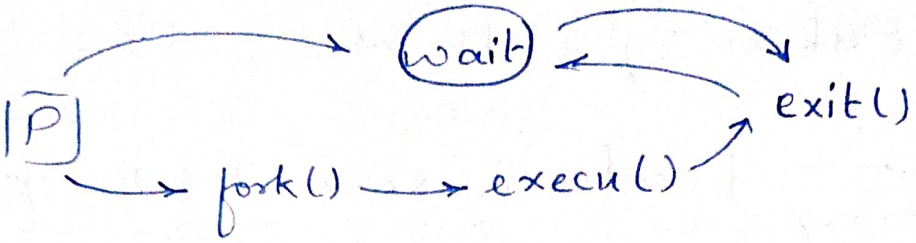
Process create multiple childs to complete task.

i) Address are different
In this data set is different of C & P.

ii) Address is duplicate.
a) Resource are shared. (queue system to
b) get access to resources)
b) C & P perform different tasks & don't share resources.

Parent process can't be terminated before all child process are not completed.

- Cascading - When parent process abort all the child start to terminated from last to parent



When process parent execute last statement & asks th system to delete it using 'exit()' by system ca

It returns its data to its parent

Process resources are dealocated.

Parent process may terminate the execution of child process using abort() system call.

- Abort system call :

i) Child has exceeded the usage of resource allocated.

ii) Task asigned to child is no longer requi

iii) The parent is exiting & the operating system doesn't allow child to continue its process.

Cascading Termination takes place

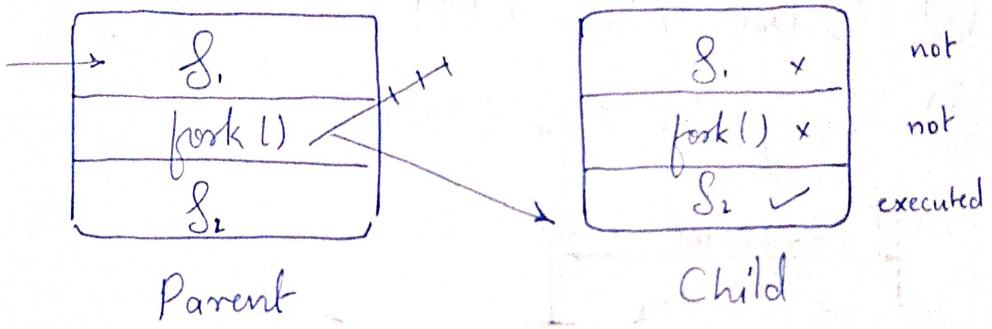
fork() system call

(1) Used to create new process.

(2) Not argument is passed in fork().

(3) After child exit(), fork() return +ve value to 'P' & '0' to 'C'.

→ If there is some problem
fork() return -ve value to both 'P' & 'C'.

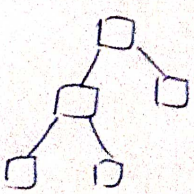


Q. main() {
fork(), printf("CSE")
fork()
printf("student")

→ 2 CSE
4 Student

→ If no conditional statement
n- no. of fork().
Total no. of process = 2^n
Total no. of c. process = $2^n - 1$.

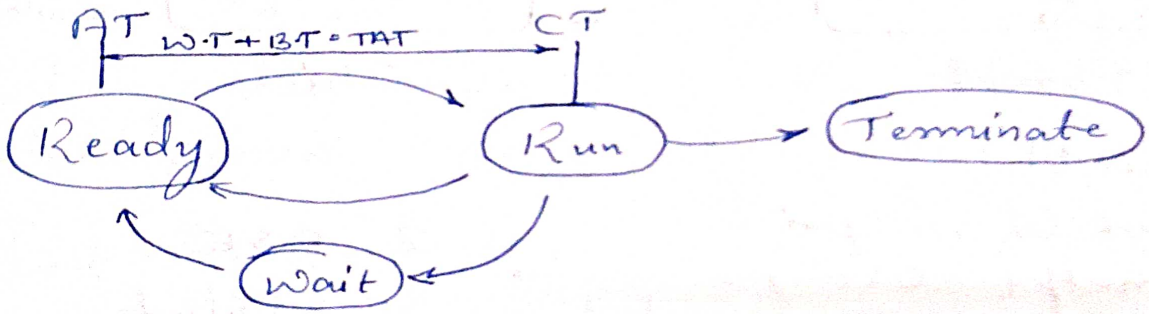
Q. main() {
if (fork() || fork())
fork()
print("Hello")



Q. main() {
if (fork() && fork())
fork()
printf("Hello")
}

CPU Scheduling Terminologies

1. Arrival Time (AT)
2. Burst Time (BT)
3. Completion Time (CT)



4. Turn Around Time (TAT) = W.T + B.T
   = C.T. − A.T. (or)
5. Waiting Time (WT) - Time spent by program in ready queue (at the start + waiting).
6. Response Time (R.T.) = first allocated to C.P.U. − A.T.

Criteria

① CPU utilization ↑ ③ Average TAT ↓
② Throughput ④ Average WT
⑤ Average RT

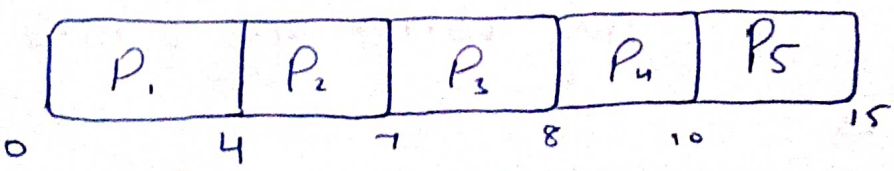
F.C.F.S. (First Come First Serve).

Criteria - Arrival Time.
Non-Preemptive - One scheduler.
Easy to implement.

CPU Scheduling Algorithm.

P. no.	A.T.	B.T.	C.T	TAT	WT	RT
P_1	0	4	4	4	0	0
P_2	1	3	7	6	3	3
P_3	2	1	8	6	5	5
P_4	3	2	10	7	5	5
P_5	4	5	15	11	6	6

→ GANTT CHARTS (Job Sequencing)

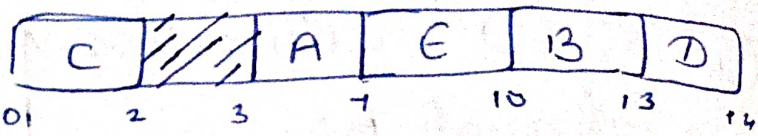


Average T.A.T. = $\frac{34}{5}$ = 6.8

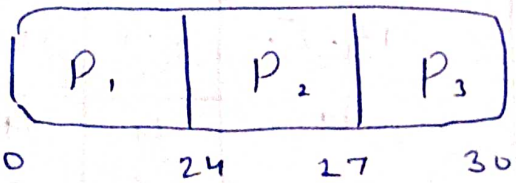
Average W.T. = 3.8

Avg. R.T. = 3.8

P. no.	A.T.	B.T.
A	3	4
B	5	3
C	0	2
D	5	1
E	4	3



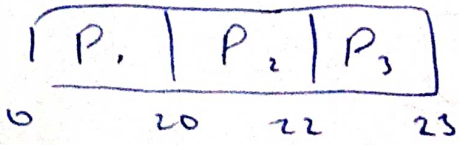
P no.	A.T.	B.T.	CT	TAT	W.T.	R.T.
P_1	0	24	24	24	0	0
P_2	0	3	27	27	24	24
P_3	0	3	30	30	27	27



Avg. W.T = 17

Avg. TAT = 27

P no.	A.T	B.T.	C.T.	TAT	W.T.	R.T.
P_1	0	20	20	20	0	
P_2	1	2	22	21	19	
P_3	1	1	23	22	21	



Avg W.T = $40/3 = 13.3$

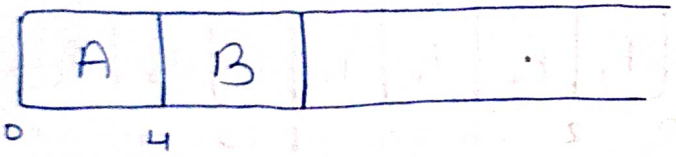
P no.	AT	B.T	C.T.	TAT	WT
P_1	1	20	23	22	2
P_2	0	2	2	2	0
P_3	0	1	3	3	2

- Convoy Effect : When a program with larger burst time increases the waiting time.

Shortest Job First (SJF).

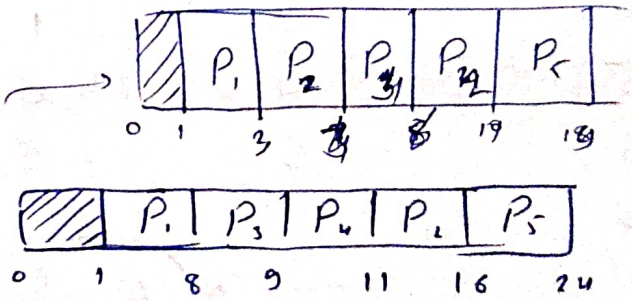
Q. [$\delta = 1$ (Context loading time)]

P no.	A.T.	B.T.
A	0	3
B	1	2
C	2	1
D	3	4
E	4	5
F	5	2



Q.

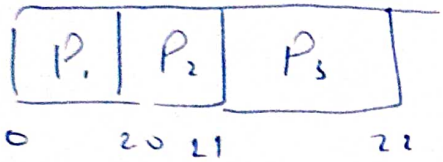
P no.	AT	BT
P_1	1	7/2
P_2	2	5
P_3	3	1
P_4	4	2
P_5	5	8



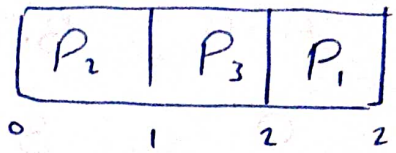
Q.

P no.	AT	B.T.	W.T.
P_1	0	20	0
P_2	1	1	20
P_3	2	1	21

P no.	AT	BT	W.T.
P_1	2	20	22
P_2	0	1	0
P_3	1	1	0



AWT = $\frac{20+21}{3}$

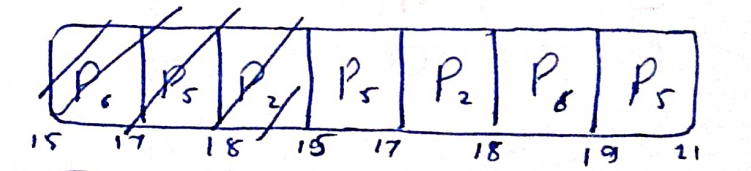


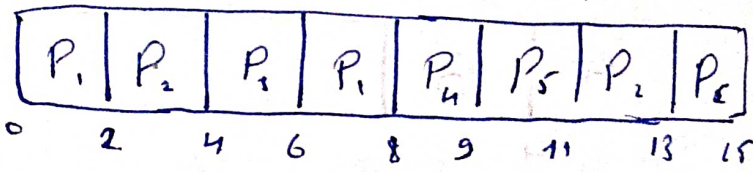
AWT = $\frac{22}{3}$

Round Robin Algorithm.

→ Preempline Algo.

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



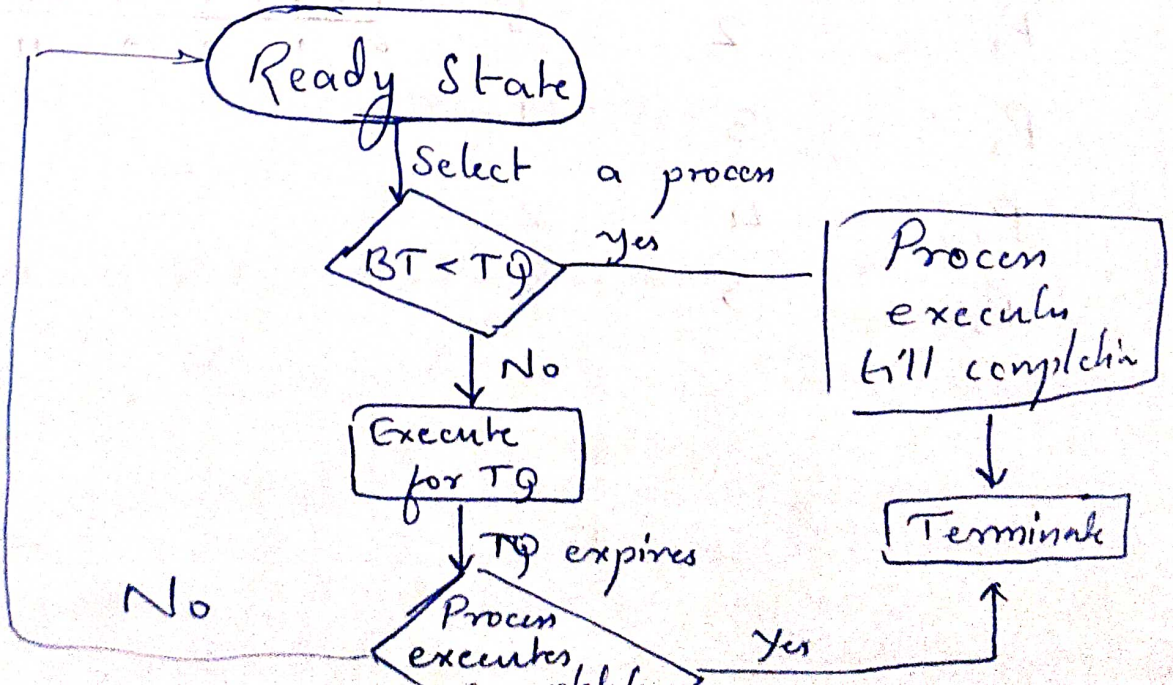


Ready Queue : $P_1\ P_2\ P_3\ P_1\ P_4\ P_5\ P_6\ P_5\ P_2\ P_6\ P_5$

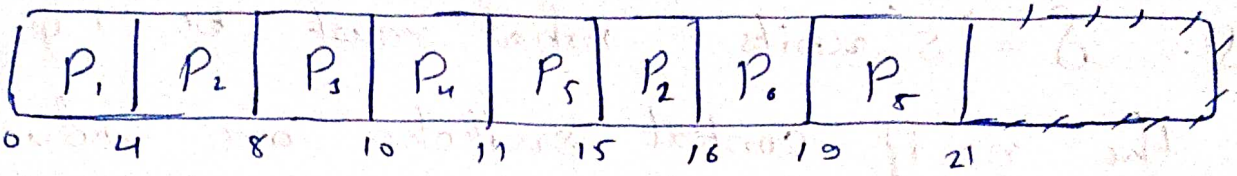
TQ = 2

W.T.

8
18
6
9
2
19



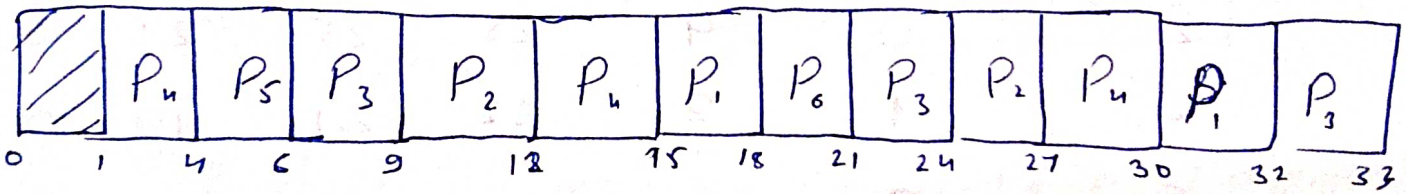
TQ = 4



Ready Queue : P_1 P_2 P_3 P_4 P_5 P_2 P_6 P_5

Q. TQ = 3.

Pno.	AT.	BT.
1	5	4
2	4	3
3	3	1
4	1	3
5	2	2
6	6	3



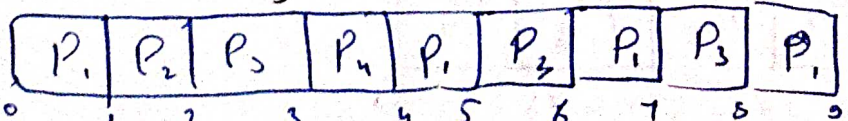
Ready Queue : P_4 P_5 P_3 P_2 P_4 P_1 P_6 P_3 P_2 P_4 P_1 P_3

Q.

	AT	BT
P_1	0	1 2 3 4
P_2	0	1
P_3	0	8
P_4	0	1

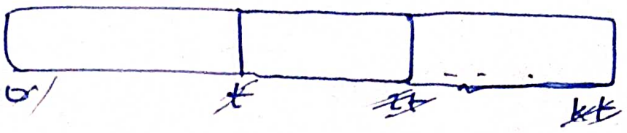
what is C.T. of P_1 when TQ = 1.

Ready Queue: P_1 2 3 4 1 3



C.T. for P_1 = 9

Q. δ = 's' units what must be T.Q. such that the no. of context switches are reduced but at the same time each process is garented to get the ~~tem~~ turn for every t seconds (n process)



$$\frac{kt + nsk}{\delta}$$

$$t \leq n \cdot s + (n-1)q$$

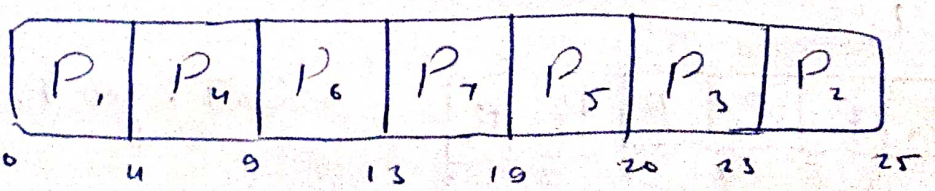
$$q \geq \frac{t - ns}{(n-1)}$$

◇ ~~Priority Queue~~

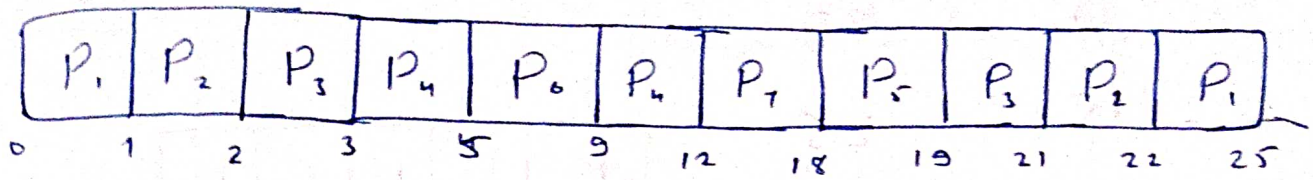
◇ Priority Scheduling.

P. no.	Priority	A.T.	B.T.	C.T.	TAT	W.T.	R.T
P_1	2	0	4	25	25	21	0
P_2	4	1	2	22	21	19	0
P_3	6	2	3	21	19	16	0
P_4	10	3	5	12	9	4	0
P_5	8	4	1	19	15	14	14
P_6	12	5	4	9	4	0	0
P_7	9	6	6	18	12	6	6

Non-preemptive

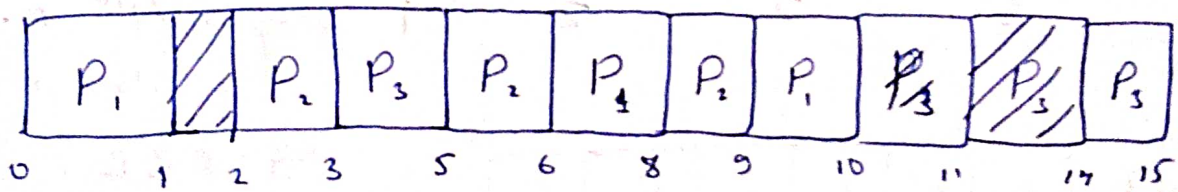


Preemptive



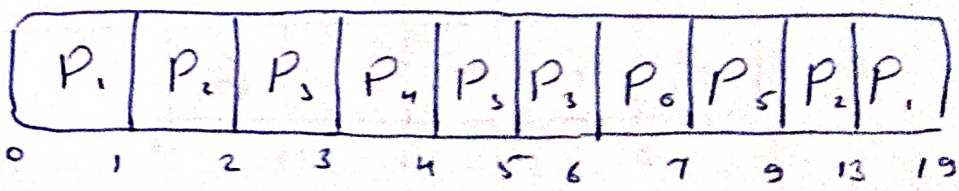
9. Preemptive Priority.

P no.	A.T.	Prio	CPU	I/O	CPU
P_1	0	2	1	8	3
P_2	2	3 (L)	3	3	1
P_3	3	1 (H)	2	3	1



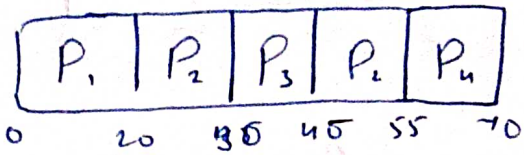
◇ Shortest Remaining Time First. (SRTF).

P no.	AT.	BT.
P_1	0	7
P_2	1	5
P_3	2	3
P_4	3	1
P_5	4	2
P_6	5	1



Q. P. no. A.T. B.T.

P_1	0	20
P_2	15	25 (15)
P_3	30	10
P_4	45	15



Q.

$P_{no.}$	AT	BT	I/O	BT
P_1	0	3	2	2
P_2	0	2	4	1
P_3	2	1	3	2
P_4	5	2	2	1

In this's case A.T. is absolute I/O don't affect it

Ready Queue prefer 1st A.T

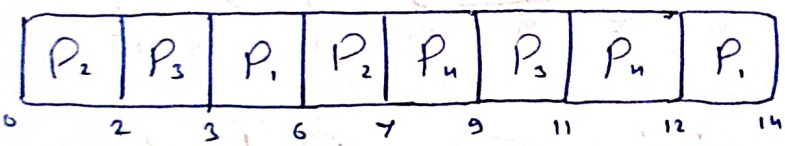
P_2-6, P_3-6, P_1-8

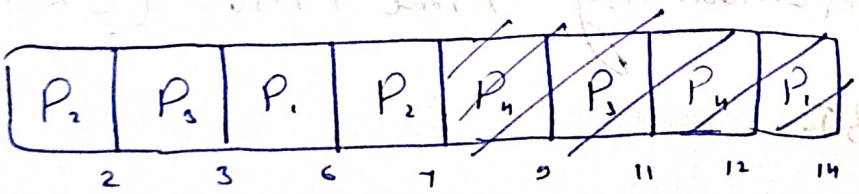
P_2-6, P_3-6, P_1-8

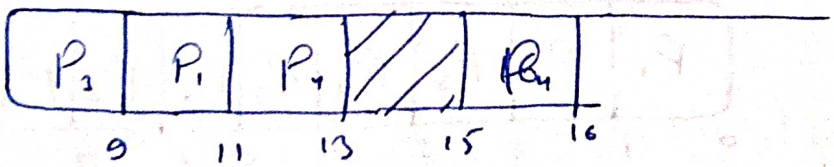
P_2, P_3, P_4 P_4-11

RQ – P_1 P_3 P_4

P_4-11



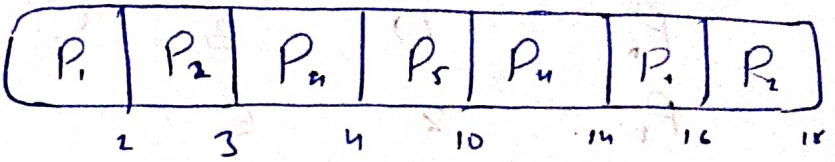




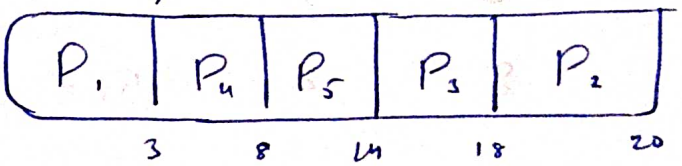
◇ Longest Job First (LJF)

Q.

P. no.	AT	BT
P_1	0	~~3~~ 2
P_2	1	2
P_3	2	~~4~~ 3
P_4	3	~~5~~ 4
P_5	4	6

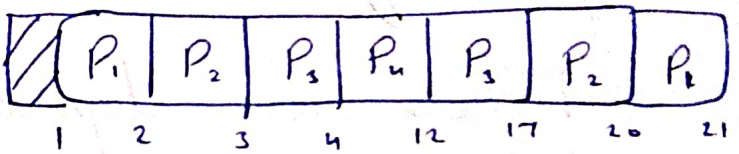


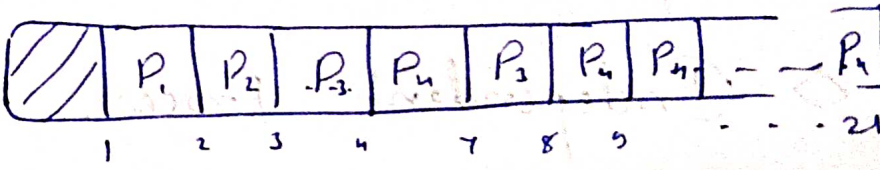
Preemptive.



Q. Longest Remaining Time First (LRTF).

P. no.	A.T.	B.T.
P_1	1	2
P_2	2	4
P_3	3	6
P_4	4	8





◇ Highest Response Ratio Next (HRRN)

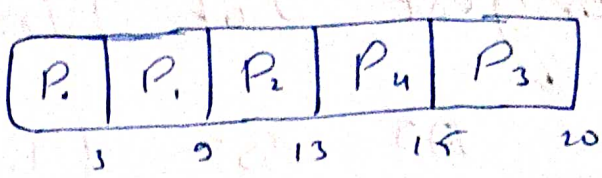
Criteria Ratio = $\frac{W.T. + B.T.}{B.T.}$ $\left\{\frac{W+S}{S}\right\}$

HRRN not only favour Shorter Job but also limits the W.T. of larger Job

(Nature : Non-Preembive)

Q.

P no.	A.T.	B.T.
P_0	0	3
P_1	2	6
P_2	4	4
P_3	6	5
P_4	8	2



$RR_1 = \frac{(9-4)+4}{5} = 2.25$, $RR_3 = \frac{(9-6)+5}{5}$ ...

$RR_3 = \frac{(13-6)+5}{5} = 2.4$ $RR_4 = \frac{(13-8)}{2}$

Process Syncronization

◇ Cooperating Processes.

Process are dependent on each other to complete a task.

◇ Independent Process.

Process that can't affect or be affected by other processes

Need for P.S.

① Modularity
② Information Speedup.
③ Computation speedup.
④ Convinence

Producer - Consumer Problem

Bounded buffer.

```
#define BUFFERSIZE = 10
typedef struct {
        ------
} item;
item buffer [BUFFERSIZE];
int in = 0;
int out = 0;
while (1) {
    while (((in+1) % BUFFERSIZE) == out)
        ; // do nothing
    buffer [in] = nextproduced;
    in = (in+1) mod buffersize;
}
while (1) {
    while (in == out)
        ; // do nothing.
    nextconsumerd = Buffer [out];
    out = (out+1) mod BUFFERSIZE;
}
```

counter = BUFFERSIZE

counter ++;

counter == 0

counter --;

Race Condition :- A situation where several process & manipulate the data & the outcome of the execution depends on perticular order in which the access takes place

Critical Section Problem.

```
do
{ entry cond^n
      critical section
  exit cond^n
      Remainder section.
} while (1);
```

Soln^n criteria for Critical Section Problem.

1. Mutual Exclusion.
2. Program
3. Bounded Waiting.