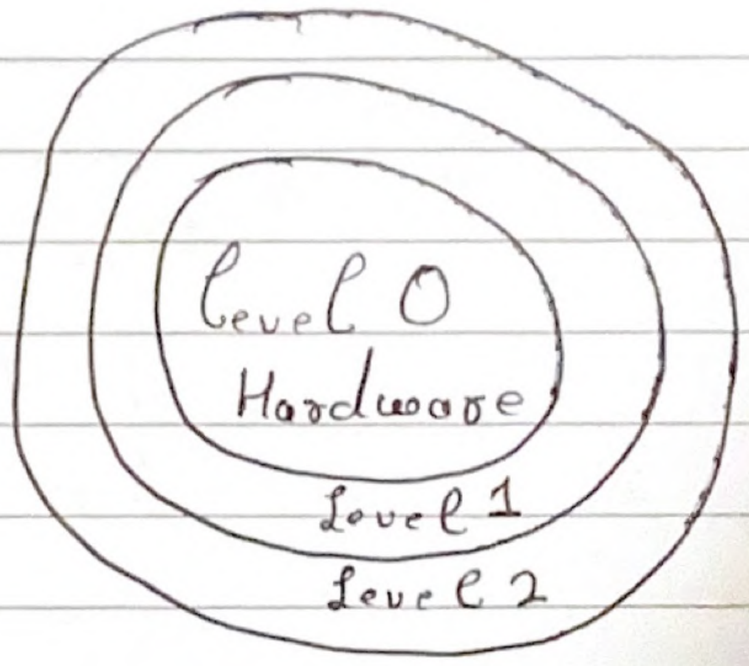
OS

* Jab pehle user direct hardware ke saath Interact karta tha tab dikkat aati thi.

* Os aaya

→ user ko hardware access karne ke liye os ko command deni padi

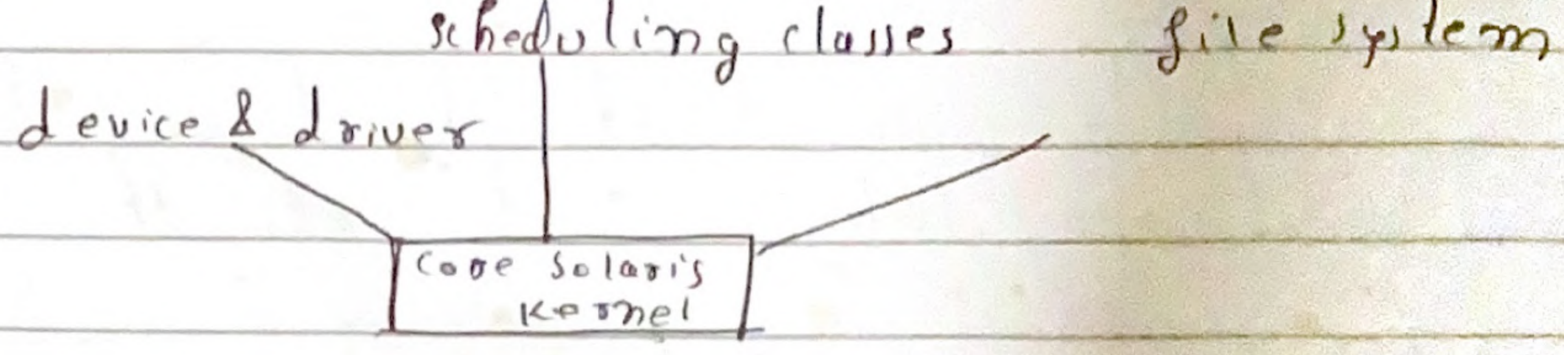
Layered OS



Every layer has specific function

Upper Layer Lower layer ka use karta hai

Solaris



Jiski requirement hoti usko call kara jata

Isme bhi bura DS ram mein load hota. so Isme jo kernel hai Ye vo heart of os wala nahi hai

Micro Controlle. Kernel → Isme sirf essential services load hoti hai aur agar kuch secondary service chahiye hoti hai toh phir vo load hoti hai.

Types

Single Programming →

OS
UP

Batch Processing → programs ka ek batch banake ram ka sonp dena

Multi Programming →

OS
P_1
P_2
P_3

Multi Tasking → Multi Programming + Time Shared OS

Context Switch → Jab processor ek process ko chodkar dooste pe jaate hai processor nahi OS decide karta hai

Preemption → Jab time shared OS hata hai ek minimum time deta hai Jab Ye hata hai →

5 ms → → → ↓ switch → P_1 → 30 ms = 25 ms

P_2 → 5 ms = complete

P_2 preempted P_1

System Call

Jab aap as a user hardware ki kisi bhi ya koi bhi resource access karna chahte ho toh aap system ko (OS ke Kernel ko request) karte ho i.e Generating System Call

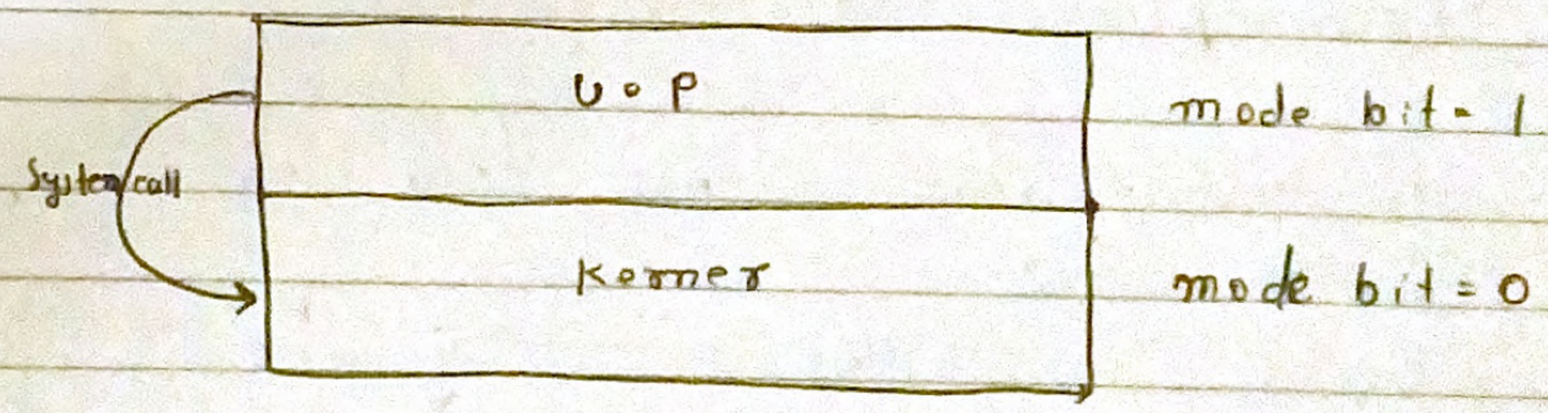
Copy

Source ——————————→ dest

Acquire input file name
write Prompt to screen
Accept Input

:
:
:

Mode width



System Call

Jab aap as a user hardware ki kisi bhi ya koi bhi resource access karna chahte ho toh aap system ko (OS ke Kernel ko request) karte ho ie Generating System Call

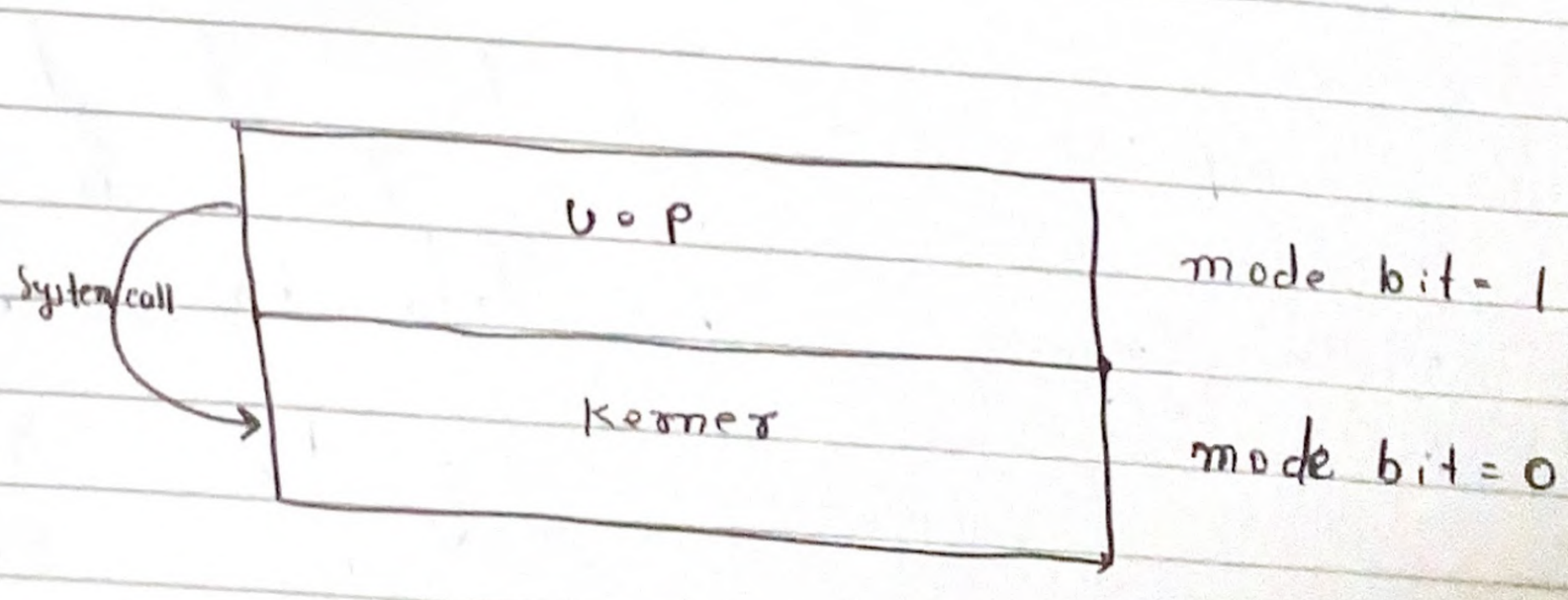
Copy

Source ——————————→ dest

Acquire input file name
write Prompt to screen
Accept Input

#

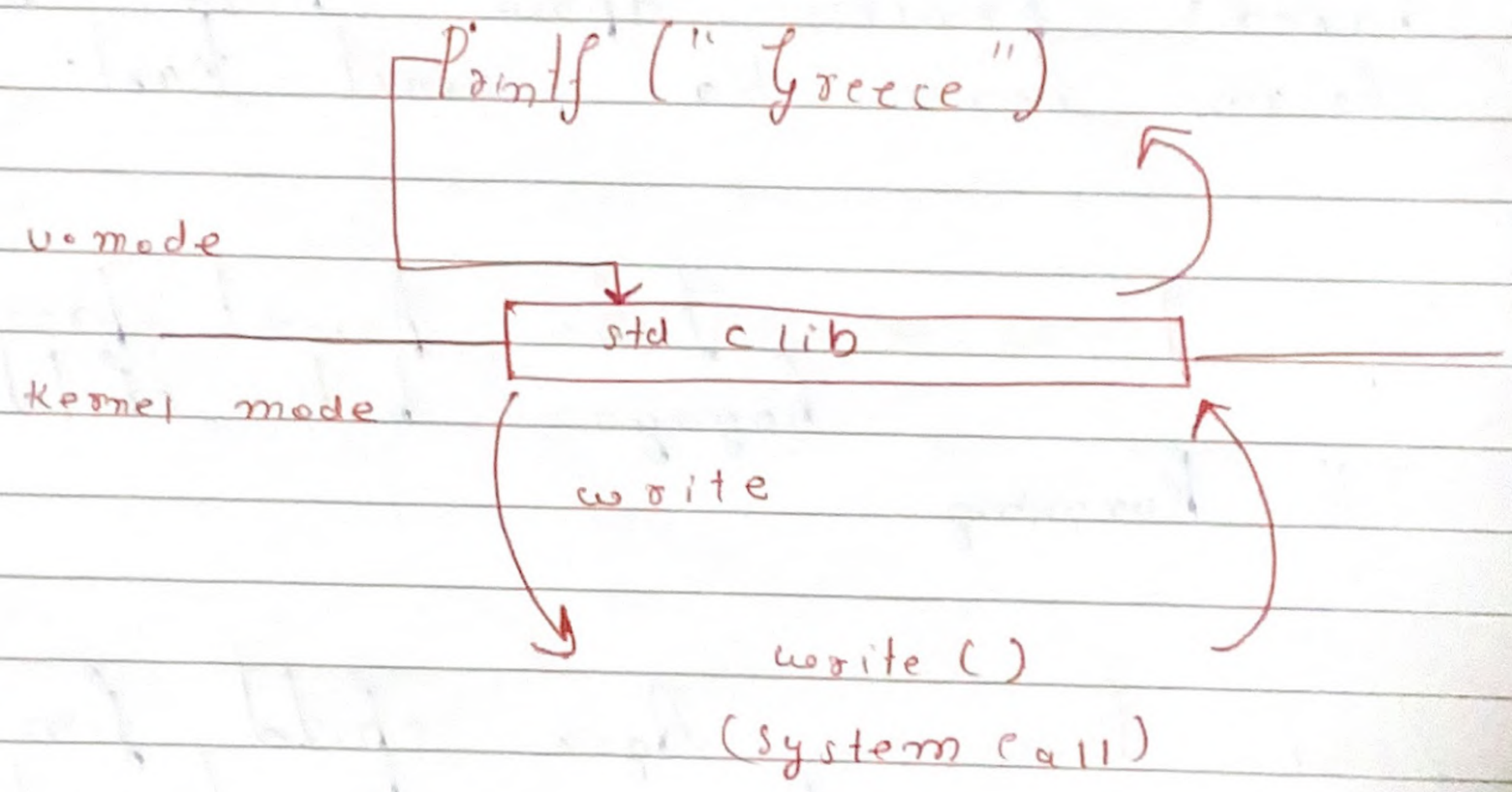
Mode width



Jab process ki mode bit change hoti hai from 1 to 0

Jab tak processor jab kuch nahi kar rahta tab vo program processor ke liye gayab ho jaata hai.

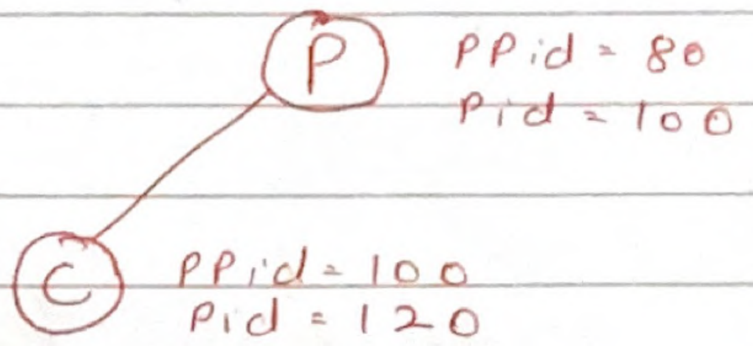
CPU sirf unhi process to process karta hai jo user mode mein ho



Process Creation

* Har ek process ka parent hota hai
* Aur Unke paas hota hai ppid, pid

PPid = Parent ki Pid
Pid = khud ki Pid



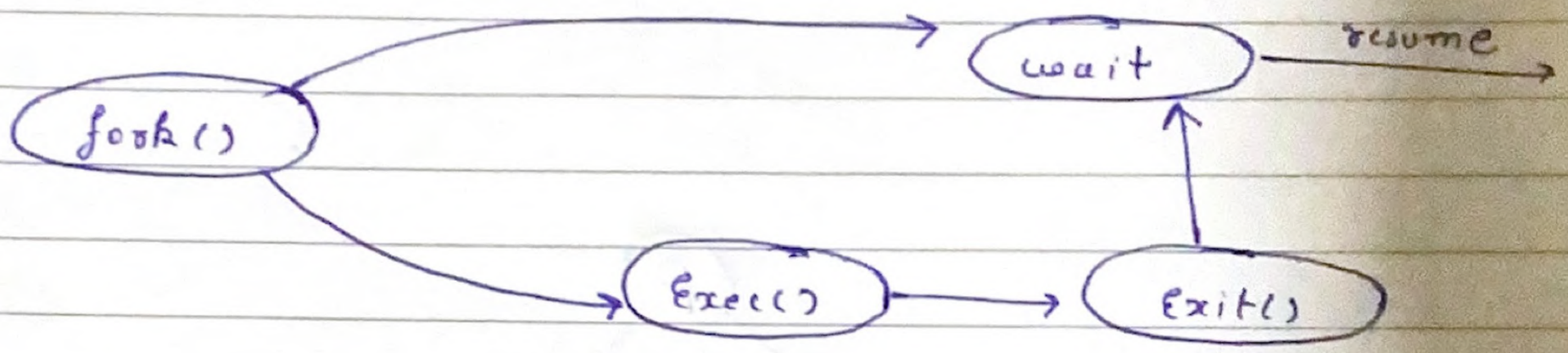
★ Parent process apne child ka execution khatam hone ka wait karti hai

Orphan Process = Jiska parent process execute hogaya but child process still Running.

Zombie Process = Agar child finish hogaya hai aur vo parent ko complete hone ki permission de raha hai but Parent hee sleep mode mein hai toh vo accept Nahi kar raha.

Address Space

- Child duplicate of parent
- Child has a program loaded into it



★ fork() ⟶ create child
Aur vo parent ka
copy hota hai

Next line se execute hota hai

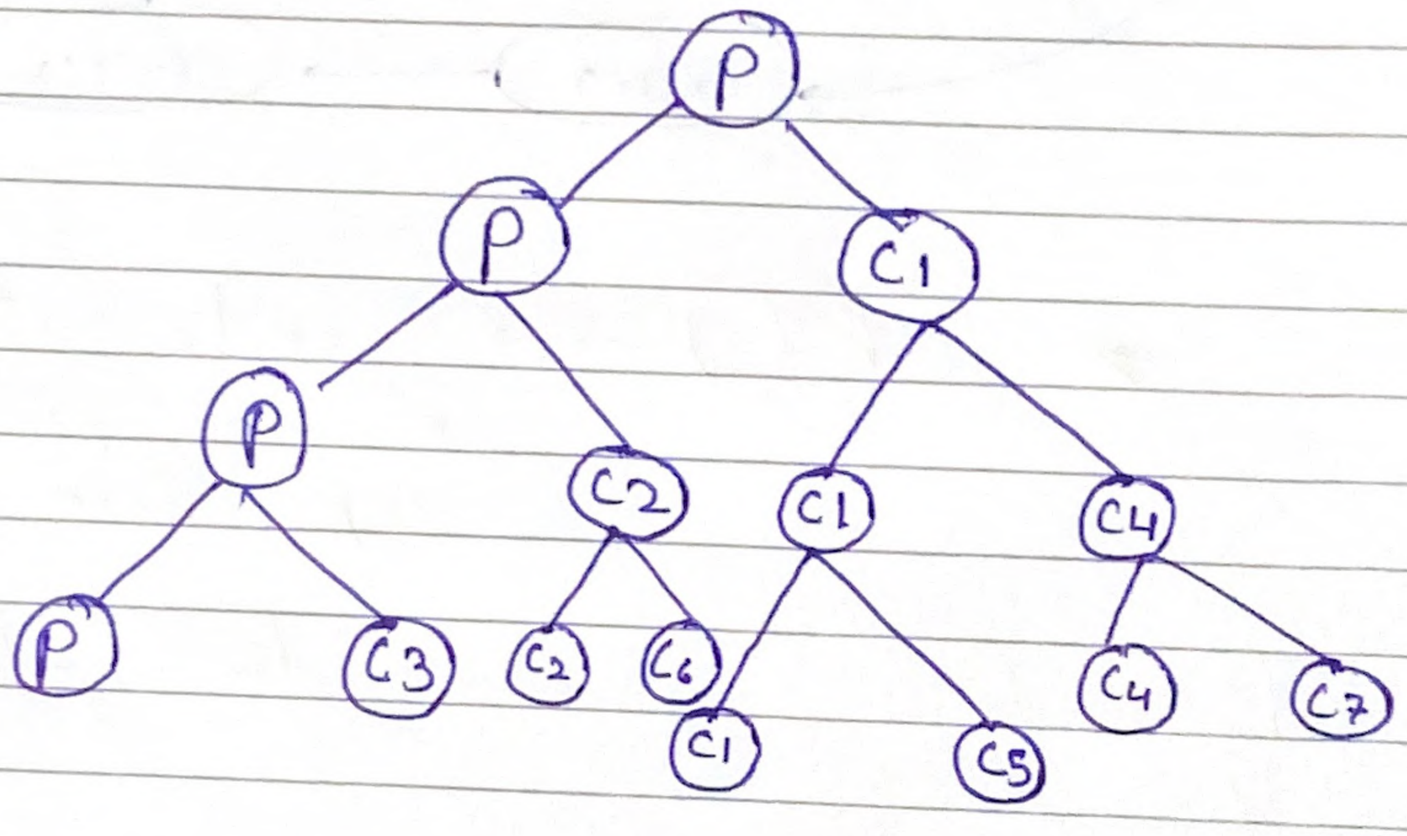
P C

30→ fork 31 ⟶ execution start.

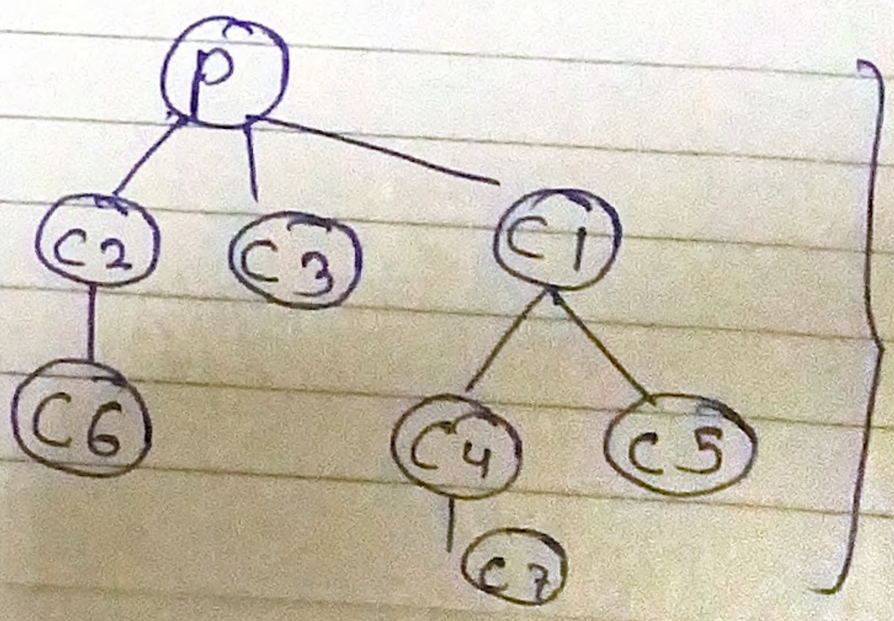
★ Exec ⟶ New code copy karta hai memory se aur 1st line se execute karta hai.

Diagram AAna Chahiye

```
main() {
    fork() C1
    fork() C2 , C4
    fork() C3 , C5 , C6 , C7
    pof ("Hello")
}
```



Simplify



Simplified

Ab jo fork hai vo parent process mein +ve value return karta hai

Aur child mein vo Zero hota hai

Code

```
pid = fork ()
if (pid > 0) {
        printf ("parent"); wait (NUL
}
else if (pid == 0) {
        printf ("Child");
}
```

execute in parent

let child complete

execute in child

Agar N fork hai toh Child process $2^N - 1$

Ex - N = 3 , 7 Child.

CLASSMATE
Date :
Page :

8

```
main () {
    if (fork () (1) && fork () (2)) {
        pof ("Hello")
    }
}
```

→ Parent Process mein (1) = +ve true

→ It will create a child C1 in which value of (1) will be = 0 so due to && it will not see (2) in C1 because && mein agar pehla 0 hai toh aage kyon chale.

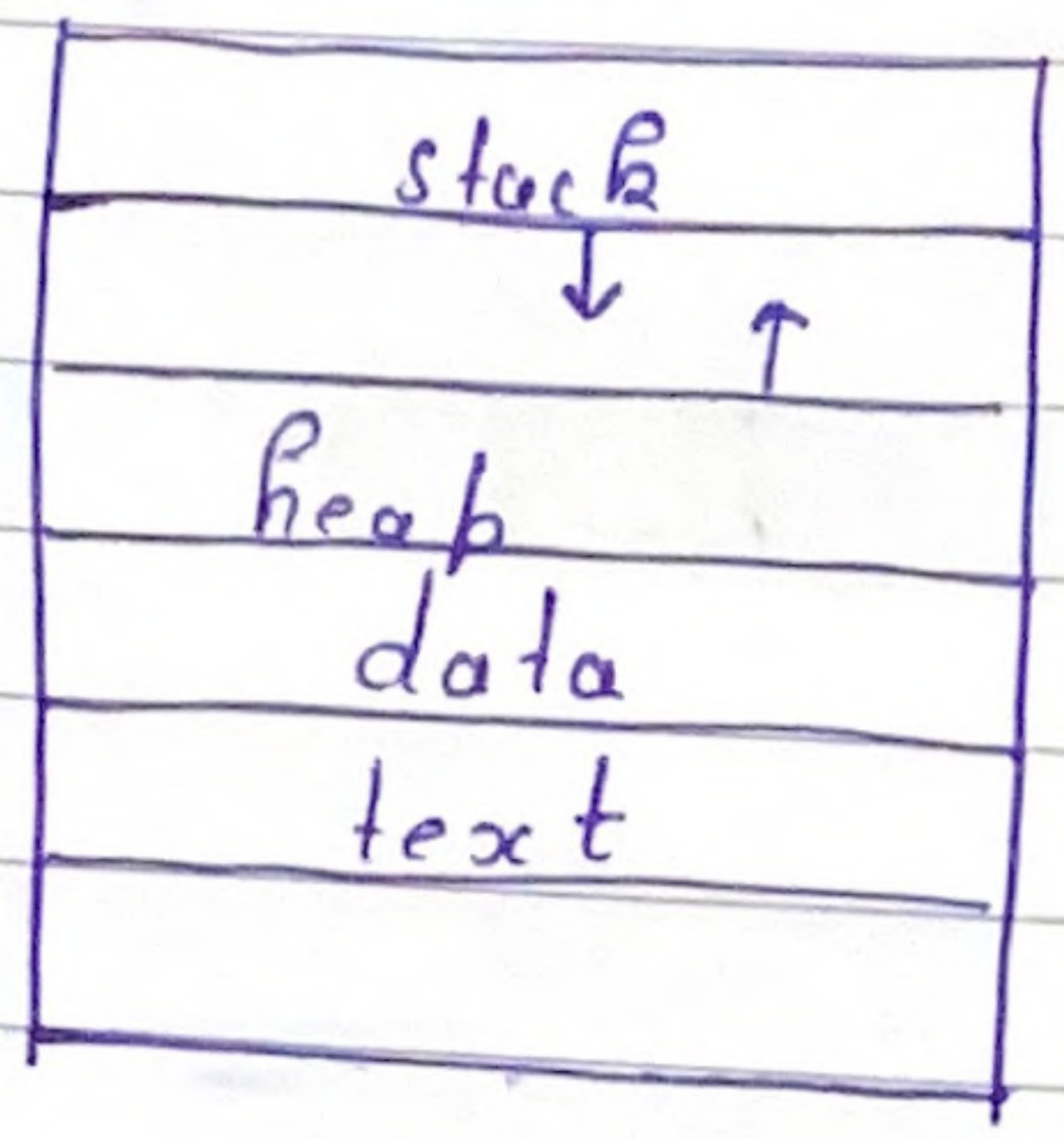
→ Back in parent process

→ (2) execute hoga +ve Generate kar-ga
(1) & (2) → true hai pof ("Hello")

→ Ab jo naya process wo (1) ko 0 (2) ko bhi 0 Maanega toh Nahi hoga → Printf ("Hello")

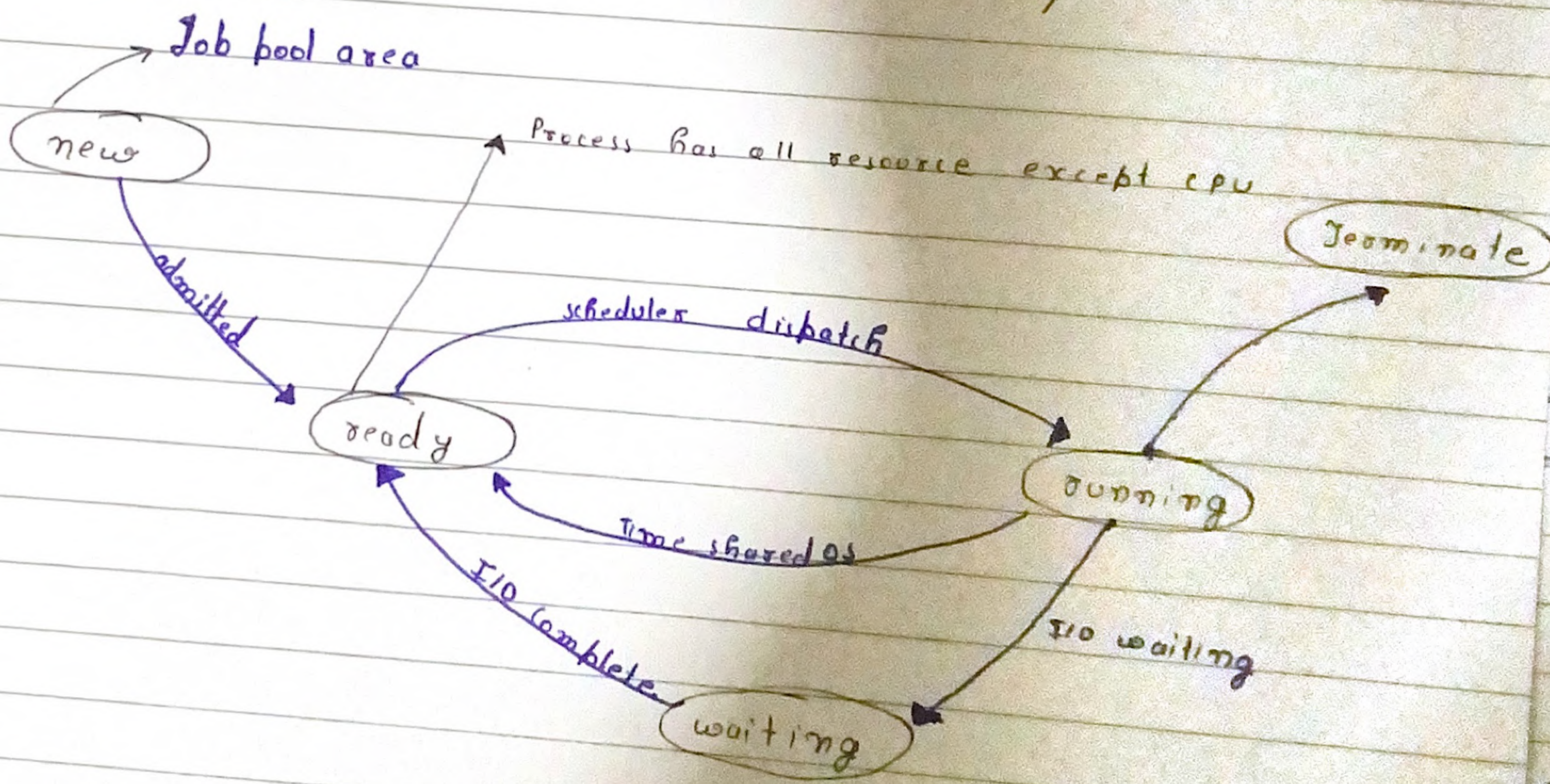
→ only one time.

Process Concept



har ek process ka stack ya aap kaho ek process ka structure aisa hota hai jisme stack or heap variable hote hai acc to program

* Program is a passive Entity
* Process is Active Entity.



p2
b2 chalega

Jab koi nayi process new se ready state mein se ready state mein aati hai tab uska naya PCB banta hai.

PCB (Process Control block)

Process state
Program Counter
CPU Register
CPU Scheduling Information
Memory Management
Accounting - Information
I/O status.

CPU Scheduling.

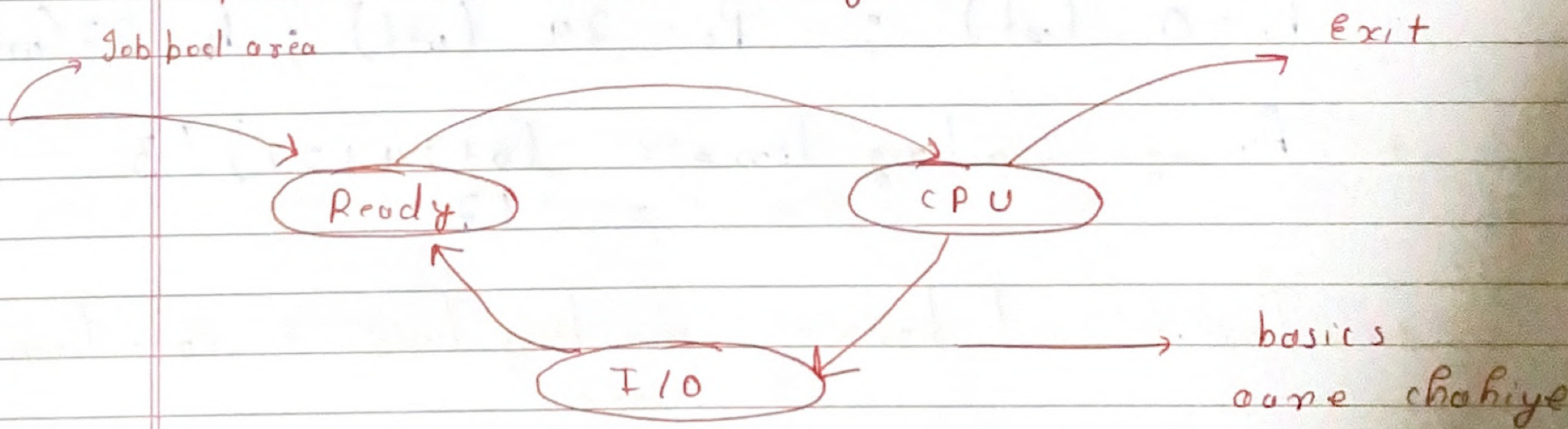
Isme concentration upper level ki chahiye.

Saari possibilities, Saare Algorithms ka Gang chart banega uske according Question solve Make sure complete focus is required.

FCFS

Concept hai Simple jo pehle aayega vo pehle execute hoga.

Process ka state diagram

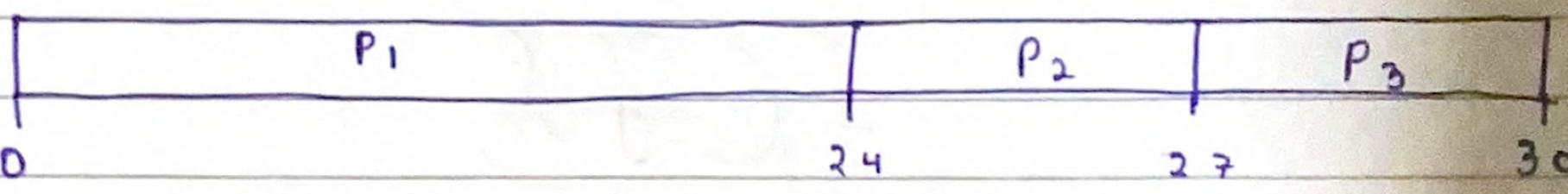


Jiska arrival time kam hoga Vahi usko process milega.

Process	Burst / Execution time
P_1	24
P_2	3
P_3	3

Ab isme saare process Zero time pe aaye hai toh Jiski Process id Kam hai prefrence usey milegi

Gang Chart



* waiting time = Arrival time = 0, to CPU Mila (P_1)
= 0
= CPU Mila - Arrival time
= 0

$P_1 = 0$ (wt) ; $P_2 = 24$ (wt) ; $P_3 = 27$ (wt)

* Average waiting time = (0 + 24 + 27) / 3
= 17

* Turn around time = waiting time + execution time.

SJF

(Shortest Job First)

Ab time kya hoga jiska execution time sabse kam hoga vo execute hoga pehle (watch arrival time of process)

Agar 2 process mein conflict ho jaye

P_1	4	P_1 chuna jayega acc to Process id.
P_2	4	

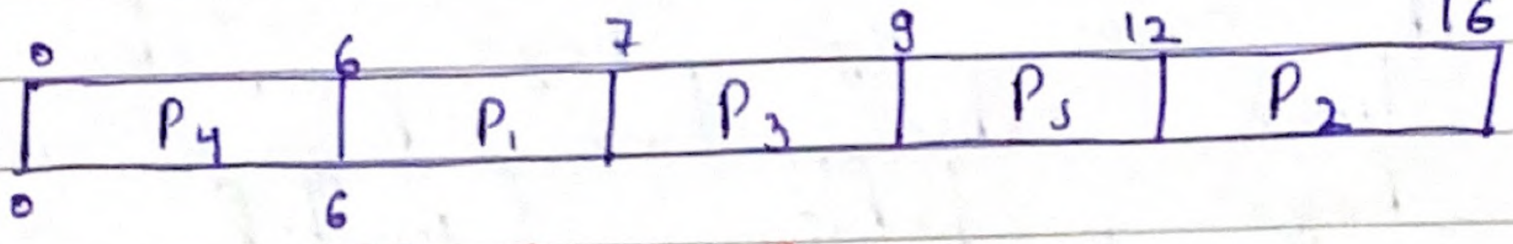
Ques

P_1	3	1
P_2	1	4
P_3	4	2
P_4	0	6
P_5	2	3

A.T = 0 → only 1 process toh vo pehle

Take Care of Arrival Time

Then bekla process execute hone ke baad jaane process aa jayenge ready Que mein.



Gantt chart

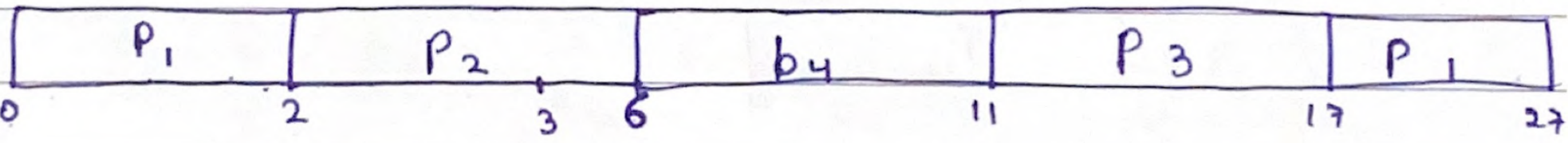
Conflict => Agar 2 process ka execution time same hai tab hum low process id ka prefer karenge.

SRTF

Ab isme concept aata hai preemption ka

* toh agar current execution mein jo process uska execution time (remaining) jyada hai kisi ready state ki process hai toh ready wali process ko hum execution mein layenge toh vo ho jayega preempted.

	A.T	Burst time
P_1	0	12
P_2	2	4
P_3	3	6
P_4	4	5



After 2 sec P_2 will arrive ex time = 4 b_2 will preempted b_1

b_2 → 1 sec execute p_3 → arrive = Ext time = 6 sec b_2 = 3 → b_2 chalega

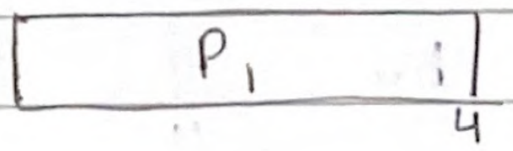
b_2 → 2 execute p_4 → 5ext time b_2 chalega

Eh baar saari process ready state mein aajaye toh phir easy hai then jiska execution time kam hai vo aayegi

Avg time
$P_1 = 15$
$P_2 = 0$
$P_3 = 8$
$P_4 = 2$

Conflict Agar Current Execution process Ka remaining time same as in ready state

P_1	0	12
P_2	4	8



$P_2 = 8$, $P_1 = 8$

P_1 => will run kyonki Agar P_1 ko ready mein le jayega toh ek chota sa time waste hoga. = Context Switch (Overhead)

Priority

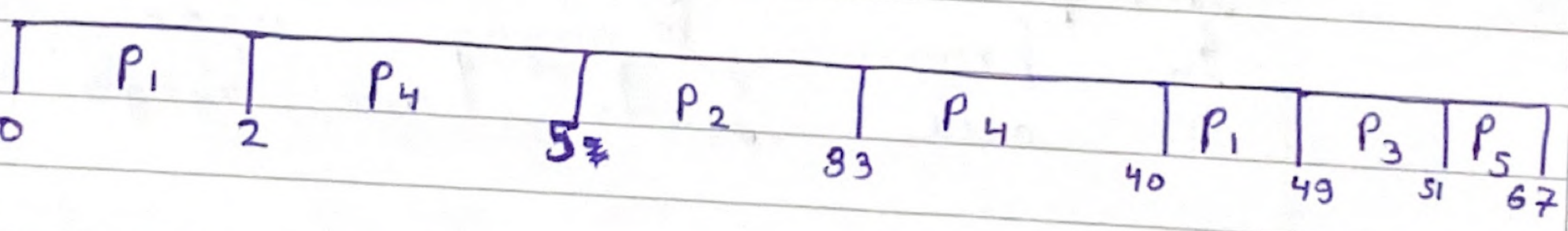
Ab isme jiski priority jyada hogi vo pehle execute hoga

by default =

Smallest Integer = Highest Priority

Arrival time pe focus karna hai agar koi higher priority vala aaya toh pehle

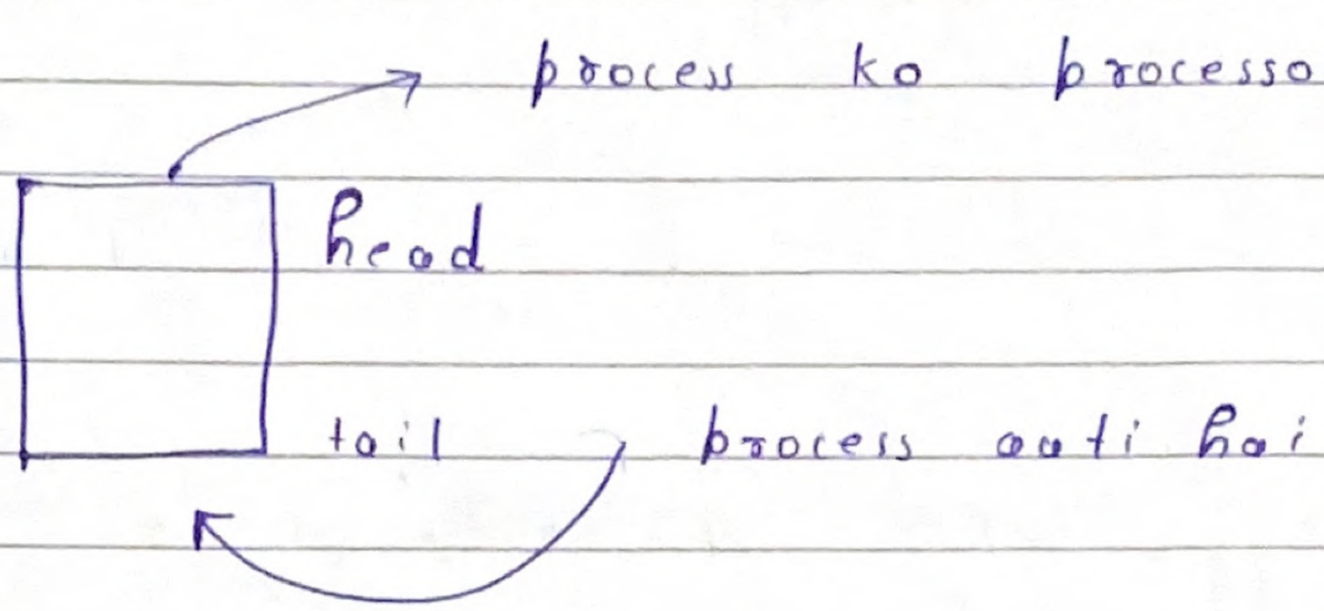
	AT #focus	Burst Time	#compare Priority
P_1	0	11	2
P_2	5	28	0
P_3	12	2	3
P_4	2	10	1
P_5	9	16	4



at 2 P_4 Priority 1 → Preempted
P_4 Poori chalegi → X
P2 5 pe aayegi → Priority high
Ek baar saari aajaye toh priority ke acc chalado

Round Robin

Ab isme ek Time Quanta hota hai ek process ko milta hai Ab isme ek data structure hota hai



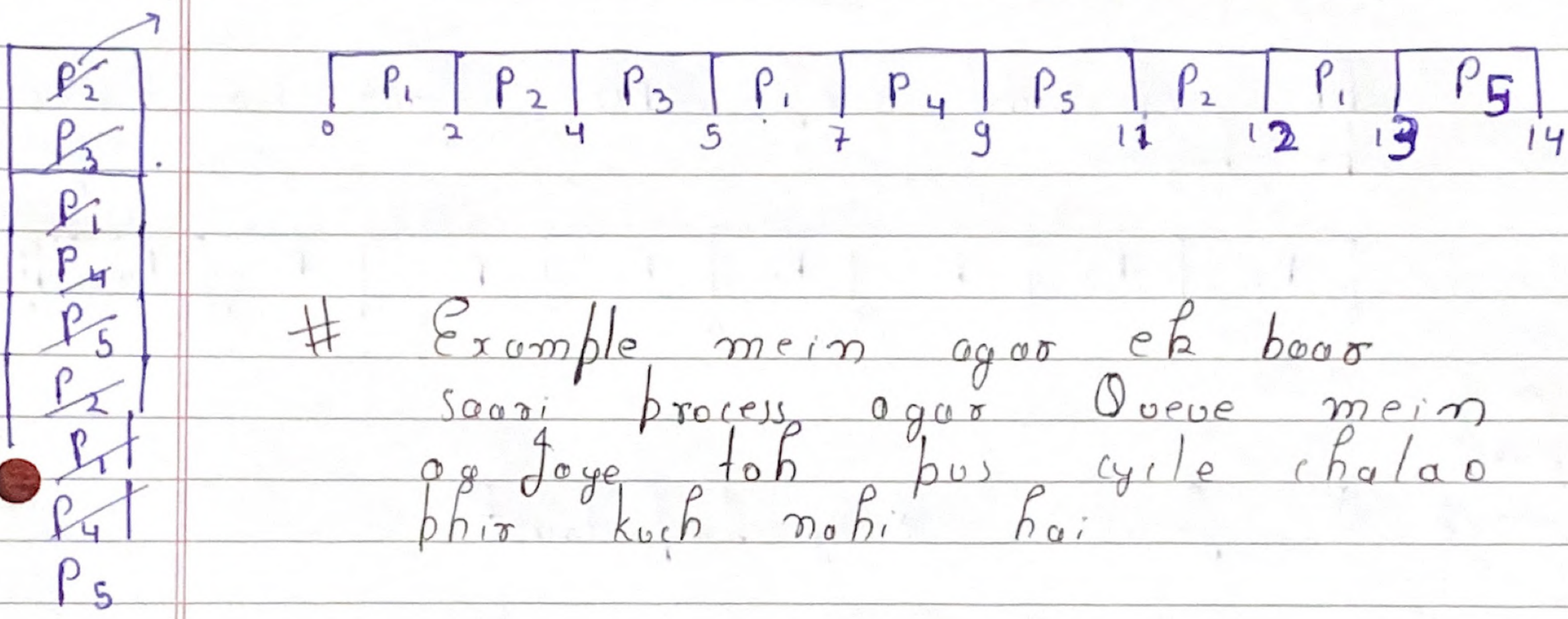
Conflict => Agar koi process processor se aaye aur koi process same samay pe Arrival time pe aaye toh tail pe pehle kisko add karoge ??

New process ko add karenge kyonki jo processor se aayegi usko context - switch ke acc. jyada time lagega.

Head / Tail ka dhyaan Rakhna.

Pid	A.T	Burst Time
P_1	0	5
P_2	1	3
P_3	2	1
P_4	3	2
P_5	4	3

Time Quanta = 2



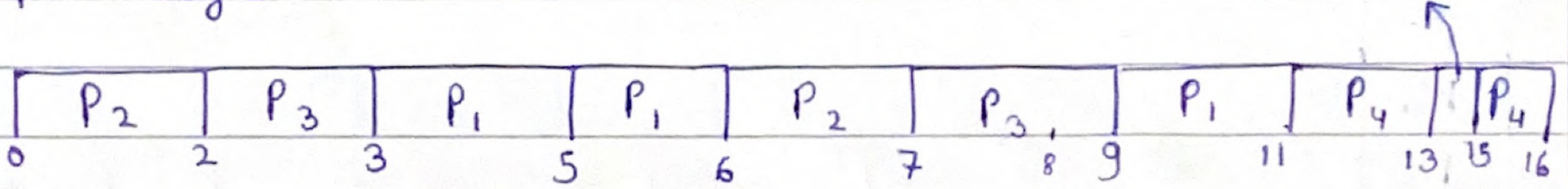
Example mein agar ek baar saari process agar Queue mein aa jaye toh bus cycle chalao phir kuch nahi hai

Agar third state I/O AAgaya

		CPU	I/O	CPU
	P_0 0	3	2	2
7 → ∝	P_2 0	2	4	1
9 − ∝	P_3 2	1	3	2
	P_4 5	2	2	1

SRTF

Ab comparision sirf CPU time ko basis pe hoga



P2 ↛ 6
P3 ↛ 6
P1 ↛ 18

focus is very Important