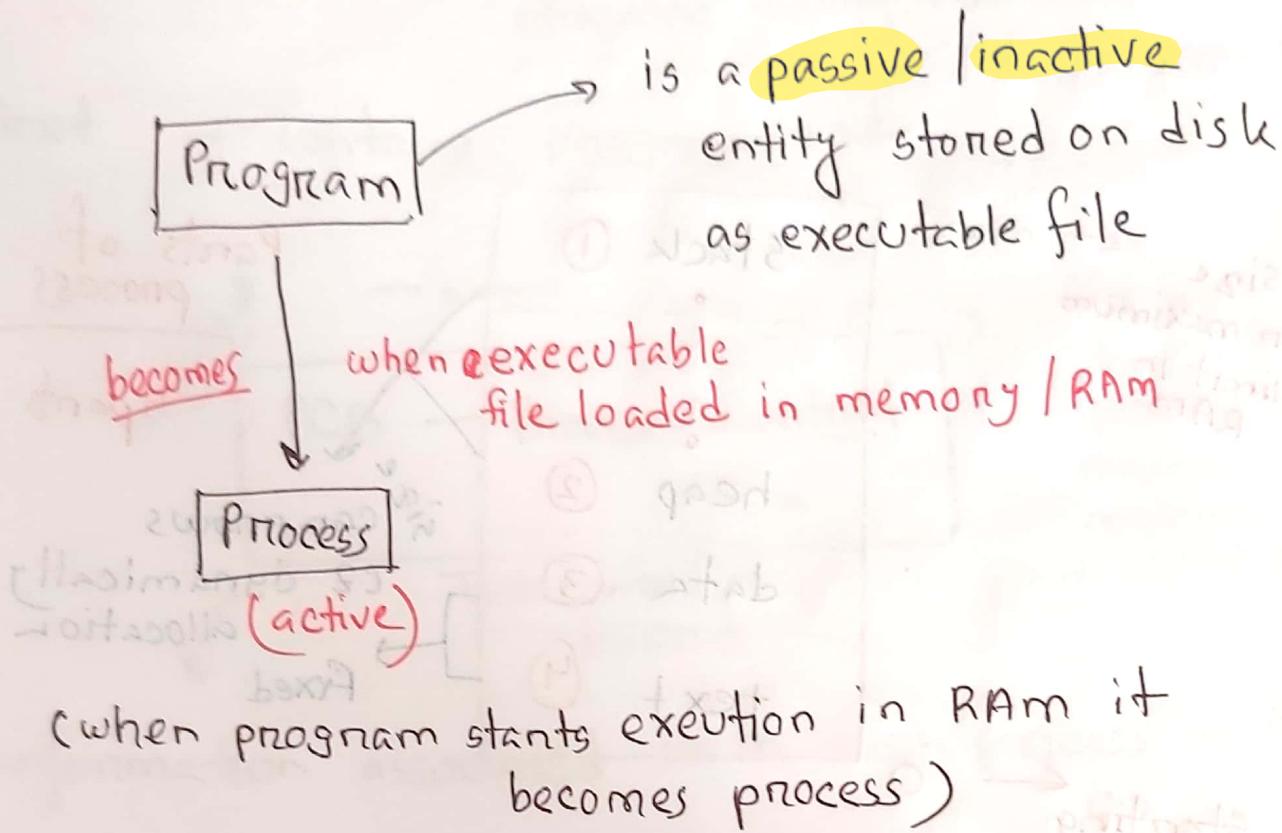


## Chapter-3

### Process



(one program can have several process)

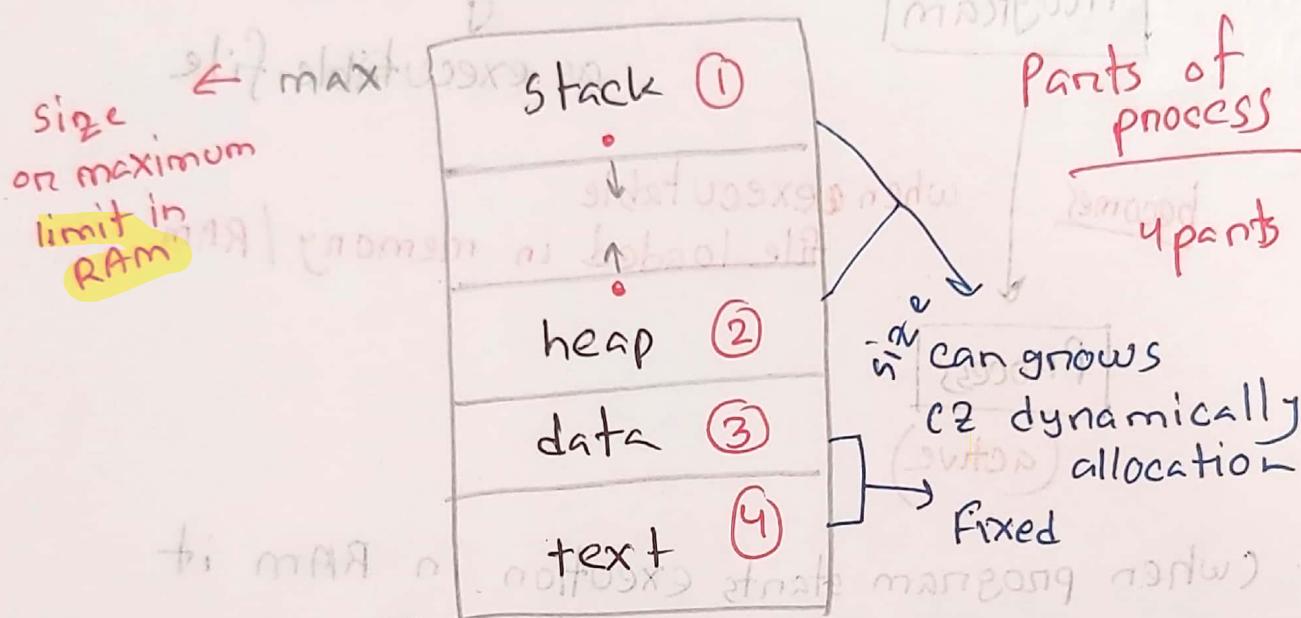
\* To start a program you must bring it to ram

\* program consists some instruction

## Process

## Components

→ Process is a program in execution



starting address of program

(How does 1 process look like in memory / RAM)

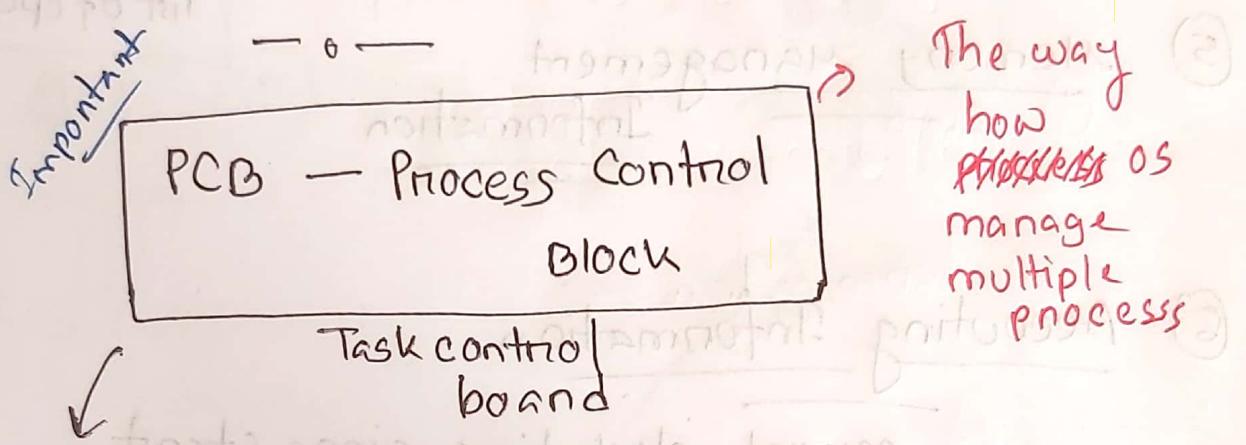
① stack → contains program code

→ contains temporary data

→ function parameters,  
return addresses  
Local variables

↓  
inside in function

- ② Data Section → contain global variables
- ③ Heap → contain memory dynamically allocated during run time
- ④ Text → contains program code



information associated with each process in the

### ① Process state:

- i - (new)
- ii - (running)
- iii - (waiting)
- iv - (ready)
- v - (terminated)

### ② Program counter : Location of instruction to next execute the (where starts of execution)

③

CPU Registers:- content of all process centric registers

Process state
Process number
Program counter
Registers
memory limits
list of open file

④

CPU scheduling Information

Priorities, scheduling pointers

⑤

Memory management

Information

⑥

Accounting Information

cpu used, clock time since start

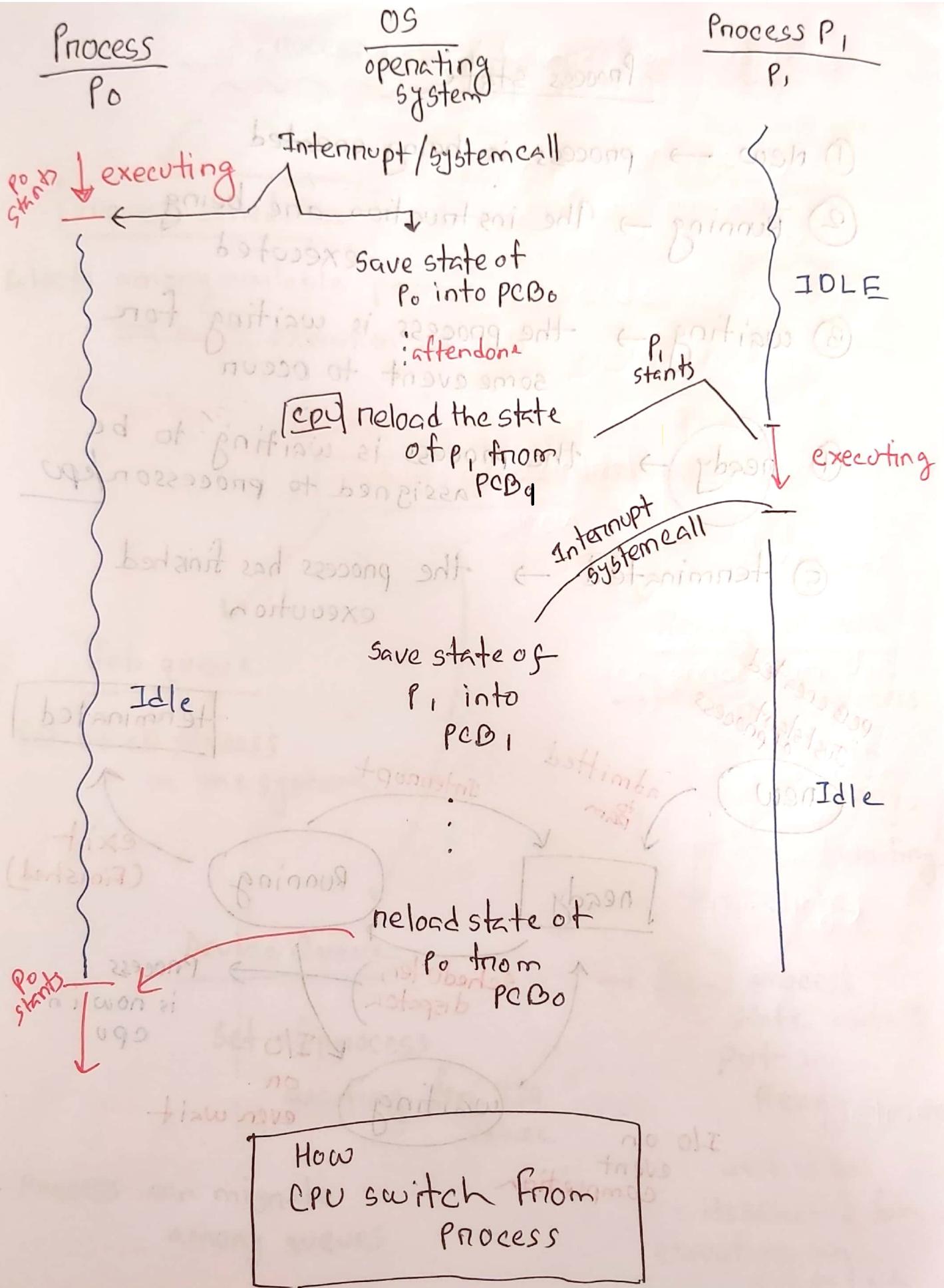
⑦

I/O status Information

I/O device allocated to process,  
list of open files

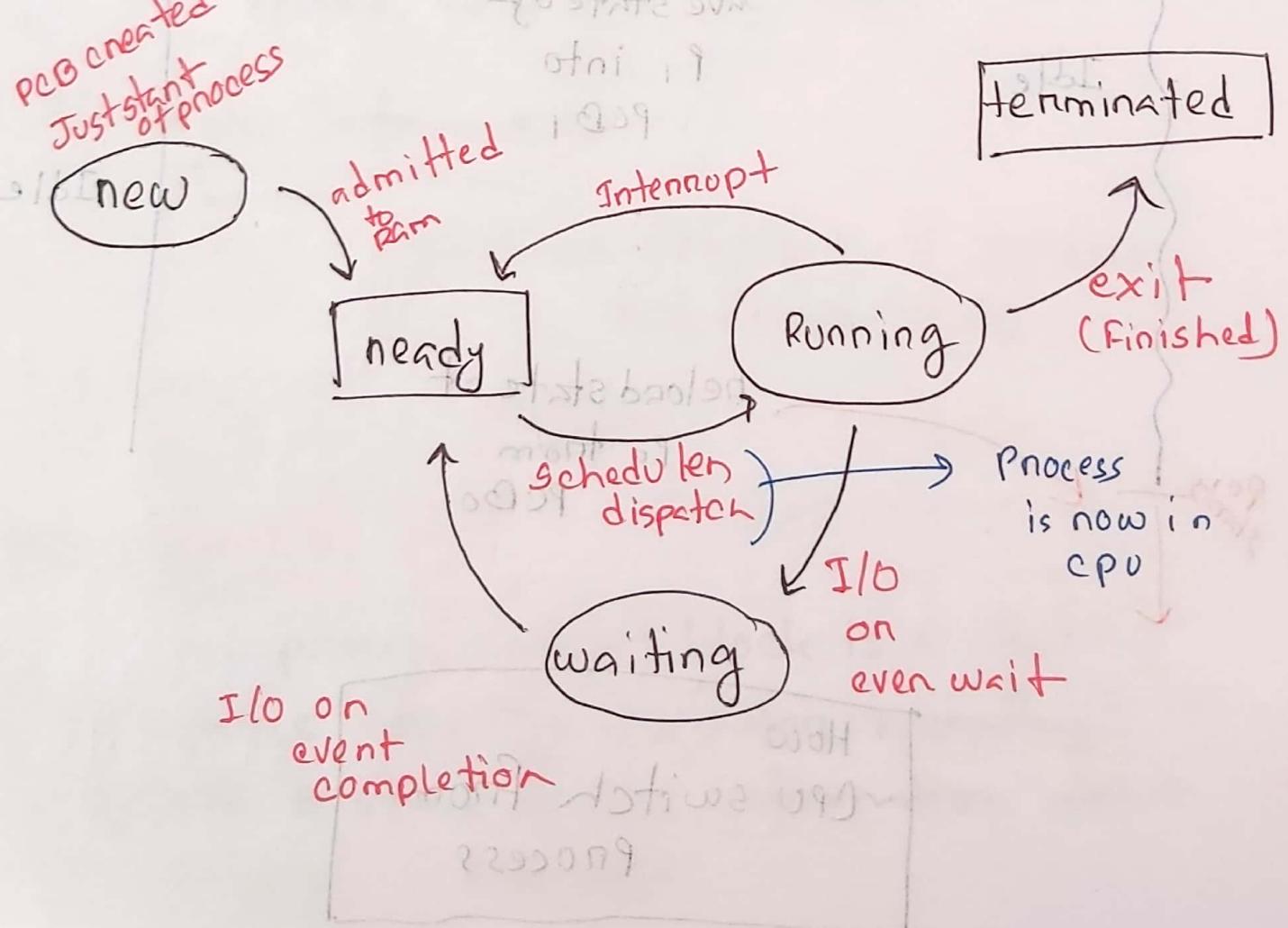
PCB definition

a process control block is a data structure used by computer operating system to store all the information about a process



## Process state

- ① New → process is being created
- ② Running → The instruction are being executed
- ③ waiting → the process is waiting for some event to occur
- ④ ready → the process is waiting to be assigned to processor/cpu
- ⑤ terminated → the process has finished execution



# Process scheduling

## Process scheduler

selects among available processes  
for next execution on CPU

maintains

scheduling

queues of processes

## Job queue

Set of all processes  
in the system

## Device Queue

Set of processes

waiting for I/O  
device

maximize CPU use  
quickly switch  
CPU for time sharing

## Ready Queue

→ Set of all processes  
residing in main memory,  
ready and waiting to execute

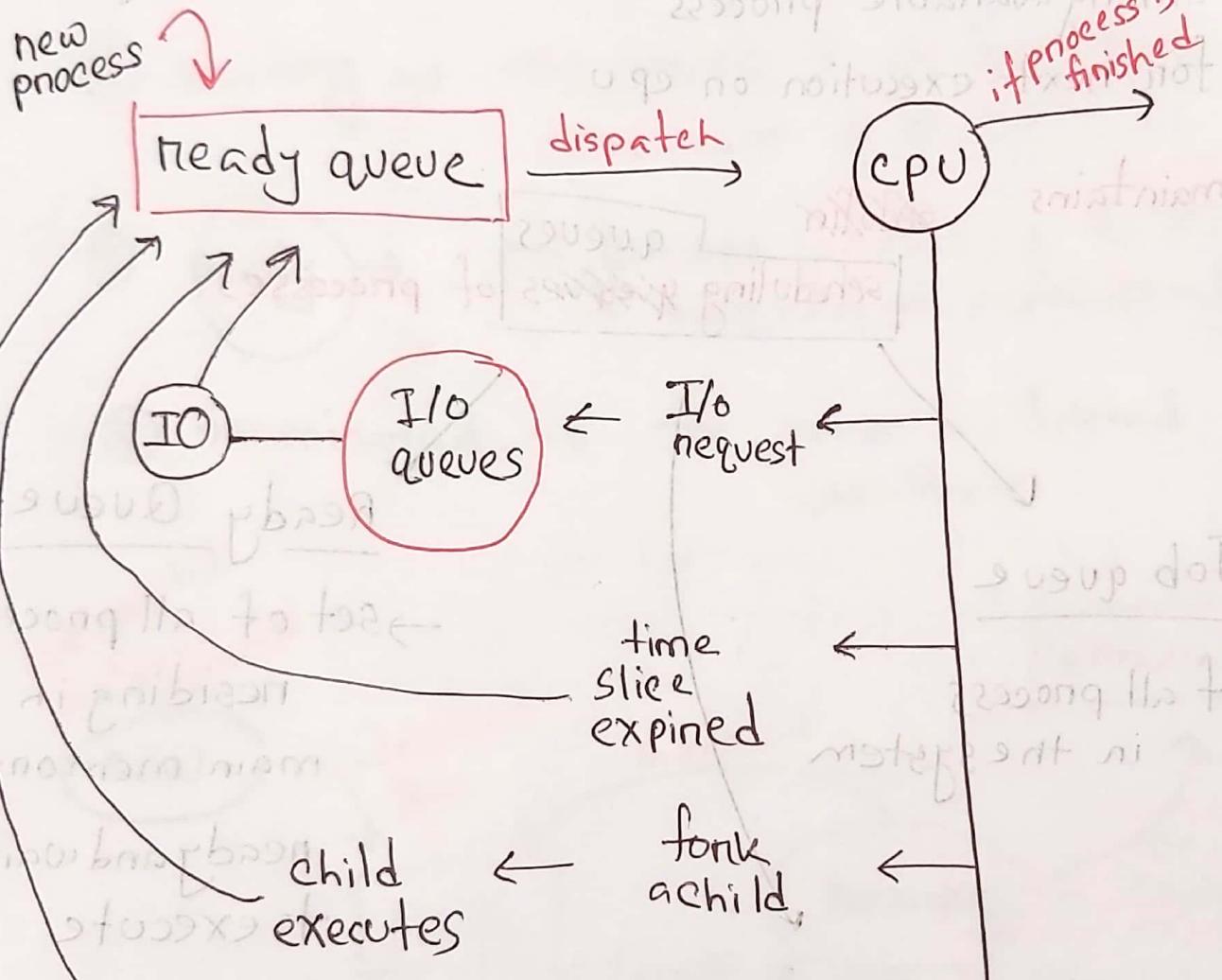
→ New process state initially put in Ready Queue

\* Process can migrate  
among queues

wait until its selected for execution on dispatch

## Representation of Process scheduling

### Queueing diagram



# Schedulers

## Short term scheduler

- CPU scheduler
- which sends a process to CPU
- select which process should be executed next and allocated to CPU
- must be very fast (is invoked / called frequently) (milliseconds)

## Long term scheduler

- Job scheduler
- selects which processes should be brought into the ready queue in main memory
- strives for good process mix
- may be slow (is invoked / called infrequently) (seconds, minutes)

→ controls the degree of multiprogramming

definition ↓

how many programs should allow at same time on admitted to main memory

## Types of process

### ① I/O bound process :-

- spend more time doing I/O than computations
- many short cpu bursts,   
bursts

### ② CPU-bound process

- spend more time doing computations
- few very long cpu bursts,   
bursts

## Medium-term scheduler :

- remove process from memory
- store <sup>in</sup> disk
- bring back in RAM to continue execution

## Context Switch

*(definition important)*

① the system must **save** the state of old process

and **load** the saved state for new process via context switch

when a cpu switches

② to another process

it is called context switch

(context of process represented in PCB)

while context switch system does not do useful work

definition :- ① + ②

## Process Creation

Parent process create **child** process which in turn create other process , forming a tree of processes

① Chrome Browser is a multiprocess with 3 different type of process

④ Browser process : manage user interface, disk, network I/O

② Render process : render web pages, deals with HTML and javascript

③ Plug in process : for each type of plugin

⑤ + ① - ; nothing

— o —

Based no Interprocess communication

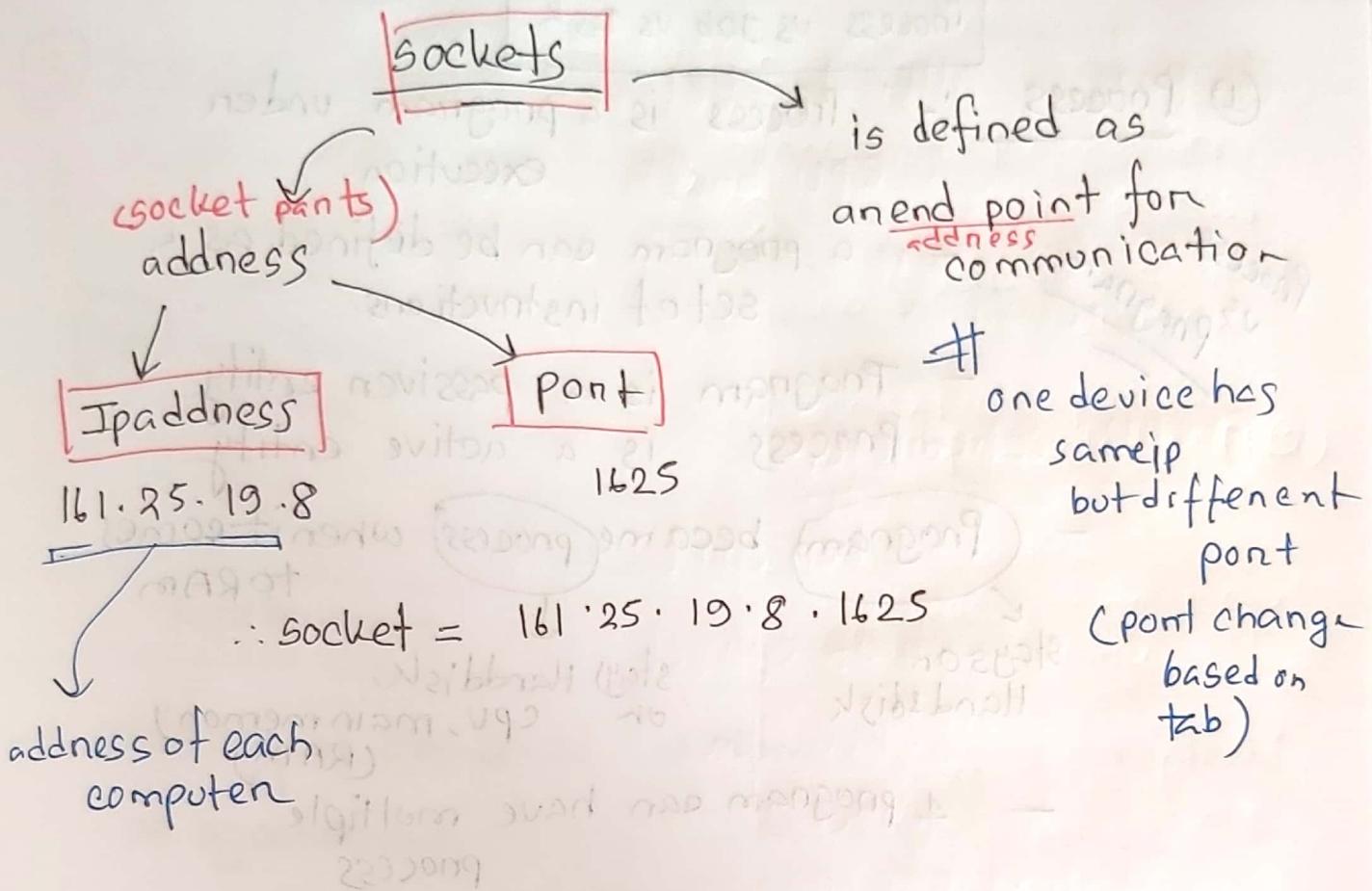
Process type

① Independent process

(not affected by other process)

② Cooperating process

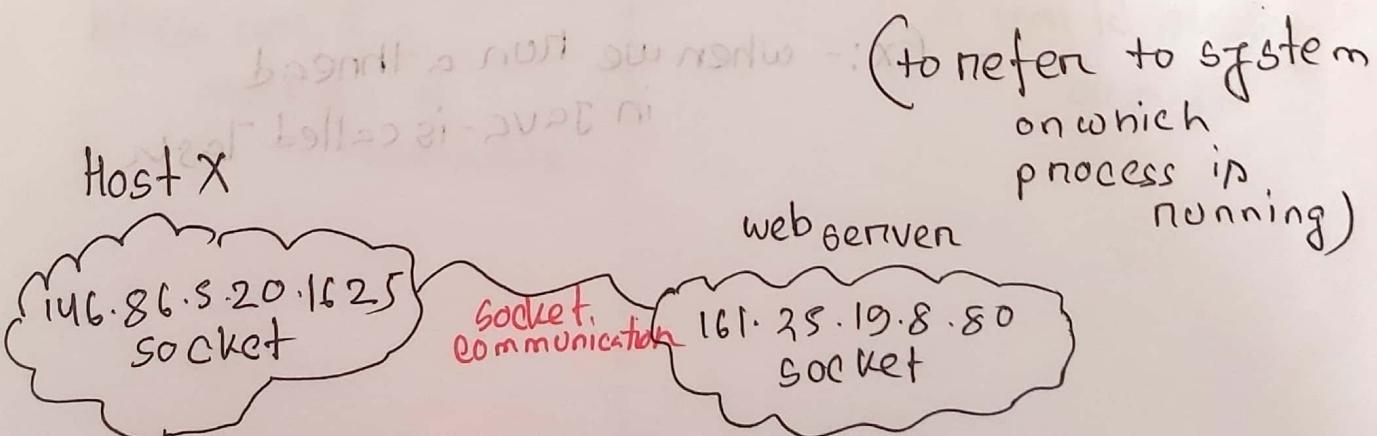
(affected by another process)



# communication  
consists between a pair of sockets

# port < 1024 → well known ports  
(used for standard  
Internet service)

# special IP address: 127.0.0.1



## Process vs Job vs Task

### ① Process

Process is a program under execution.

— a program can be defined as a set of instructions.

— Program is a passive entity  
Process is an active entity

- Program became process when it comes to RAM
- 1 program can have multiple process

Program  
stays on  
Harddisk

process  
stays Harddisk  
or CPU, main memory  
(RAM)

### ② Task :-

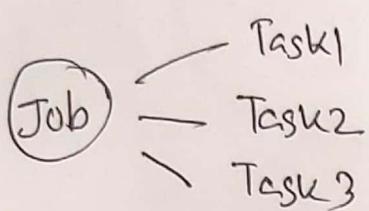
Task is a unit of work being executed

Task in OS may be synonymous with processes

→ Task is a sub part of job

Ex:- when we run a thread in JAVA - is called Task

③ Job :- → a job is a complete unit of work under execution



- a job consists many task which in turn consists of many process

Synchronous

Asynchronous

→ Blocking is considered as synchronous

→ Nonblocking is considered as asynchronous

→ Blocking send has the sender block until the message is received

→ Non blocking send has the sender send the message and continue

→ Blocking receive has the receiver block until a message is available

→ Non blocking receive has the receiver receive a valid message or null

④ Process Altenate between  
CPU bursts and I/O bursts

### ⑤ CPU scheduler:-

when CPU becomes idle, it is the job of CPU scheduler / short term scheduler to select another process from the ready queue to run next on CPU.

⑥ Scheduler / dispatcher :- is a module that gives the control of CPU to the process selected by the scheduler.

## Types of sockets

actually  
types

- ① Datagram socket  
(connectionless)
- ② Stream socket  
(connection-oriented)
- ③ Raw socket
- ④ Sequenced packet socket

access  
Prot col

UDP  
(UDP)

User Datagram  
(TCP)

Transfer control  
Protocol  
(ICMP)

Internet  
control message  
protocol

(IDP)

Internet  
gate protocol

## Degree of multiprogramming

Degree of multiprogramming describes the maximum number of processes that a single processor system can accommodate efficiently.

## Process

## Thread

- |   |                                       |
|---|---------------------------------------|
| ① Process means any program is in execution | ① Thread means a segment of a process |
| ② Process takes more time to terminate      | ② Thread takes less time to terminate |
| ③ Takes more time for creation              | ③ Less time for creation              |
| ④ More time in context switching            | ④ Less time for context switching     |
| ⑤ Less efficient in communication           | ⑤ More efficient in communication     |
| ⑥ A system call is involved in it           | ⑥ No system call is involved          |
| ⑦ Process has Process control block         | ⑦ Thread has Thread control block     |

## Process scheduling

is the activity of process manager that handle the removal of the running processes from CPU and the selection of another process on the basis of a particular strategy.

## Loop back address

A loop back address is distinct reserved IP address 127.0.0.1 to refer to system on which process is running.

what is API?

## Application program Interface (API)

is code that allows two software programs to communicate with each other.

## The principle of least privilege

the principle of least privilege states that a subject should be given only those privileges needed for it to complete its task.

- In UNIX, user is domain. True or False?

- true

## what is kernel

→ is a central component of OS.

that manages operation of computer and hardware

→ acts as a bridge between applications and data processing performed

at hardware level using

IPC and system calls

## Objectives

- To control disk, memory, task management

- To decide state of incoming process

- To establish communication between user level application and hardware

① what is system call ? explain different type of system call.

→ **System call** is a way for programs to interact with operating system.

5 types system call

② Process call :-

- deals with process such as creation & termination

Example:-

(Window)

CreateProcess()

Exit Process()

(Linux)

fork()

exit()

wait()

( )

② File management :-

- responsible for file manipulation  
creating, reading, writing into file

Example:-

(x00)

( )

(window)

• CreateFile()

ReadFile()

WriteFile()

CloseHandle()

(zwlobn)

• open()

read()

write()

close()

(Linux)

• open()

read()

write()

close()

### ③ Device management :-

- responsible for device manipulation
- reading / writing device buffer

Example :-

(windows)	(Linux)
SetConsoleMode()	ioctl()
ReadConsole()	read()

### ④ Information Maintenance

- handle information and its transfer

(Xunis) between OS and user programs

Example :-

(Windows)	(Linux)
SetTimer()	alarm()
sleep()	sleep()

### ⑤ Communication

→ useful for inter process communication

Example :-

(Windows)	(Linux)
CreatePipe()	pipe()
MapViewOfFile()	mmap()

⑧ How does communication take place in client server system? Explain

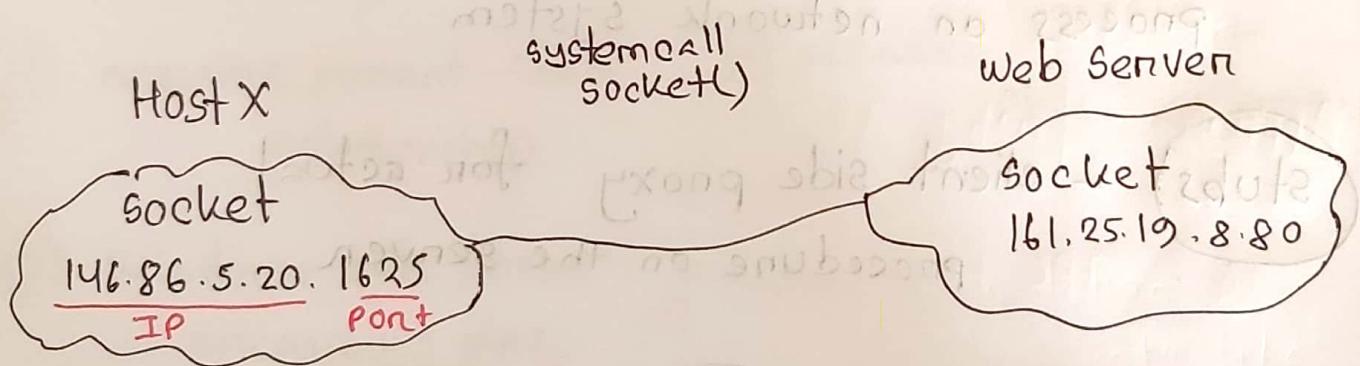
→ Client-server communication has two components client and server. The client sends request to the server and the server responds to the client request.

Three methods of client-server communication

① **socket** → low level communication channel

→ a socket is an endpoint for communication

→ a socket is identified by an IP address and a port number



Two major form

① Connection Oriented / TCP (Transmission Control Protocol)

→ guaranteed that all packets will arrive in good condition at other end and in order receive

② Connectionless (UDP - user data protocol)  
→ no guarantee that packets will delivered  
with undamaged and will receive in  
particular order  
→ UDP ~~for~~ Transmission > TCP

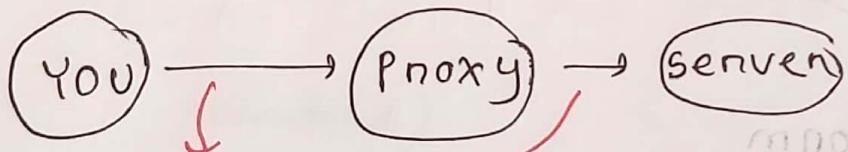
②

## RPC - Remote Procedure Calls

- \* in socket we communicate with process
  - \* In RPC "functions / procedure"
- RPC abstracts procedure calls between processes on network system

Stubs

client side proxy for actual procedure on the server



communicates

### ③ Pipe

- acts as a conduit between two process  
to communicate

#### Ordinary Pipe

- ① ordinary pipe allows communication in standard producer consumer style

- ② unidirectional

- ③ produces write to one end and consumer needs at other end

- ④ requires parent-child relationship

- ⑤ windows calls these anonymous pipe

#### Named pipe

- ① more powerful than ordinary pipe

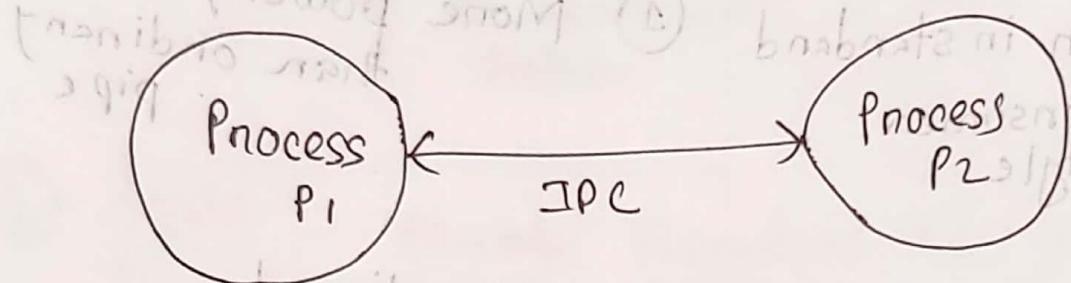
- ② bidirectional

- ④ no need parent child relationship

# IPC - Interprocess Communication

— is the mechanism provided by

OS that allows processes to communicate with each other



two types of process

- ① Independent process
- ② Co-operating process

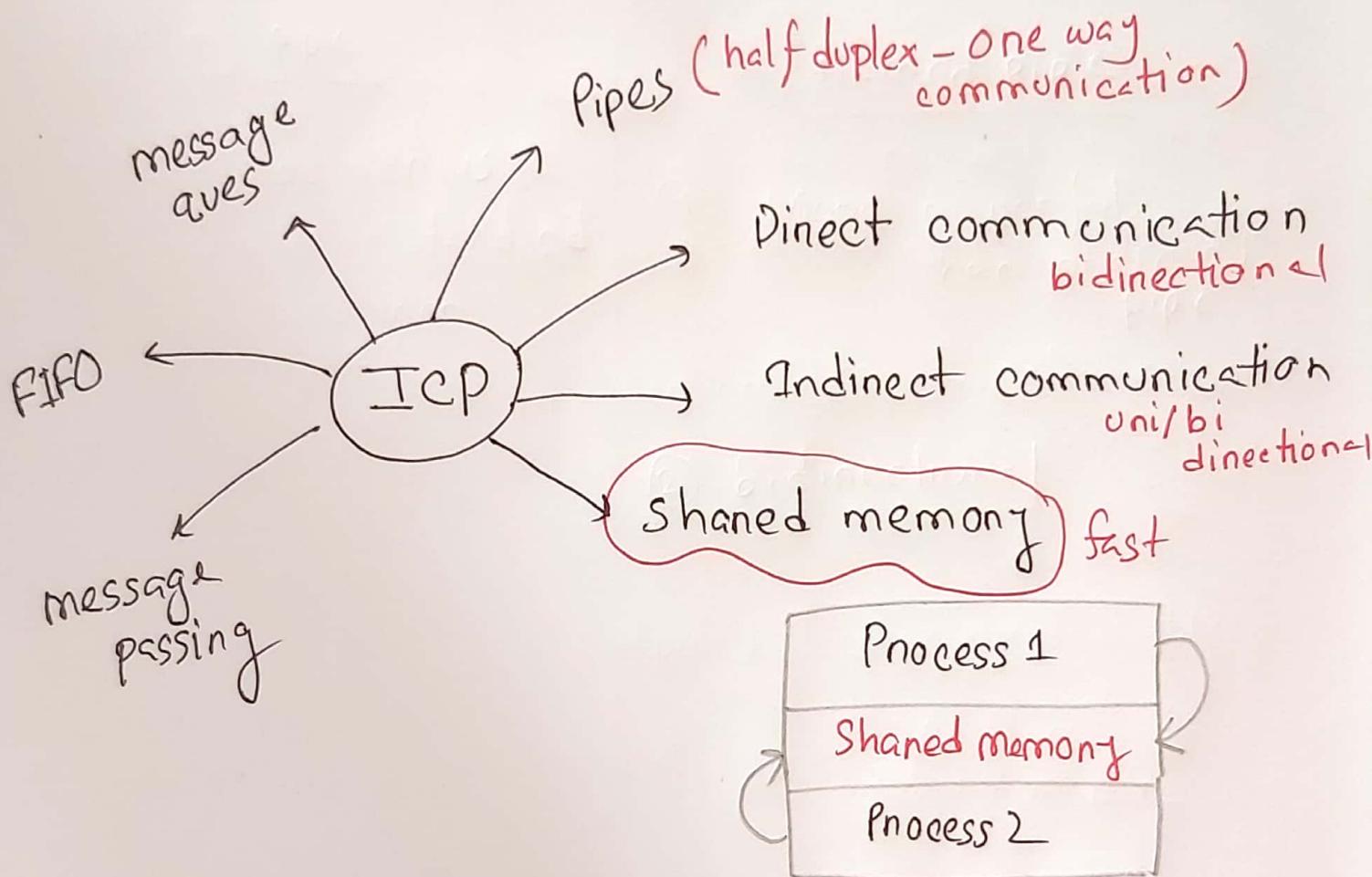
Applic.

why IPC?

IPC helps achieving

- ① Computational speed up
- ② Modularity
- ③ Information & data sharing
- ④ Privilege separation
- ⑤ Process can communicate with each other and synchronize their action

# Approaches for Interprocess Communication



## Message Passing (MP)

- is useful for exchanging **smaller amount of data**
- is **easy** to implement in distributed system than shared memory (SM)
- **slower** than SM

(MP provides a mechanism to allow process to communicate and to synchronize their actions without sharing the same address space.)

∴ MP provides two operations

send (message) receive (message)

Methods for implementing a communication link

and send() / receive() operation

- ① Direct or Indirect communication
- ② Synchronous or asynchronous
- ③ Automatic or Explicit

# send (P, message) — send a message to process P

# receive (id, message) — receive a message from any process.

↳ send and message to individual (id = name of process)

↳ receive

↳ receive the message sent by the process

↳ receiving message from random process

→ position might be available or has information about the process

# send (P, message) — send message to process P

# receive (Q, message) — receive message from process Q



# send (A, message) — send message to mailbox on port A

# receive (A, message) — receive message from Mail box A

need a common share & mailbox between two process



## Indirect communication

messages are send and received from mail box or port.

### Properties of communication link

- (1) Link established if there is a share mail box between process
- (2) link may associated with many processes
- (3) Between two process there may have several link
- (4) Link may be unidirectional or bi-directional

## Direct communication

Date: .....

- ① Process must name each other explicitly

`send(P, message)` → send message to process P

`receive(Q, message)` → receive message from Q process

## communication link property

1. links established automatically
2. link associated exactly one pair of communication processes
3. Between two process there exists exactly one link
4. link is bi-directional

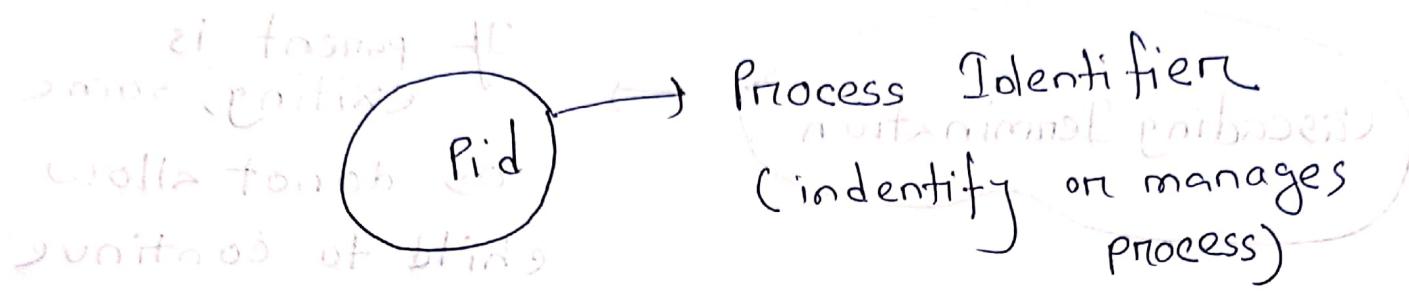
Semaphores refer to the integer variables that are primarily used to solve the critical section problem via combining two of the atomic procedures, wait and signal, for the process synchronization.



**Oricef®**  
ceftriaxone

## Process Creation

Parent process creates child process which in turn creates other processes forming a tree of processes.



## Resource sharing

① Parent & child

share all resources

② Parent & child

share no resource

③ Child share subset

of parent resources

## Execution sharing

① Parent and child

execute concurrently

② parent wait until  
child terminate

③ parent finish +  
child finish

## Process termination

→ process executes last statement and ask OS to delete it (`exit()`)

### Cascading Termination

If parent is exiting, some OS do not allow child to continue if all parents are terminated. So all child are terminated then

For child process

if no parent waiting

the terminated process is

Zombie

if parent terminated

, process are

orphans

## # Two types of scheduling :-

(1) Non-preemptive scheduling :-

once it is allocated in CPU  
it stays in CPU  
until it is terminated by  
CPU or switch to waiting state

(2) Preemptive :-

allows removal of process from CPU

## What is short term scheduler?

→ ~~what is~~ is a scheduler which selects among processes in the ready queue and allocates the CPU to one of them.

In preemptive we can context switch process if burst time process arrived next

when CPU scheduling occurs?

→ when process

① switches from [running] to [waiting] state

② switches from [running] to [ready] queue

③ switches from [waiting] to [ready]

④ Terminates

Non-preemptive

Preemptive

considerations

downgrade priority

no access to shared data

preemption while in kernel mode

interrupts

occurs during crucial OS activities

emit to forums  
about it  
(22 wrong answers)

emit to forums

## What is dispatcher?

→ is a module that gives control of the CPU to process selected by short term scheduler.

This involves activities are

context switching → switch to user mode

jumping in proper location  
in the user to restart

switching

## What are scheduling criteria?

max ① CPU utilization (keep the CPU busy as much as possible)

max ② Throughput (No. of processes that complete their execution per time unit)

min ③ Turnaround time (amount of time to execute a particular process)

min ④ waiting time

min (5) Response time

First come first served (FCFS) scheduling

## FCFS scheduling

→ easy to implement

~~14-287~~

## Execution time

What is burst time

→ it is the time  
taking by a  
process on a  
cpu in one go

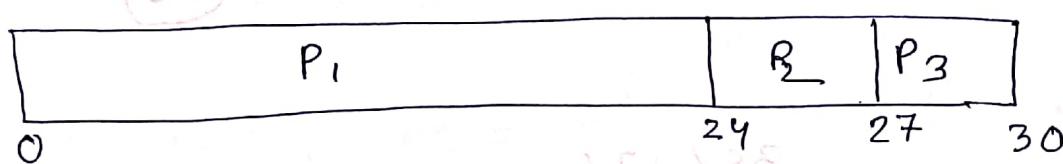
<u>Process</u>	<u>Burst time</u>
P <sub>1</sub>	24
P <sub>2</sub>	3
P <sub>3</sub>	3

∴ arriving process  $\rightarrow$  order

$P_1, P_2, P_3$

pi comes first  
so it will access CPU first

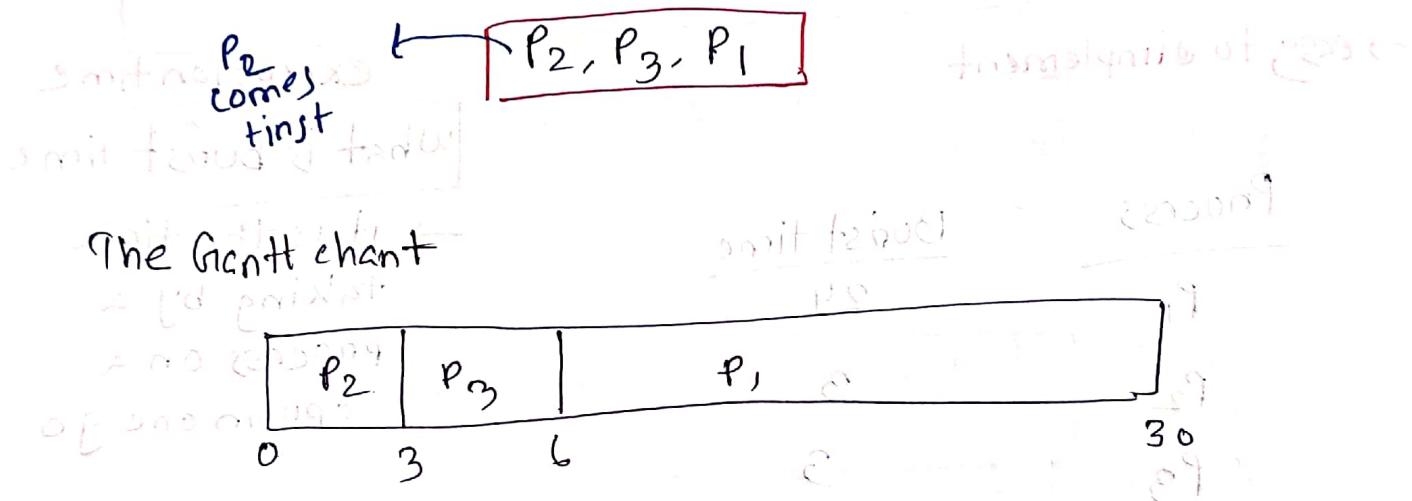
# The Giant chant for scheduling



$$\therefore \text{waiting time} ; \quad P_1 = 0s \\ P_2 = 24s \\ P_3 = 27s$$

$$\therefore \text{average waiting time} = \frac{0+24+27}{3} \\ = 17s$$

# Now if we change the process order



$$\therefore \text{waiting time} = P_1 = 6s \\ P_2 = 0s \\ P_3 = 3s$$

$$\therefore \text{average waiting time} = \frac{0+3+6}{3} \\ = 3s$$

$$3s < 17s$$

this case is better

$\therefore$  if short process comes first, avg waiting time is less.

**convoy effect** :- short process first, long process behind.

con:- FCFS suffers from convoy effect & avg waiting time higher

### Shortest Job first (SJF) scheduling

Upnos

most optimal  
better than FCFS

cons

difficult  
to know  
the length  
of

next CPU request

Process	burst time
P <sub>1</sub>	6
P <sub>2</sub>	8
P <sub>3</sub>	7
P <sub>4</sub>	9

non-preemptive  
Scheduling

# SJF scheduling chart

P <sub>4</sub>	P <sub>1</sub>	P <sub>3</sub>	P <sub>2</sub>
0	3	9	16

Sequence of execution: P<sub>4</sub>, P<sub>1</sub>, P<sub>3</sub>, P<sub>1</sub>

$$\therefore \text{avg. waiting time} = 0 + 3 + 5 + 16 / 4 = 7.5$$

shortest remaining time first

Process	Arrival	Completion time	Burst time
P <sub>1</sub>	0	4	4
P <sub>2</sub>	2	6	5
P <sub>3</sub>	3	12	5
P <sub>4</sub>	5	10	3

Preemptive scheduling

Hence P<sub>1</sub> comes first (as arrival time)

after 1s P<sub>2</sub> comes. In that time P<sub>1</sub> has done 1s time. Now P<sub>1</sub> burst time (8-1) 7 > P<sub>2</sub> burst time. 4

short burst time process first

∴ P<sub>2</sub> will context switch P<sub>1</sub>.

∴ P<sub>1</sub> have to wait until P<sub>2</sub> done

$$ef - rf + et + os = \text{unit turnaround}$$

after  $\frac{2s}{3}$   $P_3$  comes  
 $P_3$  burst time ( $\theta$ )  $> P_1$  burst time inscheduling ( $4-1)/3$   
 $P_2$  burst time ( $4-1)/3$

$\therefore P_2$  will continue.  $P_1$  have to wait still. then  $P_3$  at  $P_3$  will start cz high burst time last

after  $\frac{3s}{4}$   $P_4$  comes; (5)  
 $P_3$  burst time  $> P_1$  burst time  $> P_4$  burst time (inscheduling)  $> P_2$  burst time ( $3-1)/2$   
 $(9)$  (Optimizing feasible) stopping  $P_1$  to allume

$\therefore$  now  $P_2$  will still continue.

$\therefore$  then  $P_4$  will start cz ( $P_4 < P_1$  burst time)

$\therefore$  after  $P_4$ , remaining  $P_1$  burst time (Optimizing feasible) (finest)

at last  $P_3$  starts (high burst time)

$$\text{avg. waiting time} = \frac{(10-1) + (1-1) + (17-2)}{4}$$

$\Rightarrow$  avg. waiting time =  $\frac{10+0+15}{4} = 6.5$

avg. waiting time =  $6.5 \text{ sec}$

so final answer is

avg. waiting time =  $6.5 \text{ sec}$

less waiting time  
less complex

## Priority scheduling

PS

→ Priority number is an associated with each process (Integer)

→ smallest integer (highest priority)

### Problem of PS

(con)

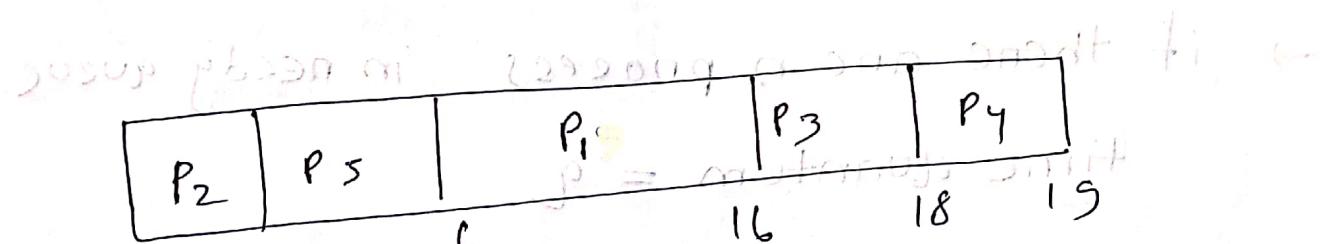
(starvation)

→ low priority problem may never execute

Sol :- increase priority as time progresses of process

<u>Process</u>	<u>Burst time</u>	<u>Priority</u>	<u>Given</u>
P <sub>1</sub>	10	3	
P <sub>2</sub>	1	1	
P <sub>3</sub>	2	2	Using
P <sub>4</sub>	1	4	Non
P <sub>5</sub>	5	5	Preemptive
		1	(No context
		5	switch,
		2	Wait until
			a process
			finishes)

Grant chart



Priority goes to P1 & P3 because P1 has lowest priority

$$\text{avg. waiting time} = \frac{6 + 0 + (6 + 18 + 1)}{4+1}$$

$$= 8.2$$

Time taken for 21 unit execution

$\lceil \frac{21}{5} \rceil$  units

Time taken for 21 unit execution

Waiting time for each process =  $\frac{\text{Time taken}}{\text{Burst time}}$

## Round Robin | RR

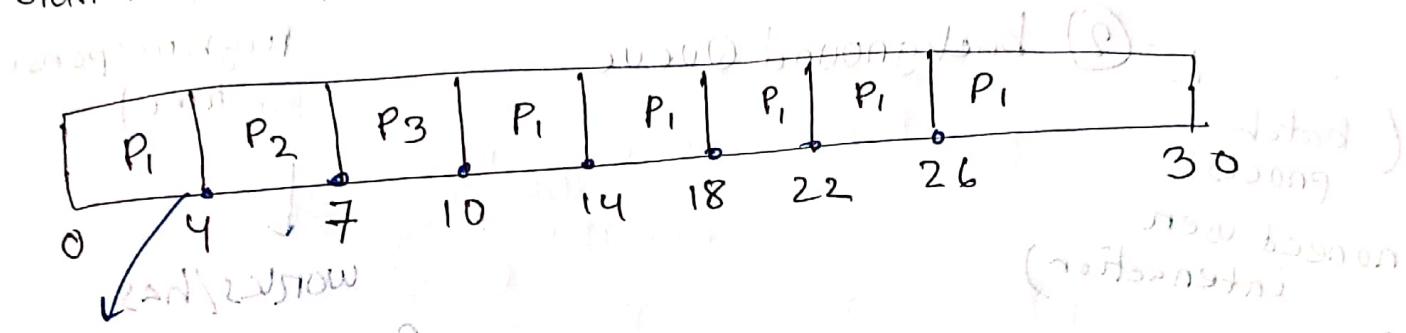
- Each process gets a small unit of CPU with time (time quantum, q) (10-100 milliseconds)
- After this time passed, process will stop and moved to end of ready queue / added
- if there are  $n$  processes in ready queue.
  - ∴ time quantum =  $q$
  - ∴ each process gets  $\frac{1}{n}$  of CPU time in chunk at most  $q$  times
  - ∴ each process gets  $q$  unit of CPU time at once
  - ∴ waiting time is not more than  $\lceil (n-1) \times q \rceil$

cons:- spend more time context switching

<u>Process</u>	<u>Burst time</u>	<u>Time quantum</u>
P <sub>1</sub>	24	q=4
P <sub>2</sub>	3	Hence, 11
P <sub>3</sub>	3	

Pneemptive

Gantt chart is:



Although higher avg. time (30)  
more context switches  
but faster response time.

∴ a should be  $>$  context switch time.  
(but not too large)

if it becomes FcFS

18 tiny

## Multi level Queue

# Ready Queue is partitioned into two queue

① foreground Queue (interactive process which have time)

② background Queue.

(batch process  
noneed user interaction)

(FCFS algorithm)

high response time

works/has,

Round Robin algorithm

(better response time)

# scheduling between two queue nodes

① fixed priority scheduling

② Time Slice

80% to foreground in RR

20% to background in FCFS

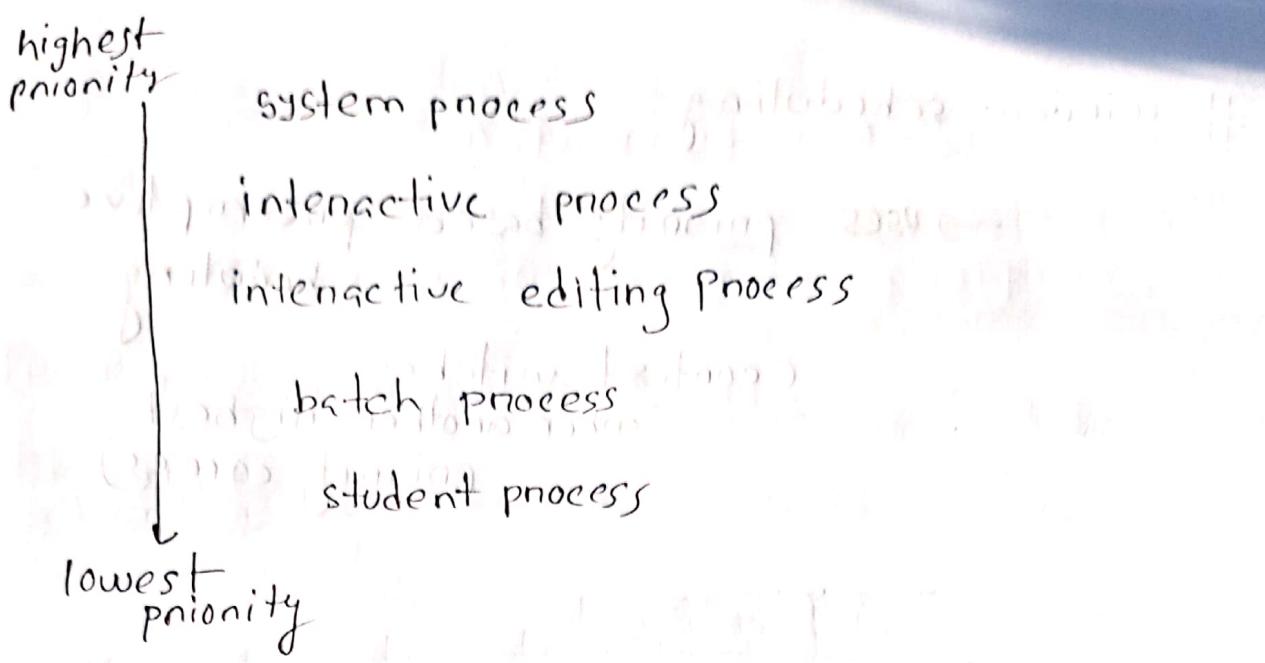
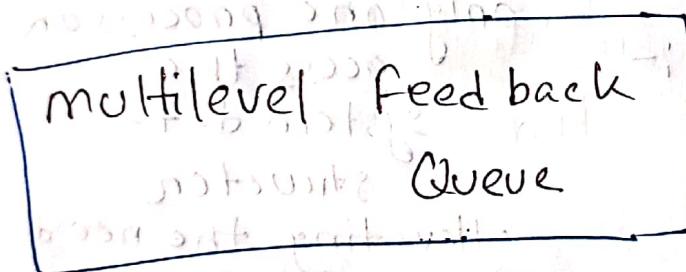


Fig:- Multilevel Queue



→ three Queue.

①  $Q_0$  — RR with time quantum = 8

②  $Q_1$  — RR if situation = 16

③  $Q_2$  — FCFS

## # window scheduling

→ uses priority based preemptive scheduling

(context switches when another highest priority comes)

## Multiple Processor Scheduling

- ① Asymmetric multiprocessor :-
  - only one processor access the shared system data structures alleviating the need of kernel data sharing
- ② Symmetric :- each processor is self scheduling
  - each has its own private area of memory

## what is time quantum?

- a preemptive scheduler will allow a process to run for a short amount of time. this is called time quantum.  
lower time quantum → more context switching → less performance

## What is preemptive scheduling?

- PS is when a process transitions from running state to ready state or from waiting state to ready state.
- it runs a task with higher priority before another lower priority task, even if the priority task is still in state critical state.
- better CPU utilization
- more less waiting at
- less waiting & response time
- RR, shortest remaining time first

what is non-preemptive?

→ Non preemptive when , once a process starts its execution in CPU, it must finish it executing other, it cannot be paused in middle (but if preemptive it can be paused)

- less optimization, reduction of state transitions
  - more personnel waiting time, patients

$\rightarrow$  SJF using FCFS

- CPU is allocated to process until it terminates or switches to waiting state

In preemptive scheduling, CPU is allocated

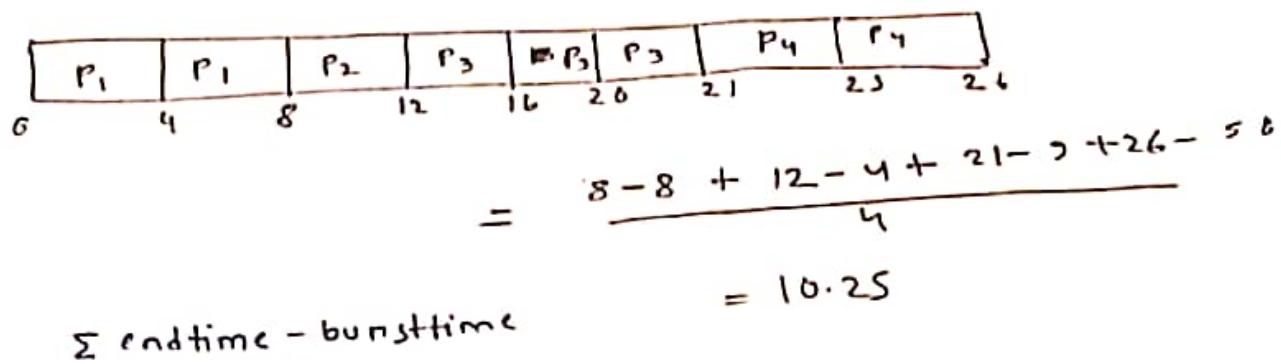
to process for short

amount of time]

## what is cpu scheduling

→ CPU scheduling is a process of determining which process will own CPU for execution while another process is on hold.

Q) Time quantum  $q = 4$



- Just follow this
- \* Turnaround time = Exit time - Arrival time
  - \* Waiting time = Turnaround time - Burst time