Operating Systems

... n... functionality of system

Date 28/4/22
Page

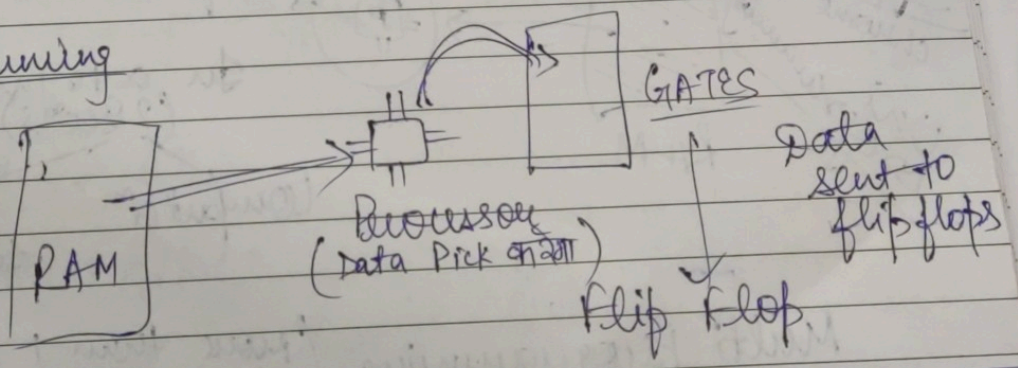
website -
tiny.cc/os2022
)) /os2022/abdt

```
#include <iostream>
using namespace std;
int main(){
int a, b, c;
c = a+b;
cin >> a;
cin >> b;
c = a+b;
cout << c;
return 0;
}
```

O.S :-

- Embedded Sys. - when u have some code & embedded that code in a chip.

Uniprogramming



29/4/22

sum.c

compile → compiler (System Software)

sum.obj (object code file) - Assembly code written in language (Mnemonics)

Sys. Soft. ka a — Assembler (act on assembly lang. during exec" of program)

sum.exe

Machine code (like 01100)

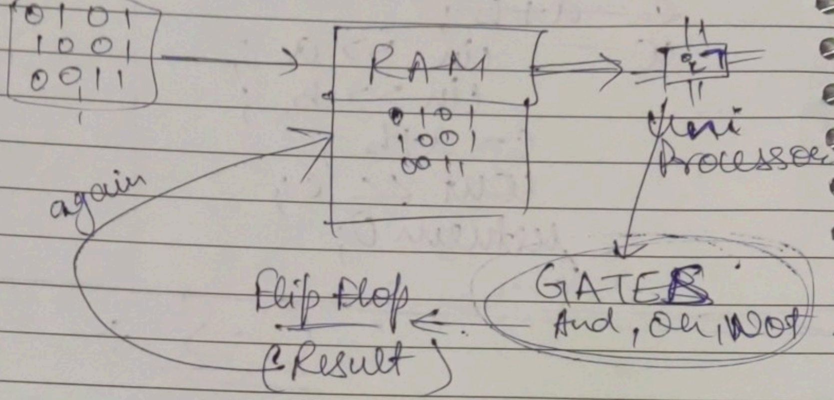
Applica" Soft -

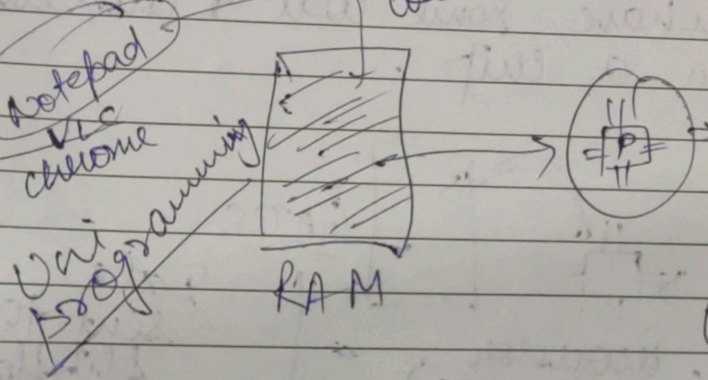
Sys. Soft.

Notepad → Notepad.exe

के icon पे double click करा then Notepad.exe का code

Notepad.c
↓
" .obj
↓
" .exe



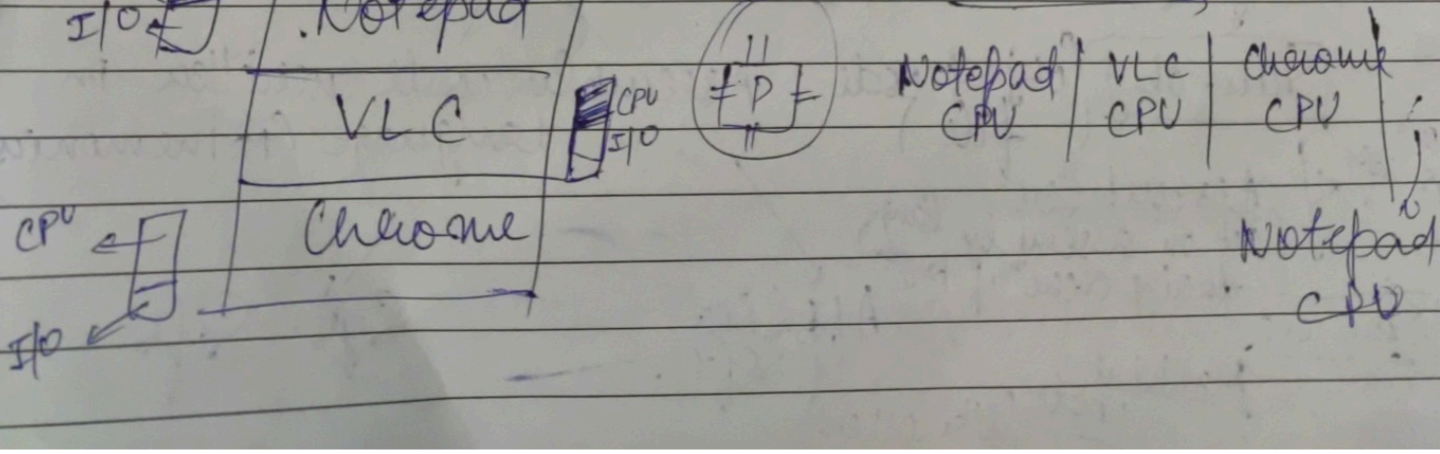


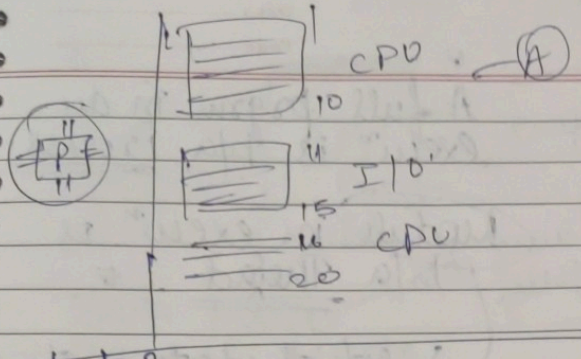
Performs Computaⁿ related tasks

In a Prgm (2 secs)

Computaⁿ | I/O (Input | Output)

Multi Programming (More than 1 prgrm can be resided into RAM at a single time)

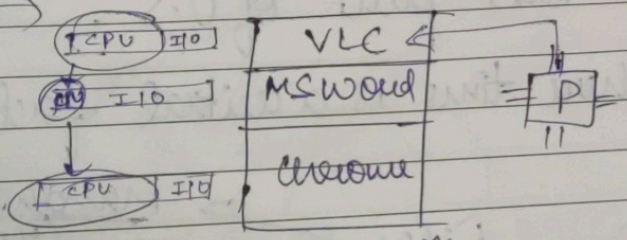




Multi Tasking
(switching of CPU progms over multiple progms)

Multi Threading - Tasking: (Time Sharing / Round Robin)

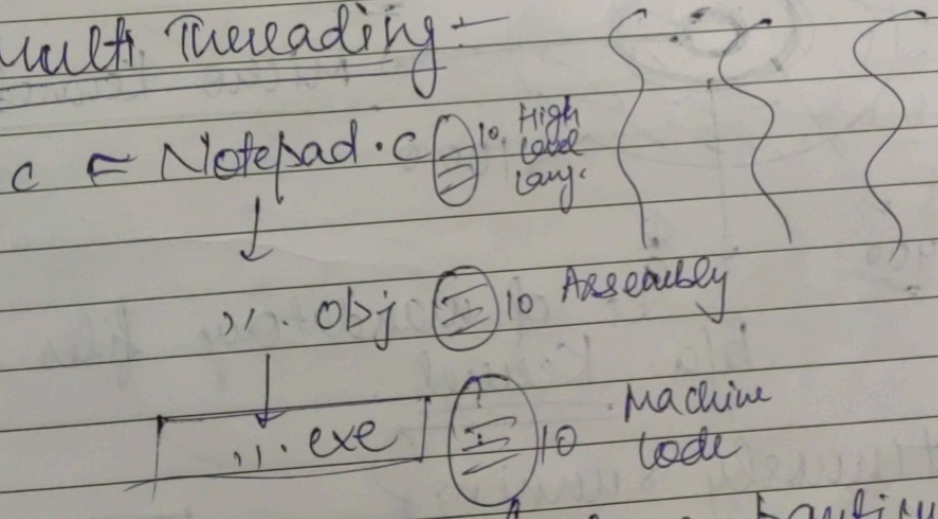
- Illusion of executing processes in HDL is known as concurrency & the P's involved in concurrency are called concurrent processes.



Serial Execn.

* Multi tasking is an Extension of Multi Programming

→ Multi Threading:-

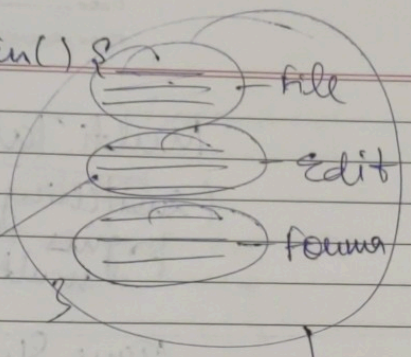


* Loading On demand (only a particular code is executed in exam)
(If we have diff. modules in progrm)

FREEMIND

Date

Page



A full prgm in an exeⁿ is b/a Process

Module in exeⁿ is b/a Thread.

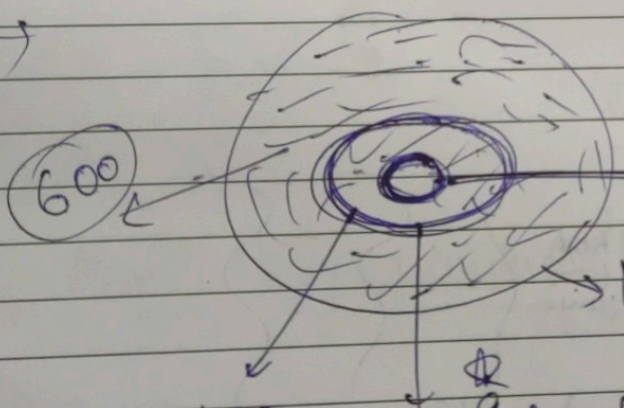
→ set of statements

4/8/22

Multi Threading : more than 1 modules in exeⁿ.

→ Kernel: it is a sub-part of O.S.

Real Time System – where time is a critical factor



Files (1000) → Mandatory (400)

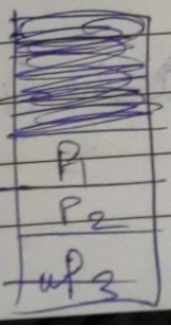
→ desirable

→ Micro Kernel

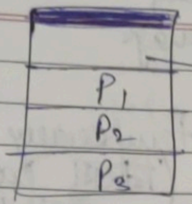
Set of mandatory files is b/a Kernel.

OS → continuously running

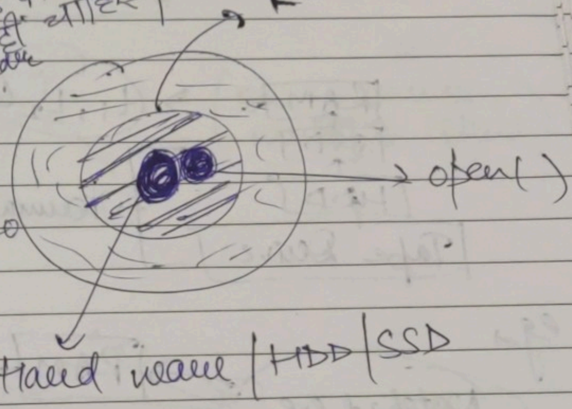
OS | Kernel



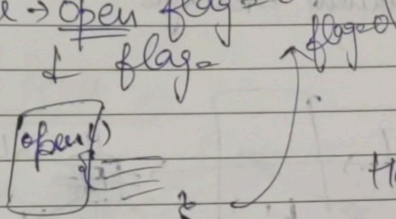
Micro Kernel approach

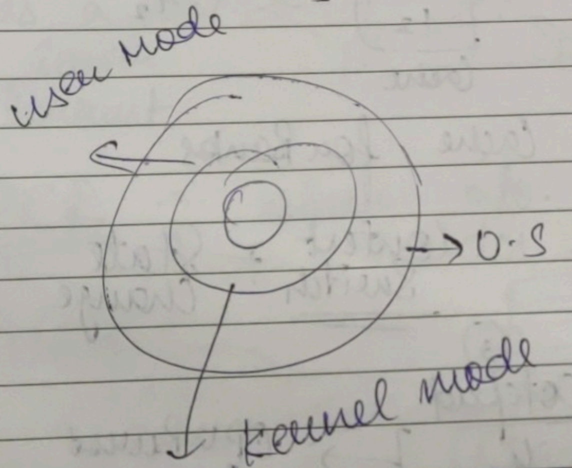


User Mode → जब kernels की files/folder नहीं चाहिए | Hardware से कुछ नहीं चाहिए। Kernel
Kernel Mode

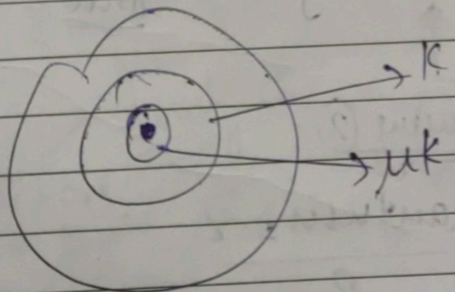


User Mode → open flag = 0
& flag =





5|8|22



Disadvantage →

If we & size of Kernel, all functionalities will come in User mode.
~~Vuln~~ Vulnerability ↑es

→ Cache Memory :-

P.T.O

Date
Page

Memory → RAM / Temporary Storage

Cache → Buffer → a small temporary storage area. (जिसमें Data आएगा, use होगा then wipe out)

iii) RAM → L1, L2 Cache → Registers

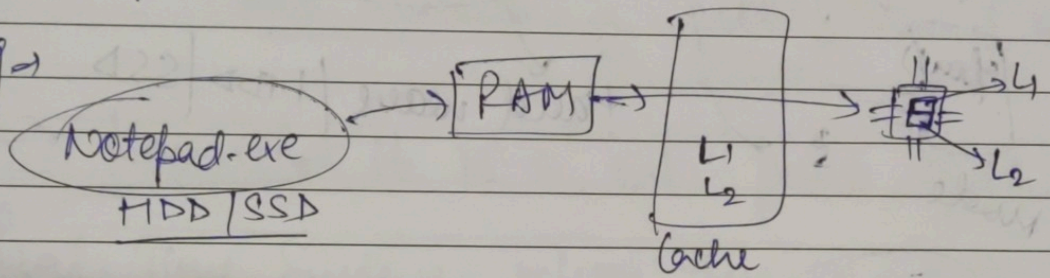
SSD ↑ RAM

HDD ↑ SSD

Tape Drive

(HDD, Tape Drive) → Permanent Storages

eg →



eg → ATM act as cache for Banks.

→ Context → State Context Switch → State Change

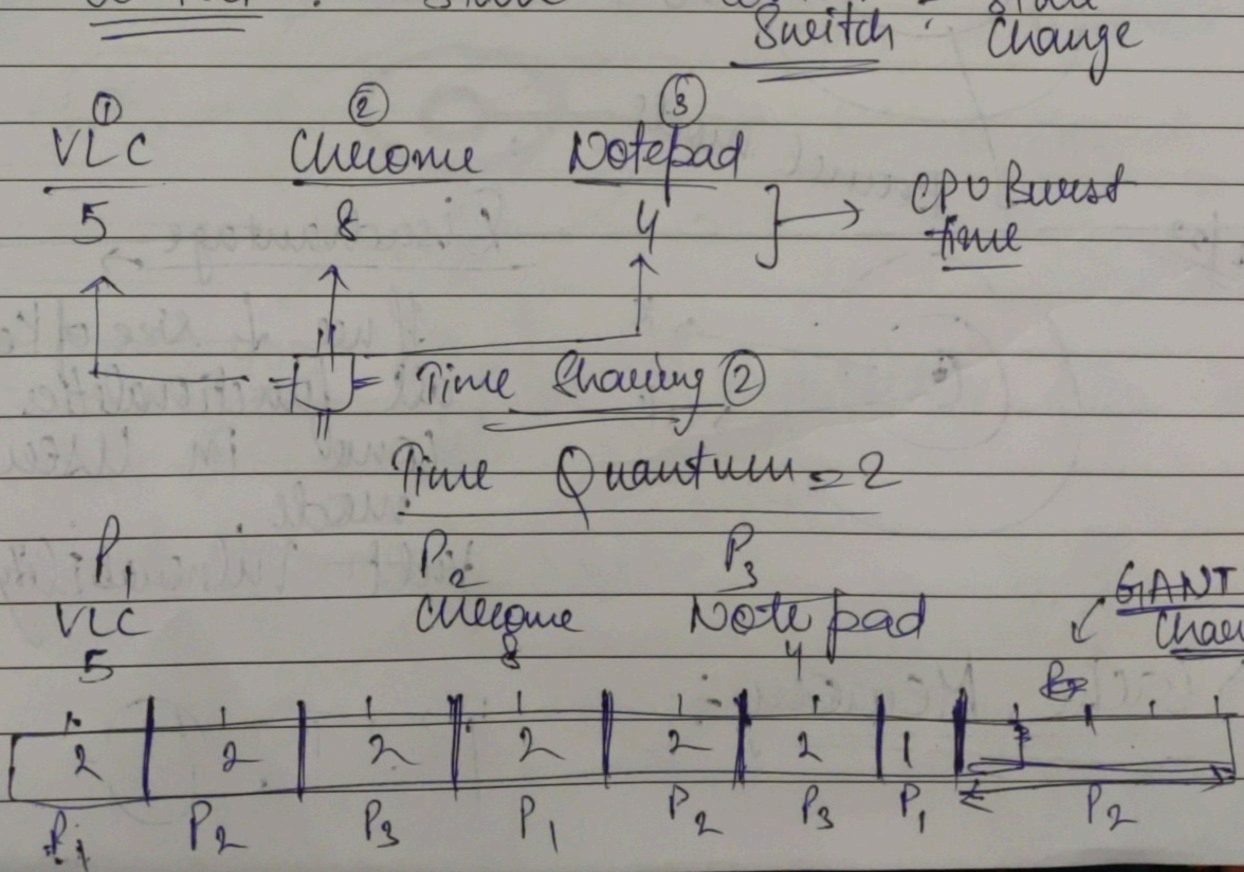
① VLC	② Chrome	③ Notepad	
5	8	4	} → CPU Burst time

Time Sharing ②

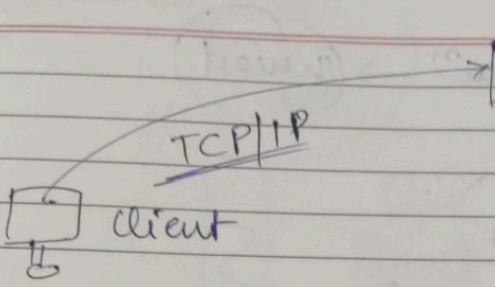
Time Quantum = 2

P_1	P_2	P_3
VLC	Chrome	Notepad
5	8	4

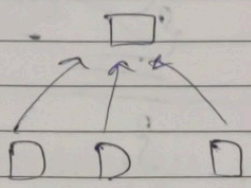
← GANTT Chart



Date 8/8/22
Page

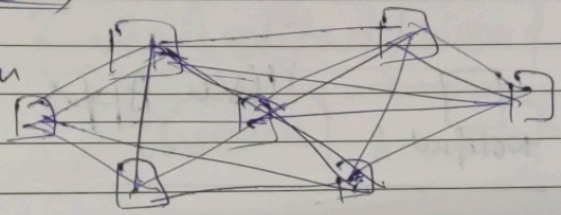


Client - Seerver → (one seerver, multiple client)



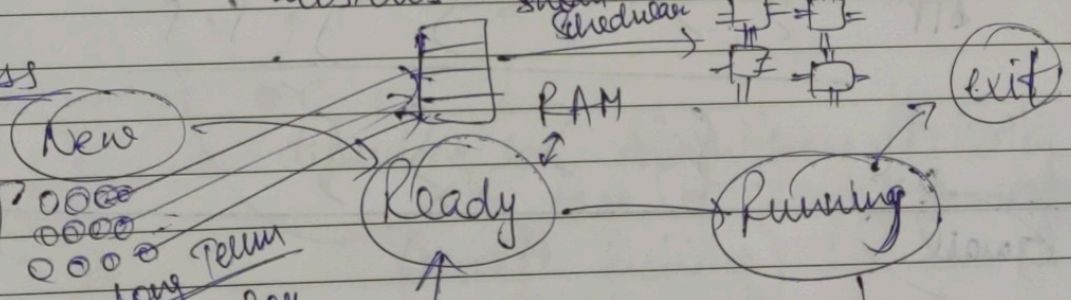
Peer to Peer →

any machine can act as a client or a seerver.
eg→ Torrent



→ PCB → where info. regarding process resides.

Process



FREEMIND

Date

Page

iii)

Module in an execⁿ → Thread

↓

Method

Function

→ main()

{ → { Thread

}

So, minimum no. of threads = 1

[:] Mapping

Notepad

User Apps

O.S

Apps

O.S

Gmail

Write

Spell check

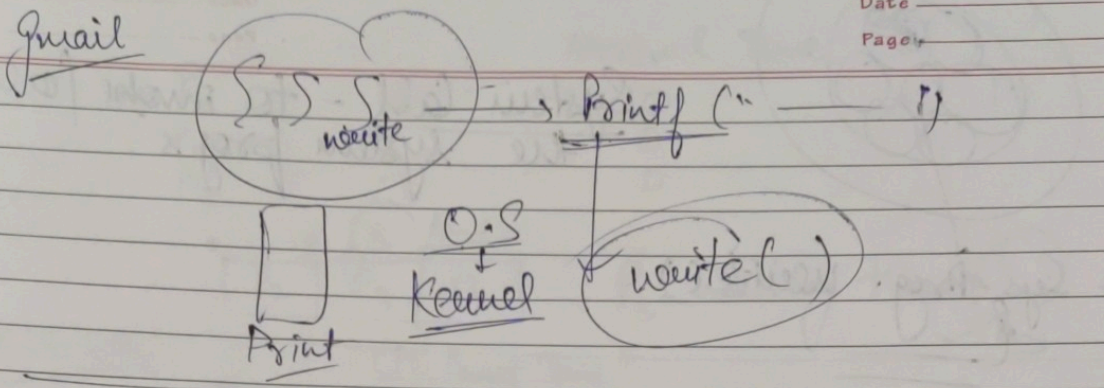
Attachment

FREEMIND

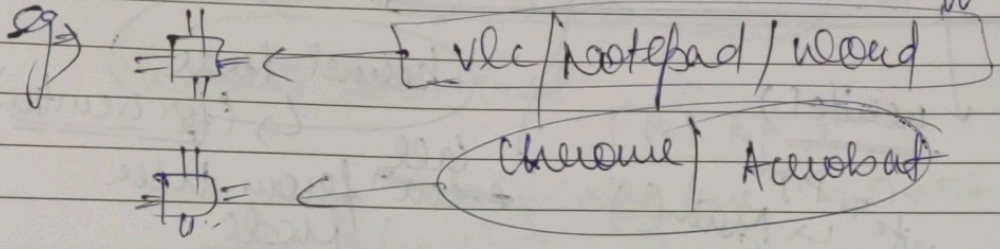
Date

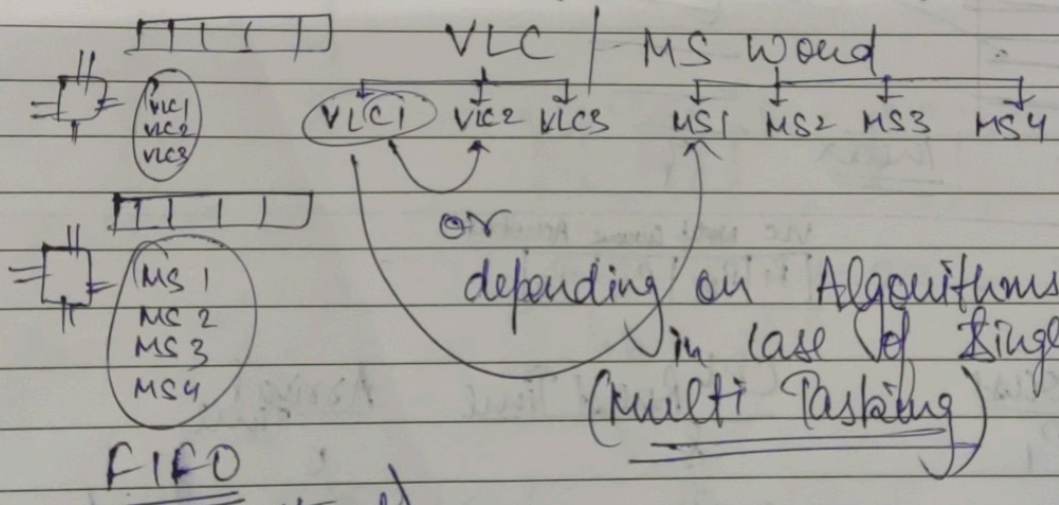
Page

Gmail



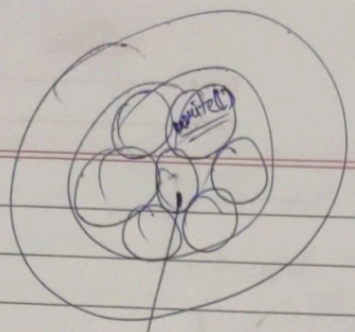
In case of multi-processors, load sharing will take place until affinity.





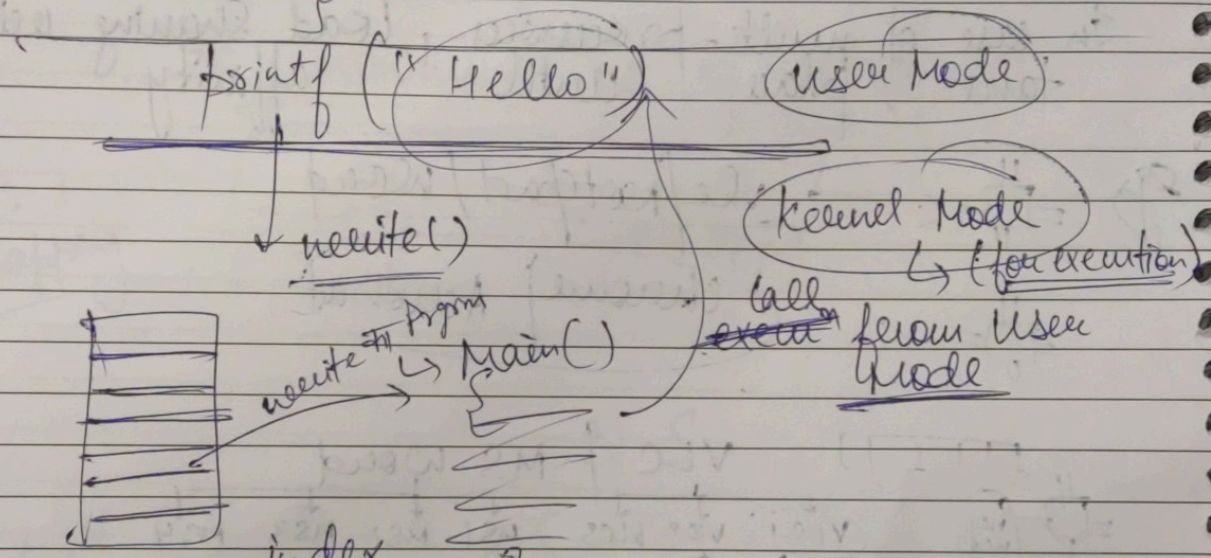
→ Sys. Prog. & Sys. Call.

↳ Sys. की functionality के लिए लिखा गया Program
is Sys. Prg.



System call - to invoke/call the system prog.s.

iii) eg- Sys. Prog. write()



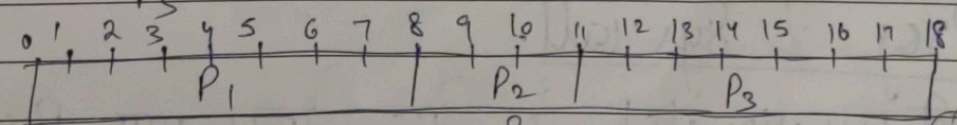
9 | 8 | 22

VLC Notep Chrome Acrobat

P_1	P_2	P_3	P_4

t=0

Process	CPU Burst Time	Arrival Time
P_1	8	0
P_2	3	0
P_3	7	0



GANTT CHART

Waiting Time = 0 + 8 + 11 = 19

3	7	18

Waiting time = 0 + 3 + 10 = 13

FREEMIND

Date

Page

		Arrival Time
P_1	18	10
P_2	3	0
P_3	7	0

0 3 (3) Ans

P_2	P_3	P_1

P_3 - waiting time = 3 Ans

18 | 8 | 22

	CPU Burst Time
P_1	8
P_2	5
P_3	7

First Come First Serve | P_1 | P_2 | P_3 |

Shortest Job First

5

P_2	P_3	P_1

Total waiting time = 0+5+12 = 17

	CPU Burst	arrival time = 0
P_1	9	
P_2	7	
P_3	2	
P_4	5	

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 23

P_3	P_4	P_2	P_1

0 + 2 + 9 + 23 = 34

14 − 23

25 − 34

	wait time
P_1	14
P_2	7
P_3	0
P_4	2

FREEMIND

Date

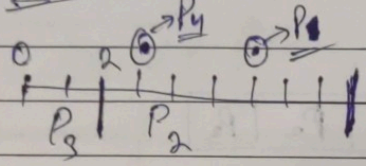
Page

→ Pre-emptive → चलते को रोक देना + 2nd चलाना

Non-Preemptive - जो चल गया वो चल गया (Till Completed / Stop)

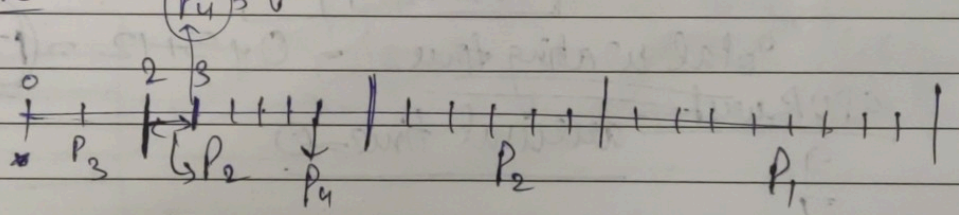
Arrival time		CPU Burst
5	P_1	9
2	P_2	7
0	P_3	2
5	P_4	5

Shortest Job first



In case of Preemptive, P_2 will go in Ready State

Preemptive -

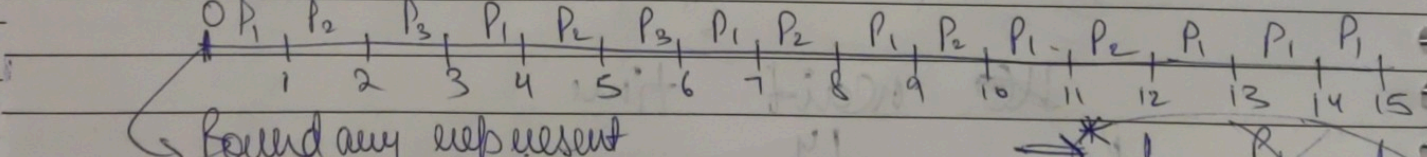


22/8/22 → Round Robin :-

	CPU
P_1	8
P_2	5
P_3	2

T.Q = 1 unit (Time Quantum)

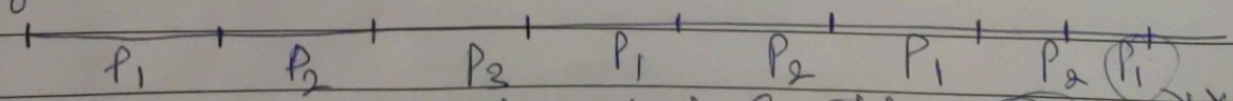
* In Shortest Job, T.Q is priority
> It is not SJF



Round any represent context Switching

* 1 P_1 1 wrong Represent a"

If T.Q = 2 Unit



No. of context Switch = (7) → X

FREEMIND

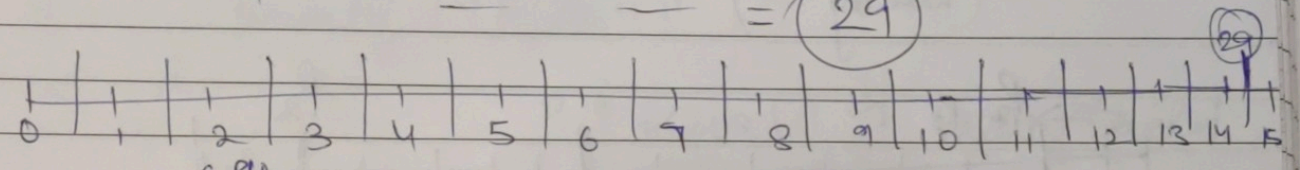
Date

Page

P_1 – already खत्म हो गया (no need to ~~save~~)
so no need to count (RAM में कुछ save क्या हो नहीं)

In T.Q = 1 unit in above example,
no. of context Switch = (14)

In T.Q = 0.5 — — — — —
— — = (29)

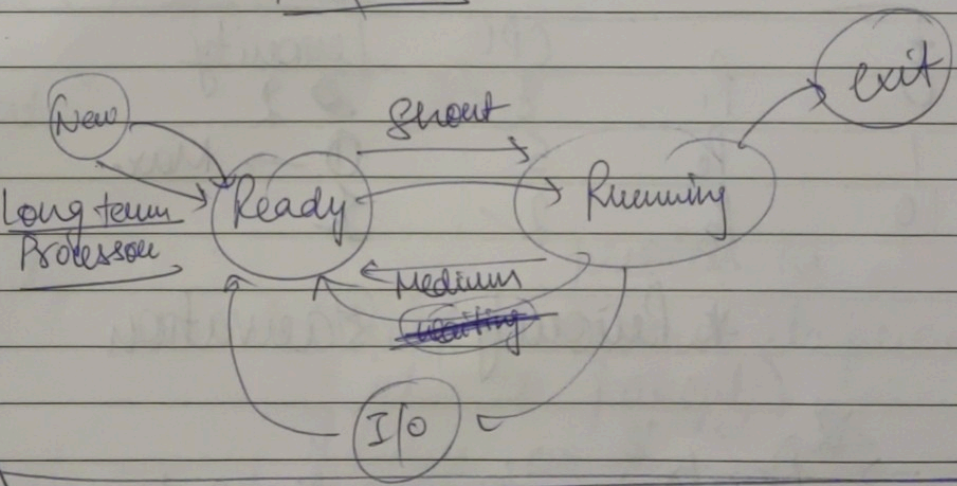


	CPU
P_1	7
P_2	5
P_3	2

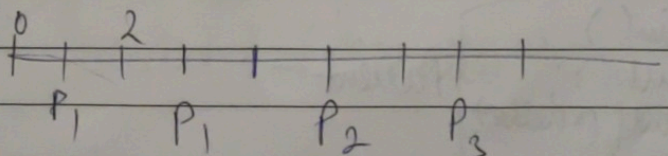
→ what will be T.Q to convert this to I come I serve
= (7) Ans

t=		CPU
0	P_1	8
2	P_2	5
3	P_3	2

T.Q = 2



23 8 22



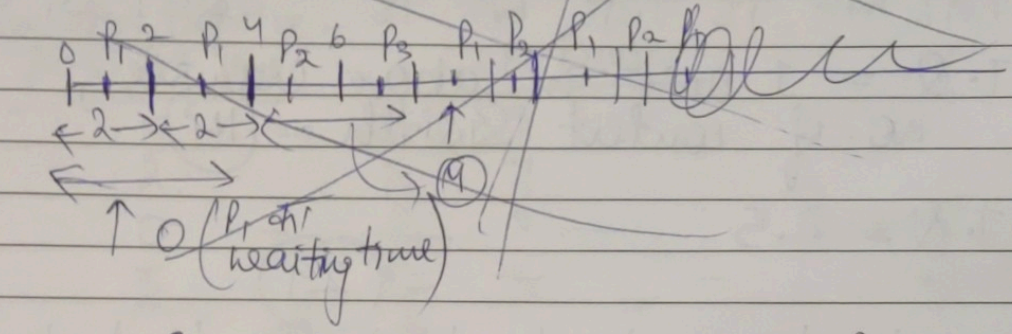
FREEMIND

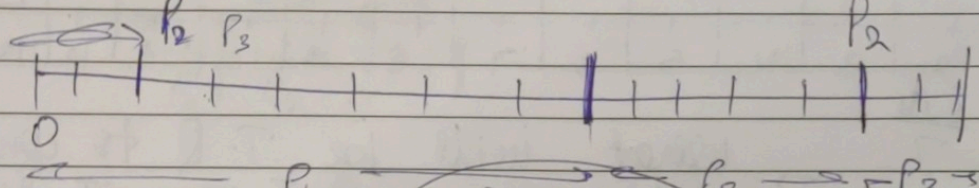
Date

Page

t		
0	P_1	8
2	P_2	5
3	P_3	2

T.Q = 2





$P_1 - 0$

$P_2 - 6$

$P_3 - 10$

FCFS

$\frac{16}{3} \rightarrow 5.3$

t		CPU	Priority
0	P_1	8	2
1	P_2	5	0 → Max.
10	P_3	2	1

* Priority, Starvation

→ fork : to create a process in Ubuntu

Sum.exe

```
main()
{ fork();
  printf("hello");
}
```

Parent

child

Date 25/8/22
Page

→ fork() command :-

= Parent खुद भी चलेगा और child को initiate भी कर देता है।

```
main()
{
  fork();
  printf("Hello")
}
```

Parent

* (fork() की value child में always '0')

printf("Hello")

child (duplicate)

* (executed from line after fork())

⇒ fork() splits

Parent → process id

child → process id

* (Child returns its process id to parent)

eg → void main()

if (true or *) (no need)

if (fork() || fork()) → always true

printf("Hello")

executes positive value (process id)

f() || f() child → 0

f() || f() → 0

पहले से 0

child

0 || 0

false

FREEMIND

Date

Page

Output –

2 times print Hello

eg →

(true && ___ needed)

```
void main()
{
    if(f() && f())
    {
        Hello;
    }
}
```

+ve value

+ve value

f()&&f() child

f()&&f() child

0&& —

X

X

output

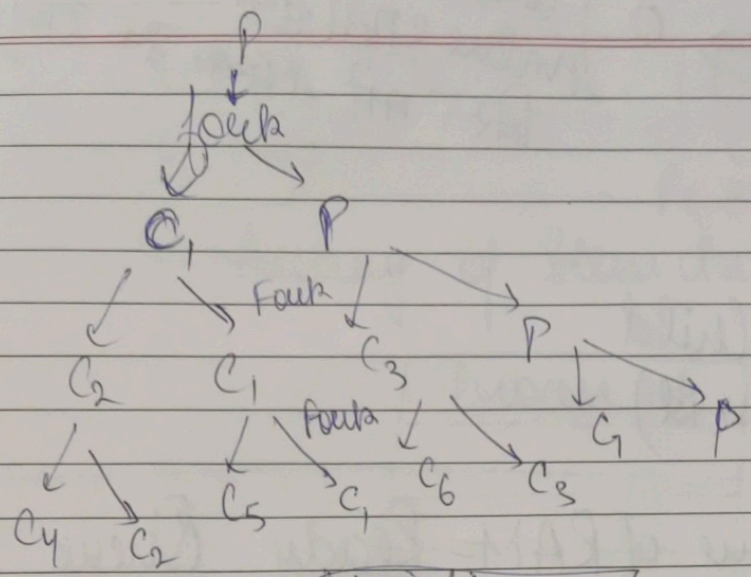
one time Hello

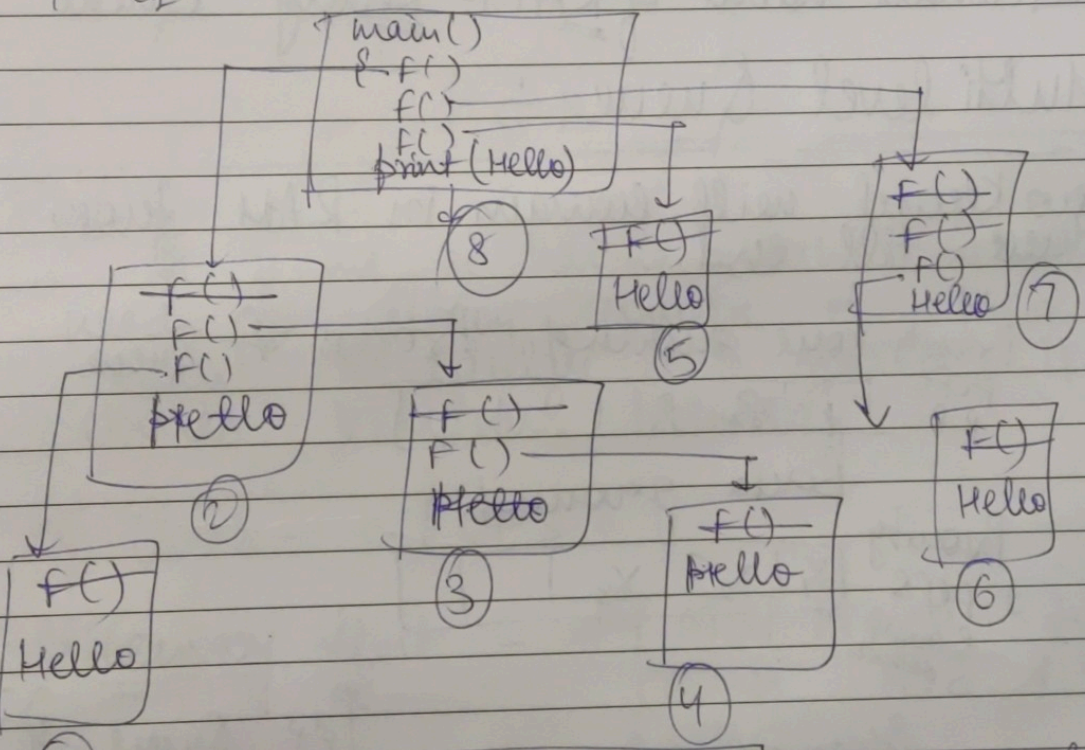
⇒ getpid(), getppid()

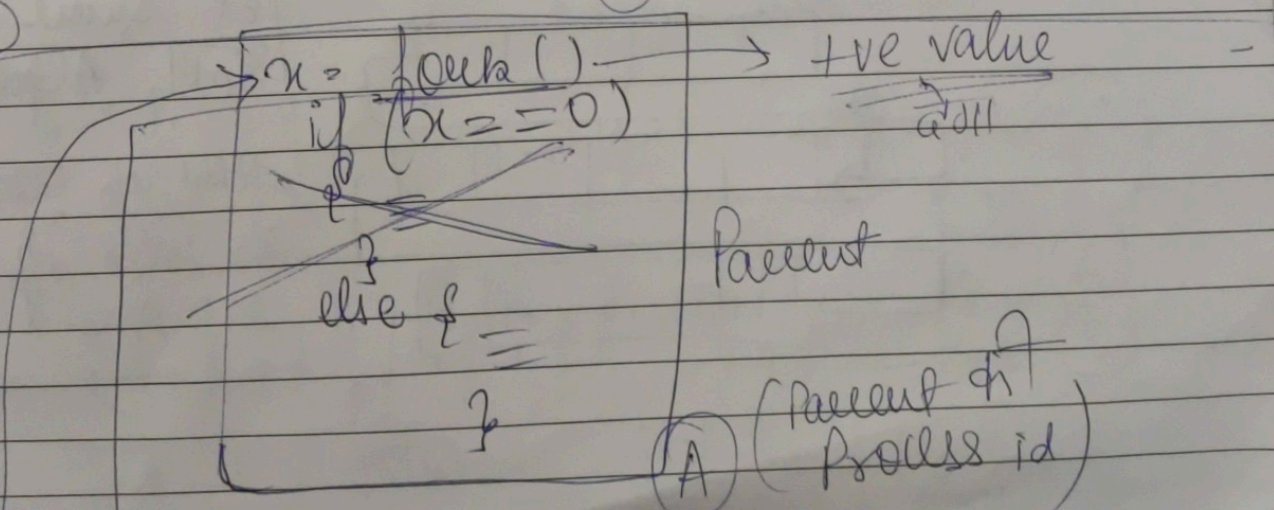
(to get process id)

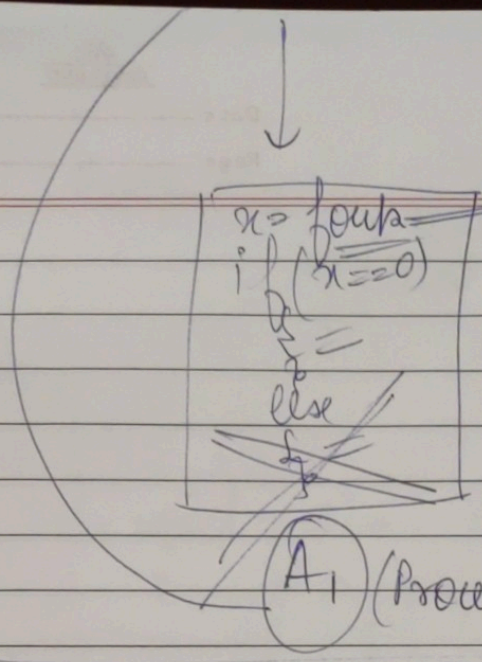
26/8/22

```
main()
{
    fork();
    fork();
    fork();
print(Hello)
}
```


P
fork
C1
P
fork
C2
C1
C3
P
fork
C4
C2
C5
C1
C6
C3
C7
P


main()
{ f()
f()
f()
print (Hello)
8
f()
f()
f()
Hello
2
f()
Hello
1
f()
f()
Hello
3
f()
Hello
5
f()
f()
f()
Hello
7
f()
Hello
6
f()
Hello
4


x = fork()
if (x == 0)
{
}
else {
}
+ve value
Parent
(Parent of Process id)
A



0 [child ki id value आती है fork से return होने पर]

* Alternate name of RAM = Ready Queue

☆ Multi level Queue :-

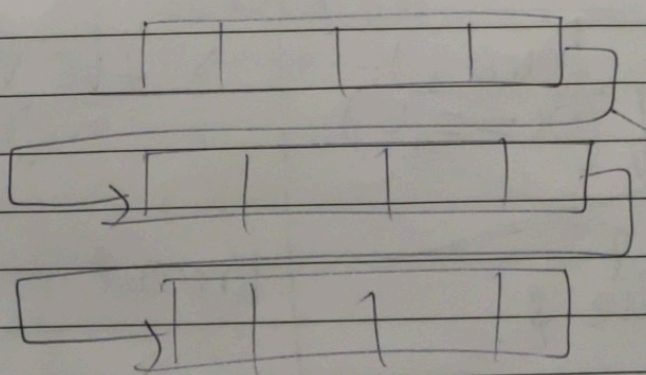
→ Micro kernel will remain in RAM from start till end.

Priority Fore ground Process की Queue

FCFS RR | P1 | P2 | P3 | | |

Back ground

Priority FCFS Round Robin | X1 | X2 | X3 | | |



हर Queue में अलग Algorithm

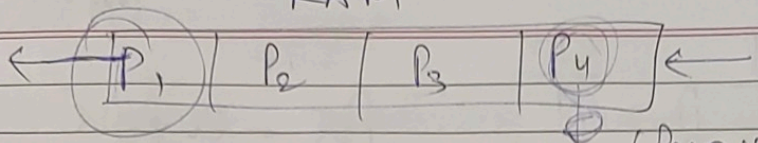
इसका o/p दूसरे के लिए i/p

FREEMIND

Date

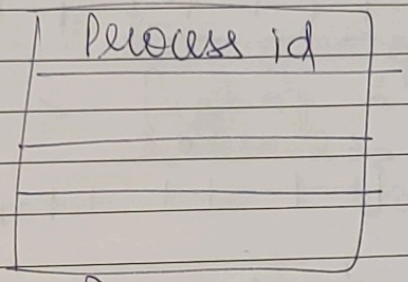
Page

RAM



PCB (Process Control Block)

Array of Structures (Node of linked list)



PCB

29/8/22

Q. If we r having flash memory there is need to deploy concept of Med- & long term scheduler || more concept comes - Virtual Memory

Q Flash - 1. FCFS

→ Convoy effect - जब बड़ा Process पहले, छोटा Process पीछे | ⇒ less Response time.

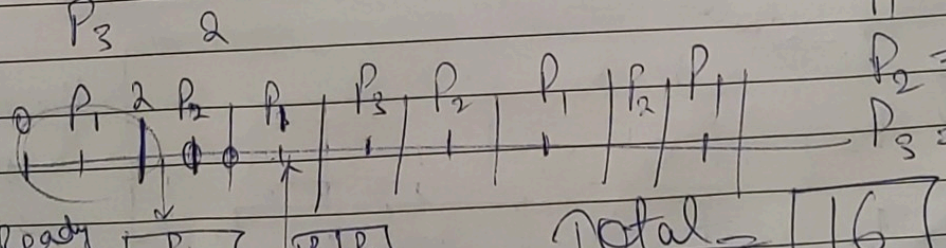
t		
0	P_1	8
2	P_2	5
3	P_3	2

2 + 4 + 1 = 7

$P_1 = 7$

$P_2 = 6$

$P_3 = 3$



Ready queue: P_1 | P_1 P_2

Total waiting time = 16

FREEMIND

Date

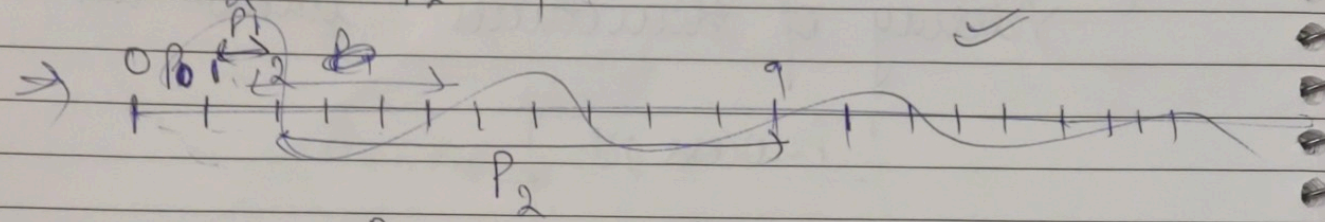
Page

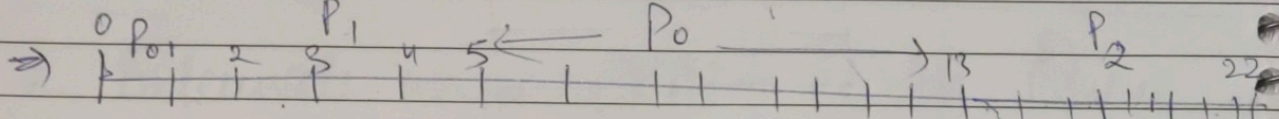
(1) ⇒

Arrival		CPU
0	P_0	9
1	P_1	4
2	P_2	9

SJF

Pre-emptive





Turn around time ⇒ $P_0 = 9+4 = 13$

(CPU + Waiting time)

$P_1 = 4+0 = 4$

$P_2 = 13+9 = 22$

Waiting time ⇒ $P_0 = 4$

$P_1 = 0$

$P_2 = 11$

Turn around time = $13+4+20 = 37$

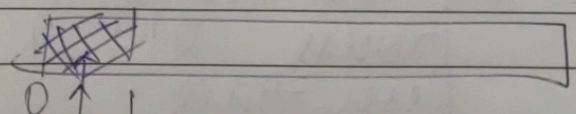
Avg. turn around time = $\frac{37}{3}$

$9+4+9$

3 3

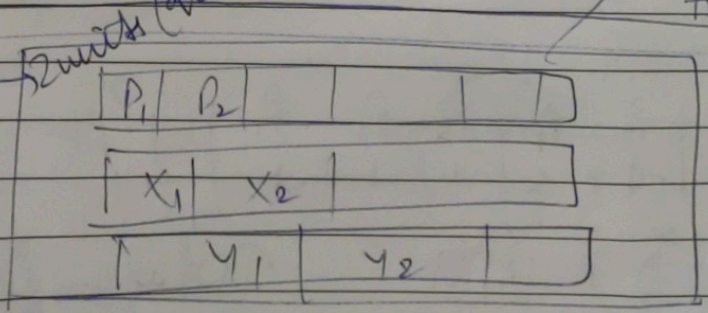
If in above ques.

Arr. time	
1	P_0
1	P_1
2	P_2



Idle (खाली)

Multilevel Queue



q_1 more priority than q_2 than q_3

FREEMIND

Date 30/8/22

Page

→ Multilevel Queue :-

Q→ 4 Processes - Multilevel Queue Scheduling

	Arrival	CPU	Queue No.
P_1	0	4	1
P_2	0	3	1
P_3	0	8	2
P_4	10	5	1

Periority > Priority

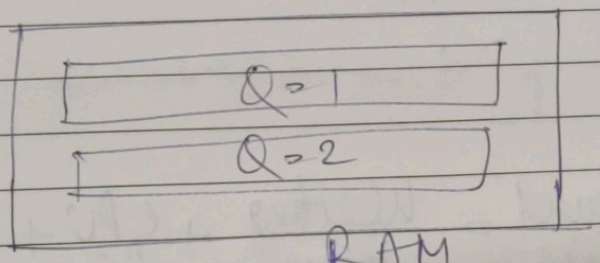
Queue 1 > Queue 2

Queue 1 - Round Robin

T.Q - 2 units

Queue 2 - FCFS

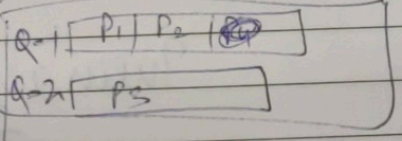
→



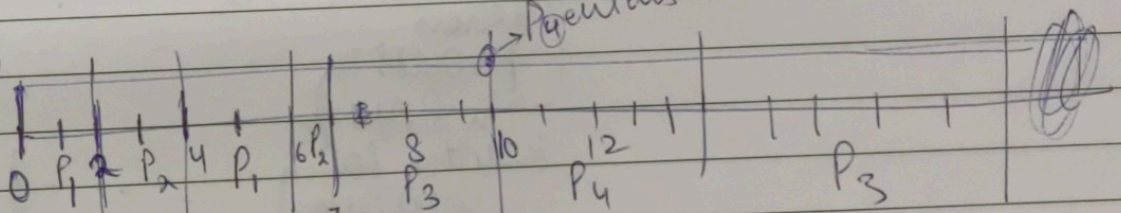
1 - P_1, P_2, P_4

2 - P_3

t=0



Ans →



GANTT CHART

Multilevel feedback Queue :-

1 Process

CPU Bound

Burst time = 40 sec

Q_1 = 2 sec (Quanta)

Q_2 = 2+5 = 7 sec

Q_3 = 7+5 = 12 sec

Q_4 = 12+5 = 17 sec

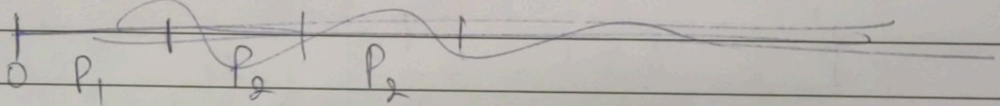
Q_4 में समाप्त हो जाएगा।

OS & Books.com

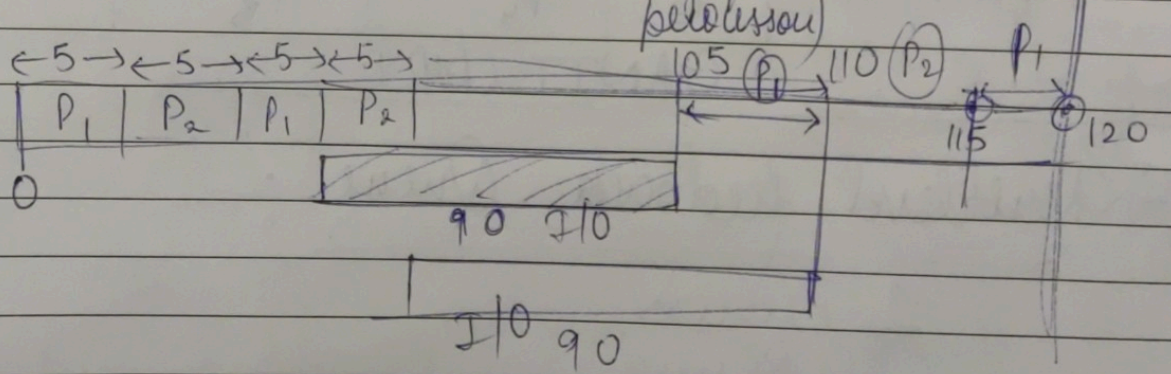
Q⇒ There are 2 processes - P_1 & P_2 created at same time $t=0$, executed alt sequence = Alternatively. I/P - O/P Burst = 90ms
BT I/O "
$P_1 = 10$ m sec
$P_2 = 10$ m sec. I/O of both P_1 & P_2 can go in parallel. Round Robin = 5 ms (Quanta).
$P_1 P_2$ → (I/O) → फिर CPU BT = 10ms का खत्म हो चुका होगा

Compute turnaround time until P_1 enters again & hence stopped.

→



Turnaround time = Waiting time (when process is in RAM & waiting for permission) + CPU BT + I/O



$P_1 = 120 + 90 = 210$

FREEMIND

Date 1/9/22

Page

Why CLI is fast ? = Because of no graphical intervention.