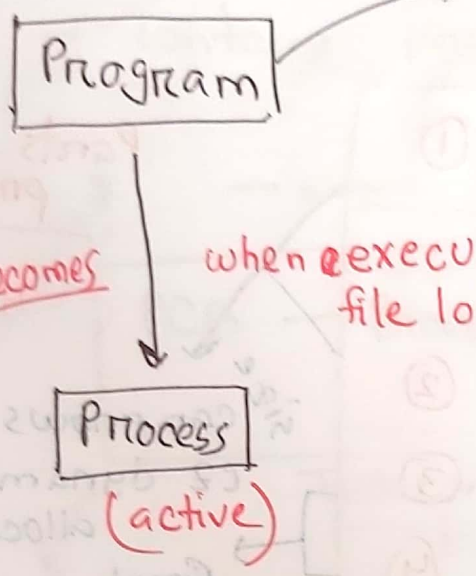
Chapter-3

Process



(when program starts execution in RAM it becomes process)

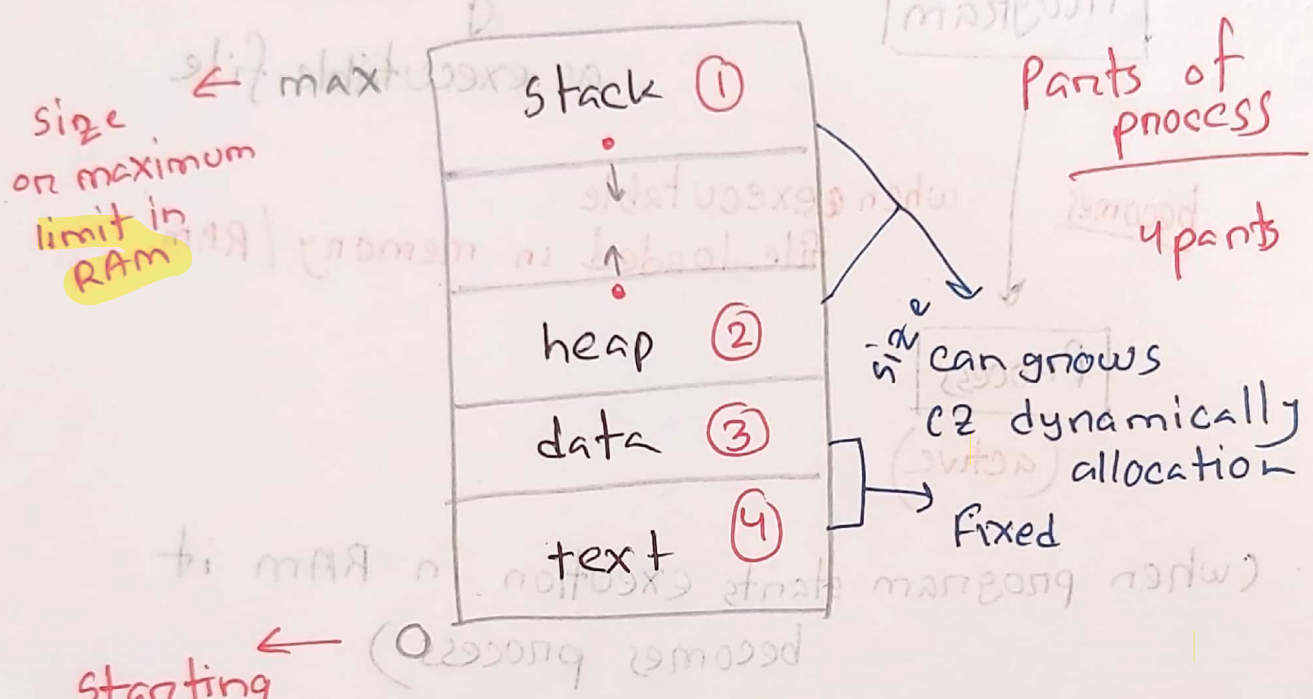
* To start a program you must bring it to ram

* program consists some instruction

* one program can have several process

Process

→ Process is a program in execution



(How does process look likes in memory / RAM)

① stack → ~~contains program code~~

→ contains tempory data

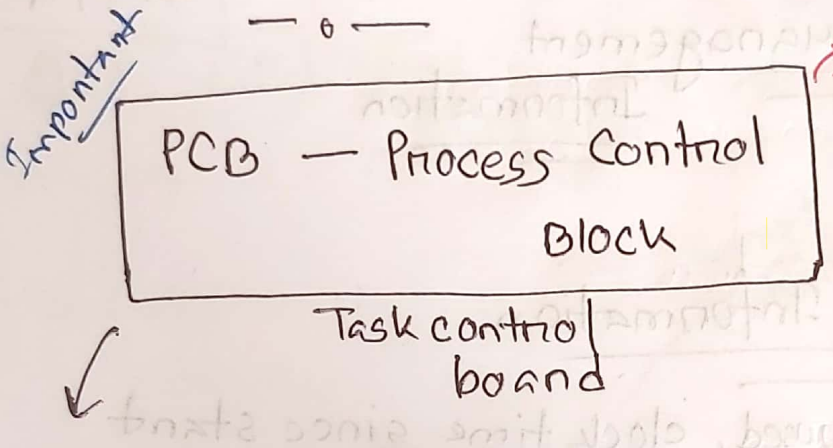
→ function parametens,
return addresses
Local variables

↓
inside in function

② Data Section → contain global vanibles

③ Heap → contain memory dynamically allocated during run time

④ Text → contains program code



information associated with each process in the OS

① Process state :

i — (new) ii — (running) iii — (waiting)

iv - (ready) v - (terminated)

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

(3) CPU Registers :- content of all process centric registers

(4) CPU scheduling Information

Priorities, scheduling pointers

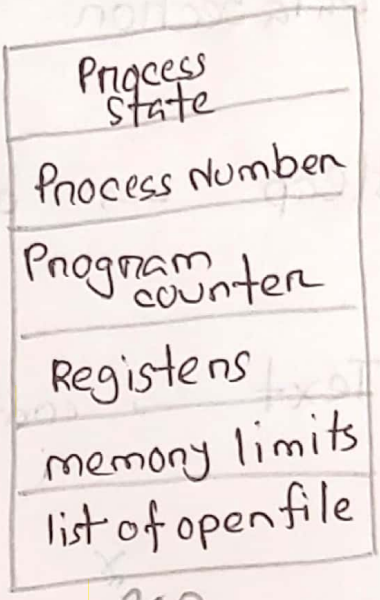
(5) Memory management Information

(6) Accounting Information

cpu used, clock time since start

(7) I/O status Information

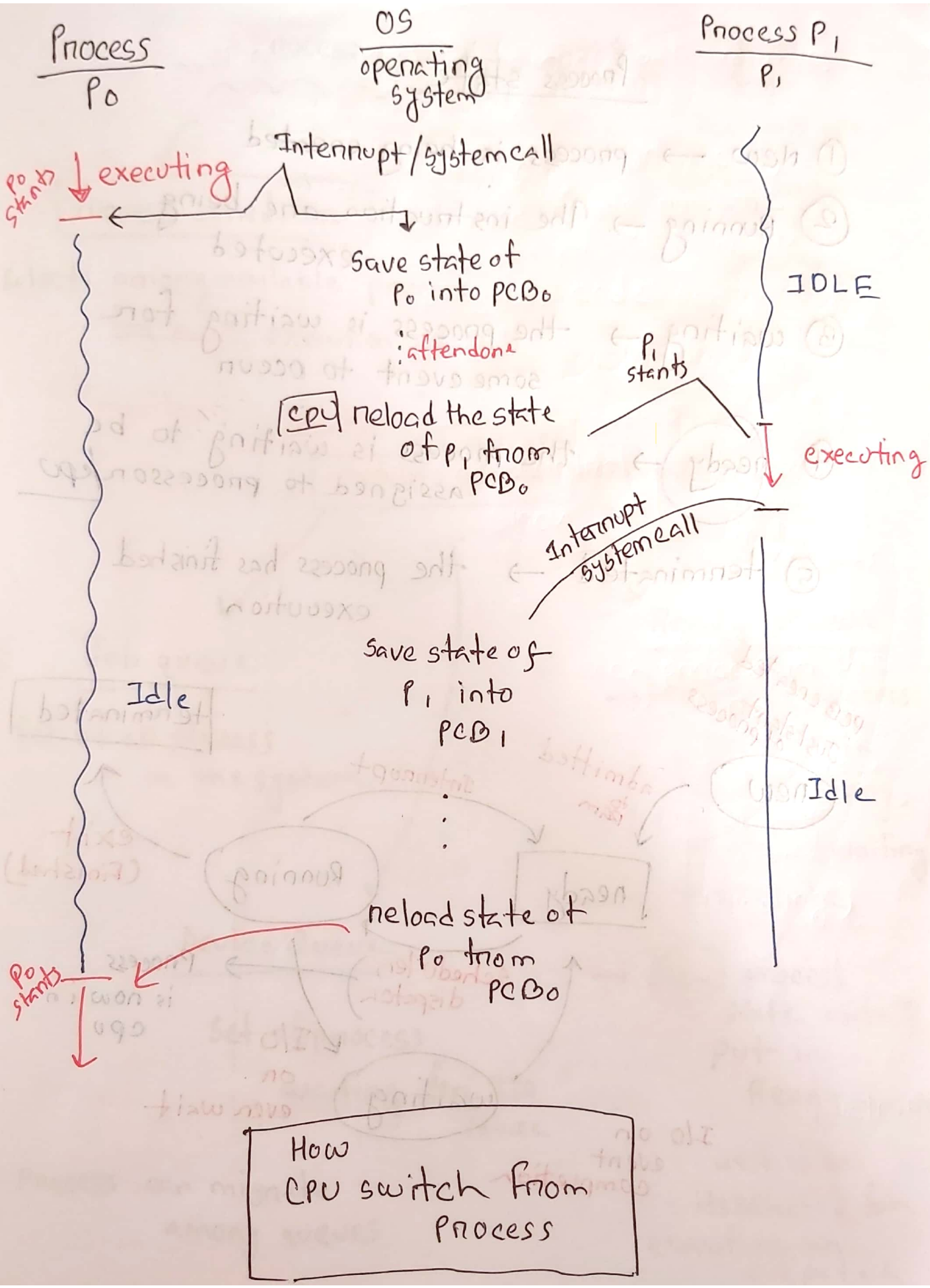
IO device allocated to process, list of open files



—o—

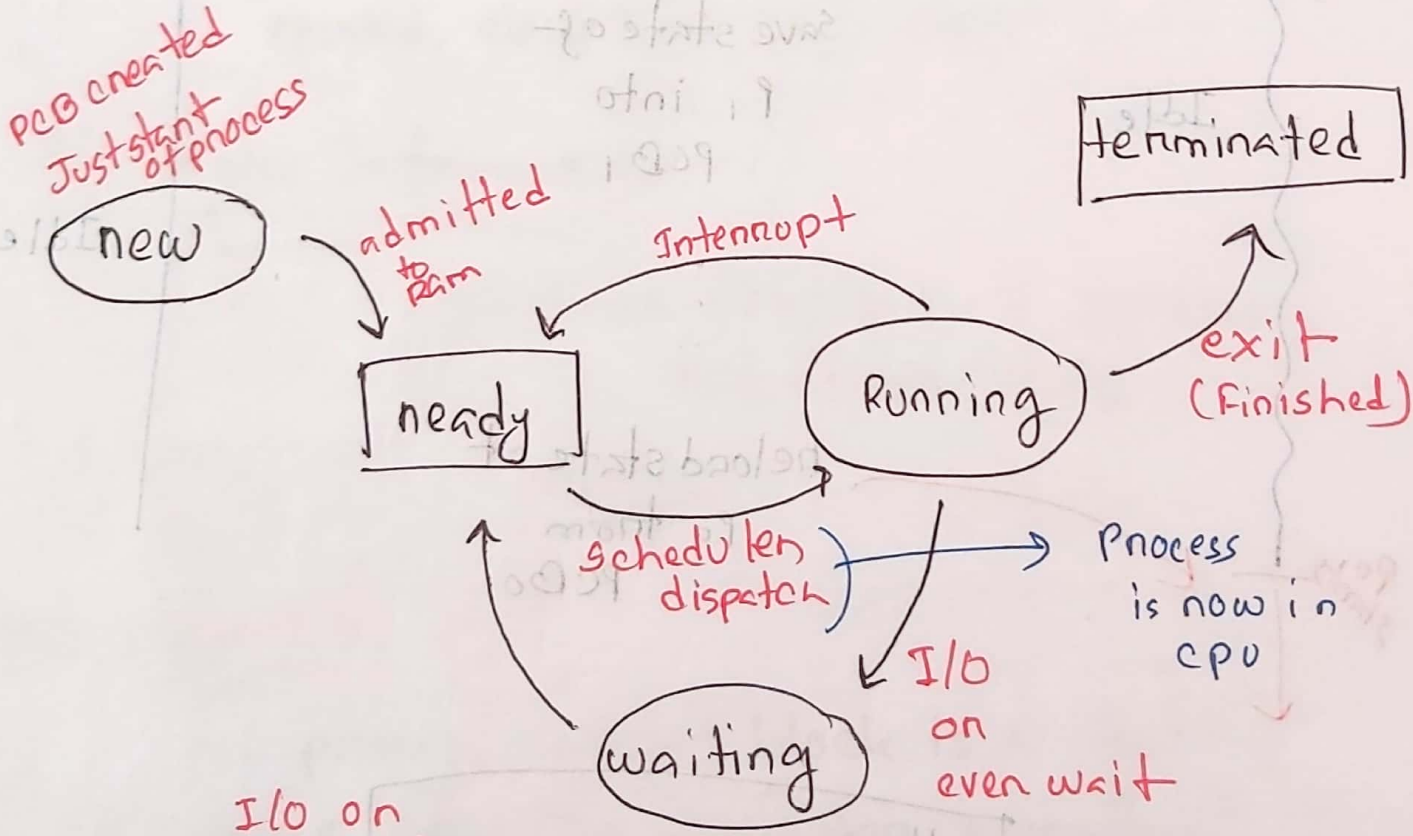
PCB definition

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


Process
P0
OS
operating
system
Process P1
P1
Internupt/systemcall
P0 starts
executing
Save state of
P0 into PCB0
attendon
IDLE
P1
starts
CPU reload the state
of P1 from
PCB0
executing
Internupt
systemcall
Save state of
P1 into
PCB1
Idle
Idle
reload state of
P0 from
PCB0
P0
starts
How
CPU switch from
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

maximiz cpu use
quickly switch cpu for time shaning

Process scheduler

selects among available process for next execution on cpu

↳ maintains

scheduling queues of processes

Job queue

Set of all process in the system

Ready Queue

→ set of all process residing in main memory, ready and waiting to execute

Device Queue

Set of process waiting for I/O device

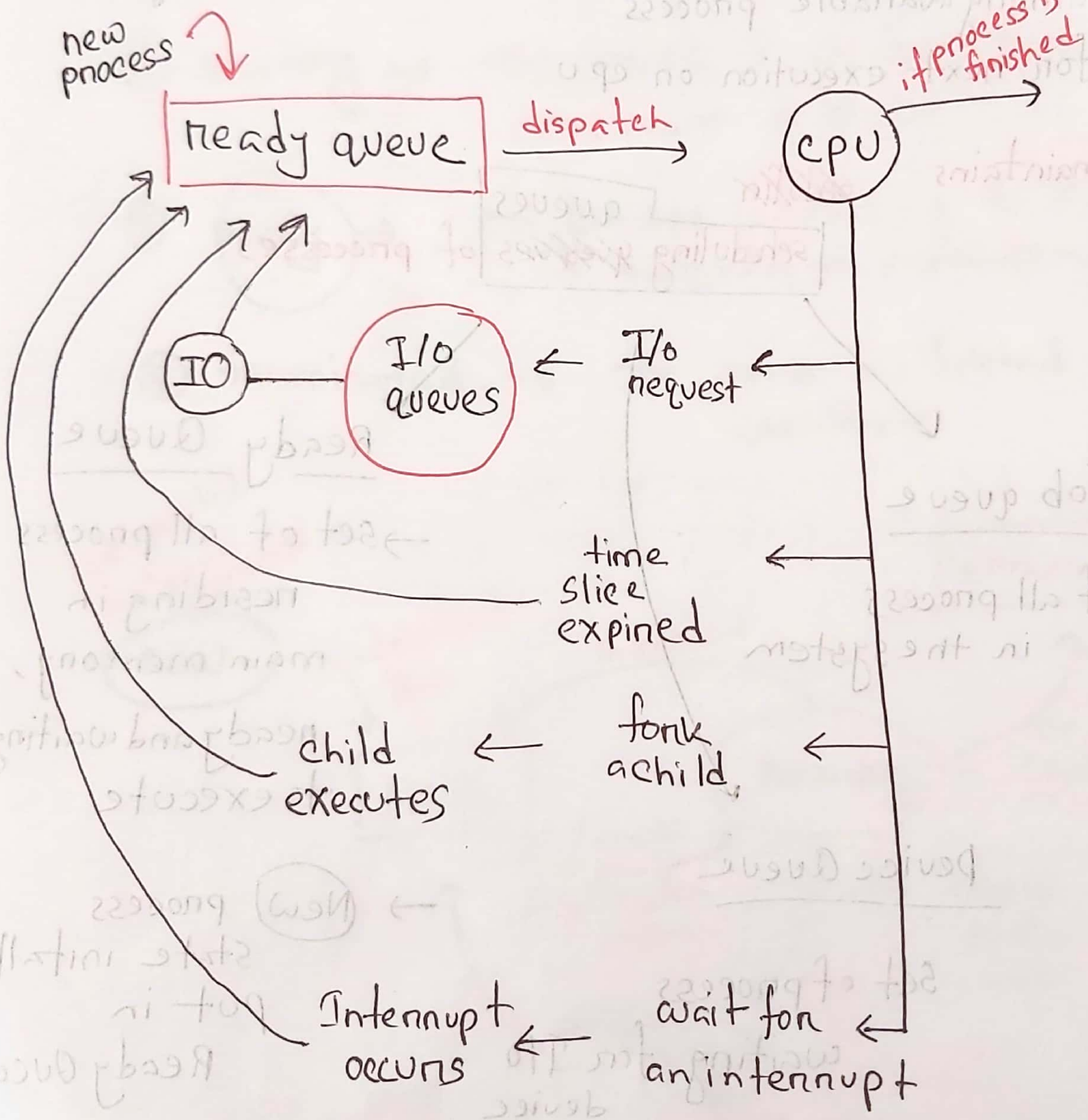
→ (New) process state init all put in Ready Queue

wait until its selected for execution on dispatch

* Process can mignate among queues

Representation of Process scheduling

Queueing diagram



Scheduler

Short term scheduler

→ CPU scheduler

→ which sends a process to CPU

→ select which process should be executed next and allocates to CPU

→ must be very fast
(is invoked / called frequently)
(miliseconds)

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 in-frequently
(seconds, minutes)

→ controls the degree of multiprogramming

definition

how many program should allow at same time

- on admitted to main memory

Types of process

① I/O bound Processs :-

— spend more time doing I/O than computations

— many short cpu ~~bursts~~ bursts

for how long you using the cpu

② cpu-bound process

— spend more time doing computations

— few very long cpu ~~bursts~~ bursts

— o —

Mediom-term-scheduler :

— remove process from memory

(swaping) — store in disk

— bring back in RAM to continue execution

Context switch

definition important

② the system must save the state of old process and load the saved state for new process via context switch

① when a cpu switches to another process it is called context switch

(context of process represented in PCB)

while contex switch system does not do useful work

Definition :- ① + ②

—— o ——

Process Creation

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

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

① 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

— o —

Based no Interprocess communication

Process type

① Independent process
(not affected by other process)

② Cooperating process
(affected by another process)

Sockets

is defined as an end point for communication

(socket parts)
address
- Ipaddress: 161.25.19.8
- Port: 1625

∴ socket = 161.25.19.8.1625

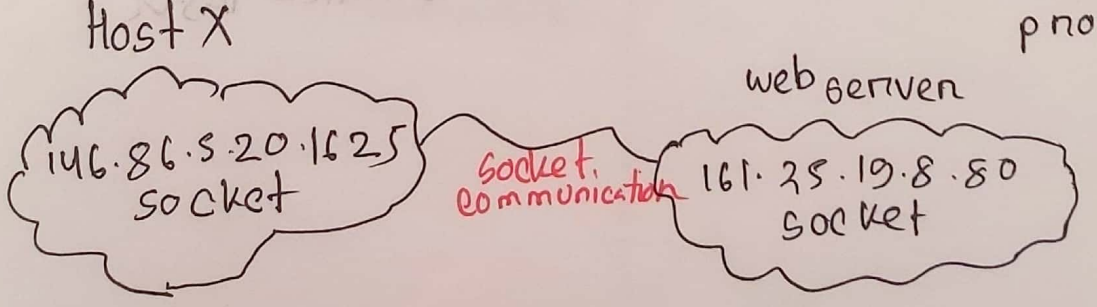
address of each computer

one device has same ip but diffenent port (port change based on tab)

communication consists between a pair of sockets

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

special Ip address: 127.0.0.1 (to refer to system on which process is running)



Process vs Job vs Task

① **Process** :- Process is a program under execution.

Process vs Program

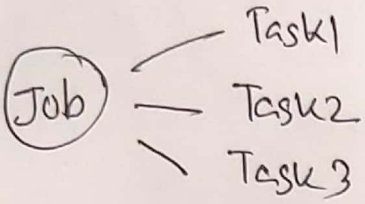
- ∴ a program can be defined as a set of instructions.
- Program is a passiven entity
  Process is a active entity
- Program became process when it comes to RAM
  - Program stays on Harddisk
  - process stay Harddisk, or CPU, main memony (RAM)
- 1 program can have multiple process

② **Task** :-
- Task is a unit of work being executed
- Task in os may be synonyous with process
- 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 senden block untill the message is received	→ Non blocking send has the senden send the message and continue
→ Blocking ~~recio~~ receive has the recieven block untill a message is available	→ Non blocking recive has the reciven receive a vaild message or null

(*) Process Altenate between
CPU bnusts and I/O bnusts

(*) CPU scheduler:-

when cpu becomes idle, it is the Job of cpu schedulen/shont tenm Schedulen to select anothen process from the neady queue to run next on cpu.

(*) Schedulen dispatchen :- is a module that gives the control of CPU to the process selected by the schedulen

Types of socket

actually 3 types

1. Data gram socket (connectionless)
2. Stream socket (connection-oriented)
3. Raw socket
4. Sequenced packet socket

multi cast socket

access → Protocol

UDP (UPD)
User Datagram

(TCP)
Transfer control protocol

(ICMP)
Internet control message protocol

(IDP)
Internet data protocol

Degree of multiprogramming

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

Process	Thread
① Process mean 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 managen that handle the removal of the running process from cpu and the seleection of another process on the basis of a panticulan strategy.

Loop back addness

A loop back addness is distinct neserved ip addness 127.0.0.1 tonefen to system on which process is nonning

what is api?

Application program Intenface (API) is code that allows two softwane prognams to communicate with each other

The principal of Least privilege

~~the principal 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 handware

→ acts a bridge between applications and data processing performed at handware 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 handware

(x) 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

(1) Process call :-

- deals with process such as process creation & termination

Example :-

(Window)
CreateProcess()
Exit Process()

(Linux)
fork()
exit()
wait()

(2) File management :-

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

Example :-

(window)
* createFile()
ReadFile()
writeFile()
CloseHandle()

(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 Maintenace

- handle information and its tranfer between os and user program

Example :-

(windows)	(Linux)
SetTimer()	alarm()
sleep()	sleep()

⑤ Communication

→ useful for interprocess communication

Example :-

(windows)	(Linux)
CreatePipe()	pipe()
MapViewOfFile()	mmap()

(x) How does communication take place in client server system? Explain

→

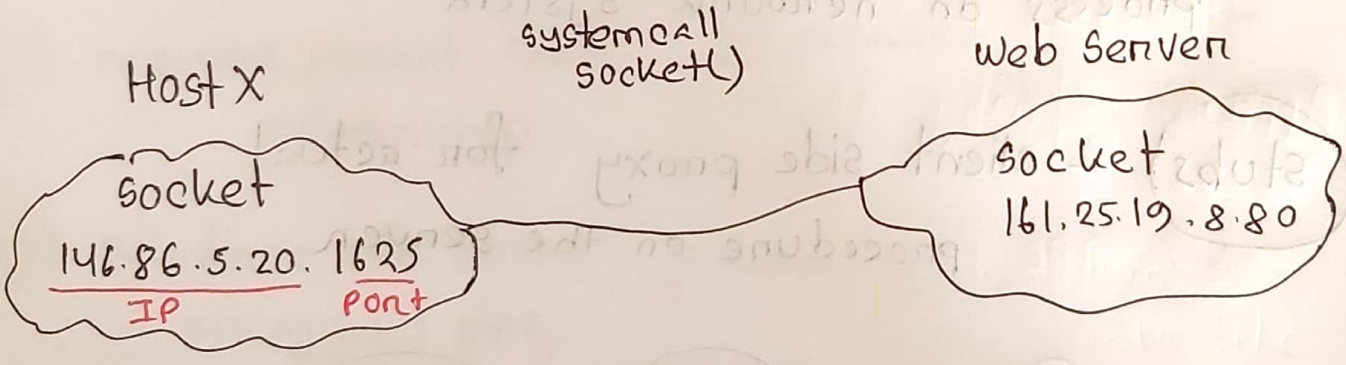
Client-server communication has two components client and server. The client send request to the server and the server responds to the client request

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)

→ guranted that all packets will arrive in good condition at other end and in order recive

② Connectionless (UDP - user data protocol)

→ no guarantee that packets will delivered with undamaged and will receive in particular order

→ UDP ~~for~~ Transmission > TCP
faster

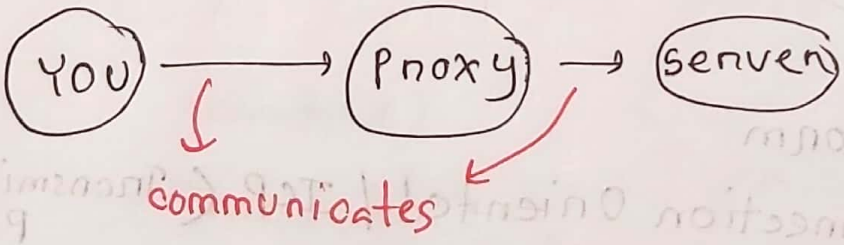
② RPC - Remote Procedure Calls

Function

* in socket we communicate with process
* In RPC " " " Functions / Procedure

RPC abstracts procedure calls between process on network system

stubs - client side proxy for actual procedure on the server



③ Pipe

- acts as a condult between two process to communiate

Ondinary Pipe

① ondinary pipe allows communication in standand producen consumen style

② unidinectional

③ Producens write to one end and consumen needs at othen end

④ nequine panent-child nelationship

⑤ windows calls these anonymous pipe

Named pipe

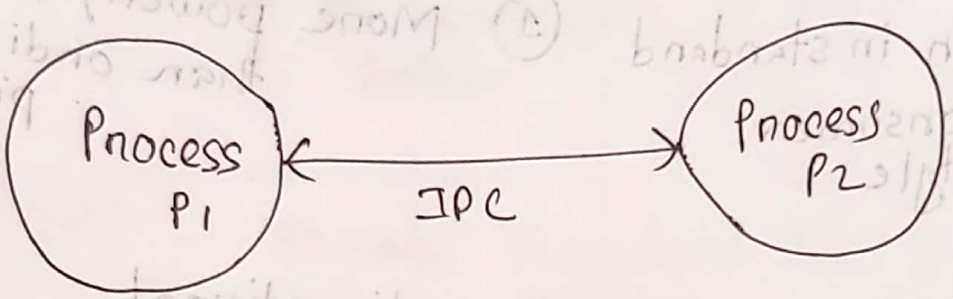
① Mone powenful than ondinary pipe

② bidinectional

④ Noneed panent chid nelationship

IPC – Interprocess Communication

– is the mechanism provided by OS that allows processes to communicate with each other



two types of process

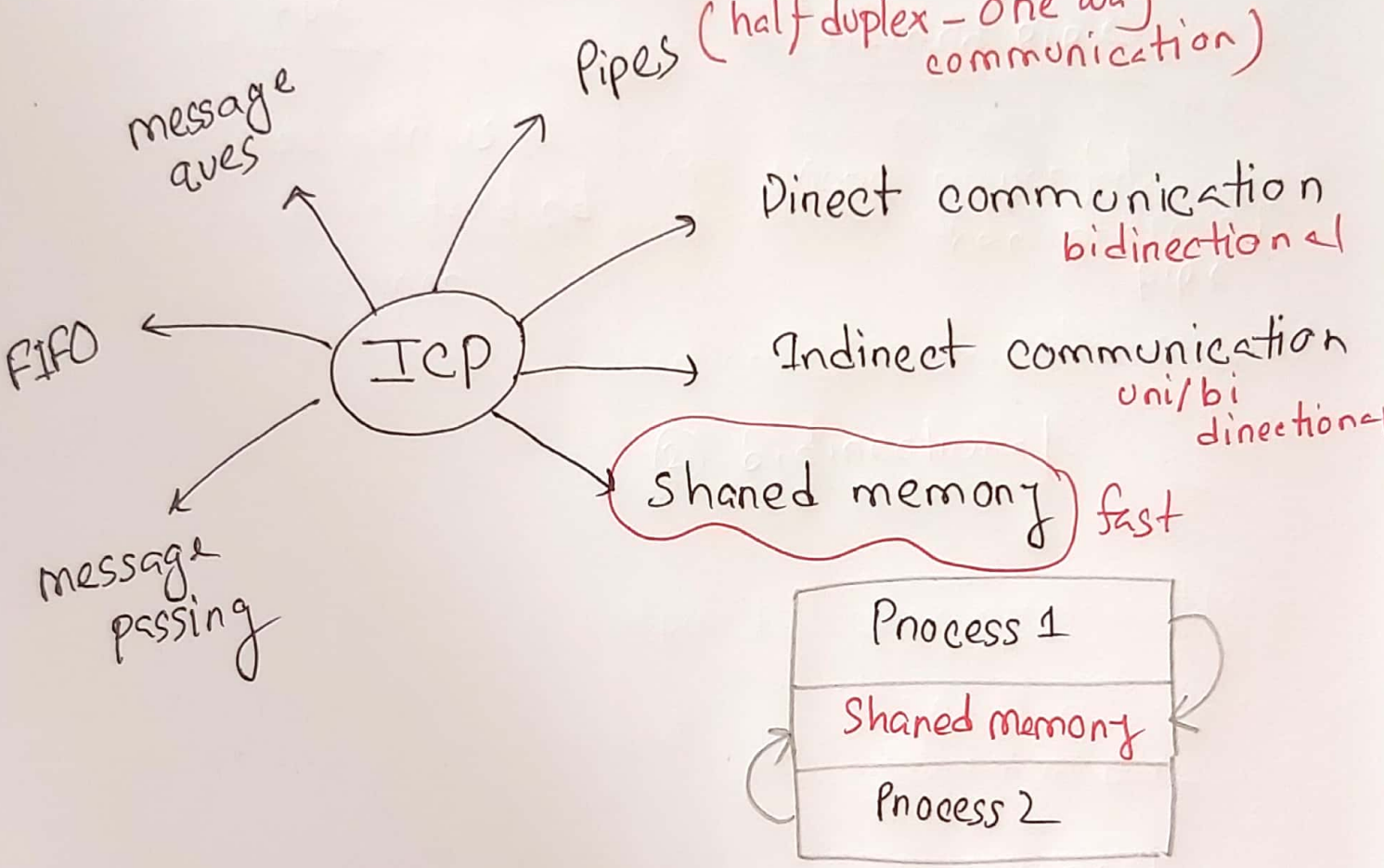
1. Independent process
2. Co-operating process

why IPC?

IPC helps achieving

1. Computional speed up
2. Modulanity
3. Infonmation & data shaning
4. Pnivilege sepenation
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

3 methods

1. Direct or Indirect communication
2. Synchronous or asynchronous "
3. Automatic or Explict "

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

Receive (id, message) — receive a message from any process. (id = name of process)

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

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

↑ Direct Communication

send (A, message) — send message to mail box on port A

receive (A, message) — receive message from mail box A

need a common shared mailbox between two process

Indirect Communication

Indirect communication

- messages are send and received from mail box or port

(

Properties of communication link

(1) Link established if there is a shane mail box between process

(2) link may associated with many processes

(3) Between two process there may have several link

(4) Link may be uni-directional or bi-directional

Direct communication

Date :

(*) Process must name each other explicitly

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

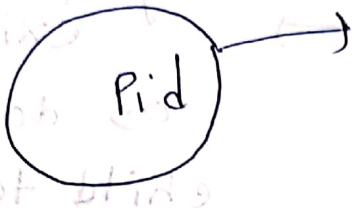
receive (Q , message) → receive message from Q process

communication link property

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

Process Creation

Parent process create child proces which in turn create other process framing a tree of process



Process Identifier
(indentify or manages process)

Resource sharing

1. Parent & child share all resource
2. Parent & child share no resource
3. Child share subset of parent resource

Execution sharing

1. Parent and child executes cocurrently
2. parent wait until child terminate

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 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 is a process is alloctacted in cpu it stays in cpu untill it is terminated by cpu on switch to waiting state

② preemptive :-

allows removal of process from cpu

what is short term scheduler ?

→ ~~when cp~~ is a scheduler which selects among process in the ready queue and allocates the cpu to one of them.

In preemitive 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 → access to shared data

→ preemption while in kernel mode

→ interupts occurs during crucial OS activities

What is dispatcher?

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

This involves activites are
- → context switching
- → switch to user mode
- → jumping in proper location in the user to restart

what are scheduling critenia?

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

max (2) Throughput (No. of process 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

—— o ——

First come First served (FCFS) scheduling

FCFS scheduling

→ easy to implement

execution time

what is burst time

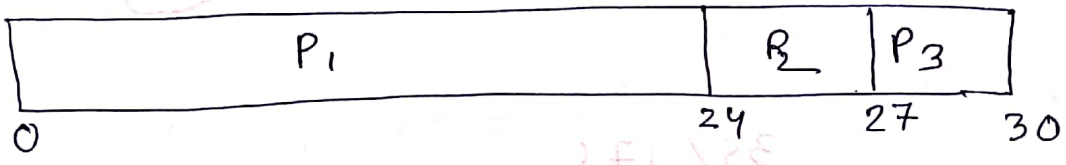
→ it is the time taking by a process on a cpu in one go

Process	Burst time
P_1	24
P_2	3
P_3	3

∴ arriving process order : P_1, P_2, P_3

P_1 comes first it will access cpu first

The Gantt chant for scheduling



∴ waiting time ; $P_1 = 0s$
$P_2 = 24s$
$P_3 = 27s$

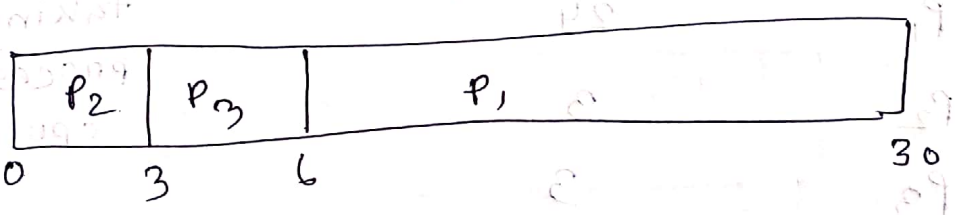
∴ avenage waiting time = $\frac{0+24+27}{3}$

= 17s

Now if we change the process onden

P_2, P_3, P_1 ← P_2 comes tinst

The Gantt chant



∴ waiting time = $P_1 = 6$
$P_2 = 0$
$P_3 = 3$

∴ avenage waiting time = $\frac{0+3+6}{3}$

= 3s

3s < 17s
this case is betten

∴ 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

— o —

Shortest Job first (SJF) scheduling

↓ pros

most optimal
better than FCFS

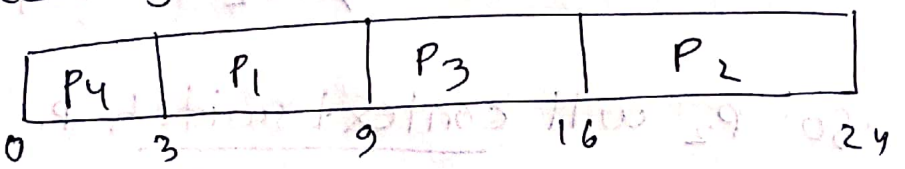
↓ cons

difficult to know the length of ~~next~~ next cpu request

Process	burst time
P_1	6
P_2	8
P_3	7
P_4	3 → shortest first

non-preemptive scheduling

SJF scheduling chant



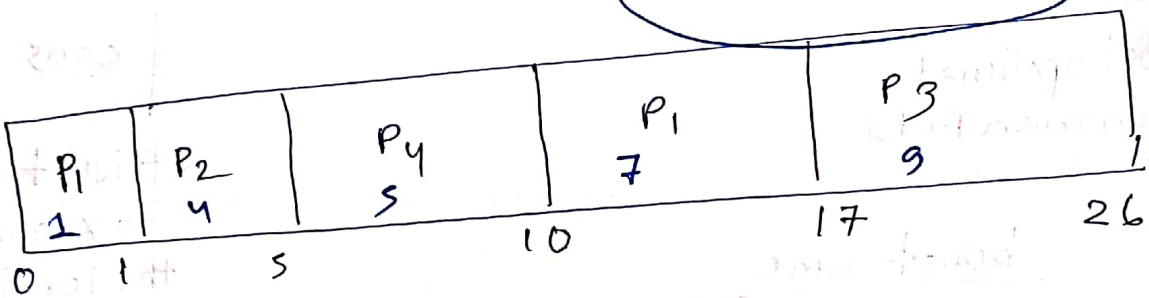
P4, P1, P3, P1

∴ avg. waiting time = 0 + 3 + 9 + 16 / 4 = 7.75

shortest remaining time first

Process	arrival time	Burst time
P_1	0	8 (1+7)
P_2	1	4
P_3	2	9
P_4	3	5

Preemtive scheduling



∴ Here P_1 comes first (as arrival time)

∴ after 1s P_2 comes ∴ In that time P_1 has done 1s time. Now P_1 burst time (8-1) 7 > P_2 burst time.

∴ So P_2 will context switch P_1.

∴ P_1 have to wait untill P_2 done

short burst time process first

after 2s P_3 comes;

P_3 burst time (9) > P_1 burst time (7) > P_2 burst time (4-1)/3

inscheduling

∴ P_2 will contine. P_1 have to wait still. then P_3 at last will start cz high burst time

after 3s P_4 comes;

P_3 bursttime (9) > P_1 bursttime (7) > P_4 bursttime (5) > P_2 burst time (3-1)/2

∴ NOW P_2 will still continue.

∴ then P_4 will start cz (P_4 < P_1 bursttime)

∴ after P_4, remaining P_1 start

at last P_3 starts (high burst time)

$$\therefore \text{avg. waiting time} = \frac{\overbrace{(10-1)}^{P_1} + \overbrace{(1-1)}^{P_2} + \overbrace{(17-2)}^{P_3} + \overbrace{(5-3)}^{P_4}}{4}$$

$$= 6.5_s$$

---

less waiting time
less complex

Priority scheduling PS

→ Priority number is associated with each process (Integer)

→ smallest integer (higest priority)

Problem of PS

cons

(Starvation) → low priority problem may never execute

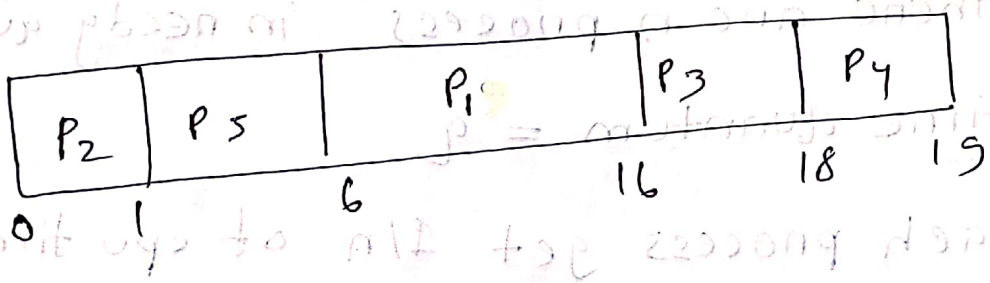
(aging) Sol:- increase priority as time progress of process

Process	Burst time	Priority	Given
P_1	10	3	
P_2	1	1	
P_3	2	4	
P_4	1	5	
P_5	5	2	

lowest priority first

Using Non Preemptive (No context switch, wait until a process finish)

Gantt chart



$$\therefore \text{avg. waiting time} = \frac{6+0+16+18+1}{4+1}$$

$$= 8.2$$

Round Robin | RR

→ Each process gets a small unit of cpu time (time quantum, q) (10-100 miliseconds)

∴ After this time passed, process will stop and moved to end of ready queue /added

→ if there are n process in ready queue.

∴ time quantum = q

∴ each process get 1/n of cpu time in chunk at most q times unit at once

∴ waiting time is not more than $(n-1) \times q$

cons :- spend more time context switching

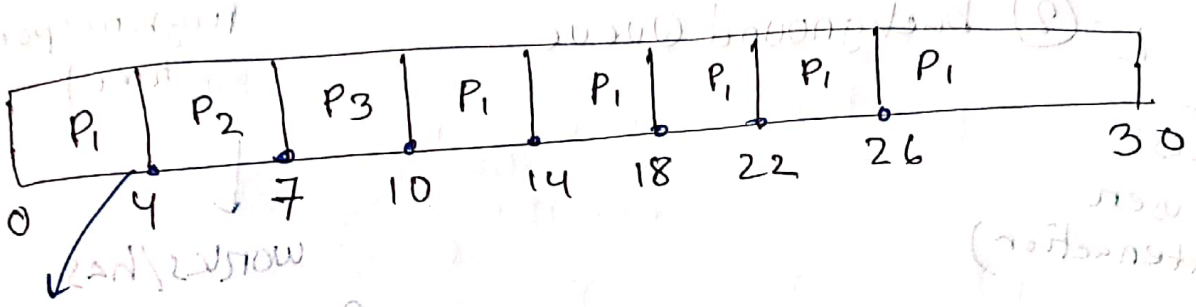
Process	Burst time
P_1	24
P_2	3
P_3	3

time quantum $q = 4$

Here,

Preemptive

Gantt chant is:



context switching

∴ Althrooght, higher avg. time

∴ but faster response time.

∴ q should be > context swtich time.

(but not too large)

↓

then it becoms FcFS

18 Final

Multilevel Queue

Ready Queue is ~~partitioned~~ partitioned into two queue

① foreground Queue (interactive process which have high response time)

② background Queue.

(batch process no need user interaction)

(FCFS algorithm)

works/has Round Robin algorithm

(better response time)

scheduling between queue

① Fixed priority scheduling

② Time Slice

80% to foreground in RR

20% to background in FCFS

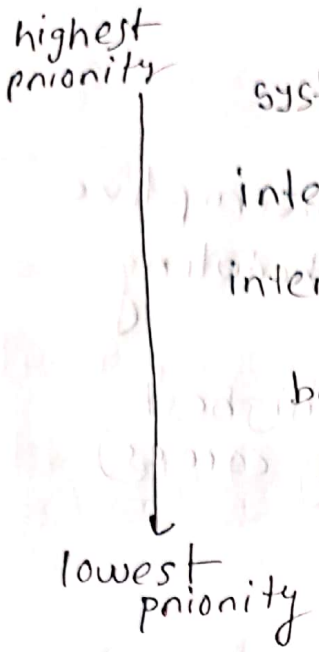


Fig:- Multilevel Queue

multilevel feed back Queue

→ three Queue.

① Q_0 — RR with time quantam = 8

② Q_1 — RR " " " = 16

③ Q_2 — FCFS

window scheduling

→ uses prionity based preemptive scheduling

(context switches when another highest prionity comes)

Multiple Processor Scheduling

(1) Asymmetric multipnocesing :- only one processon acess the system data structen alleviating the need of kernel data shaning

(2) Symmetric " :- each processon is self scheduling each has its own private queue of ready process

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 a ready state.

→ it runs a task with higher priority before another lower priority task, even if the lower priority task is still learning.

→ better cpu utilization

→ 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 preemptive it can be paused)

→ less optimization

→ more response/waiting time

→ SJF, FCFS

→ cpu is allocated to process untill it terminates or switches to waiting state

[In preemptive; cpu is allocated to process for short amount of time]

what is cpu scheduling

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