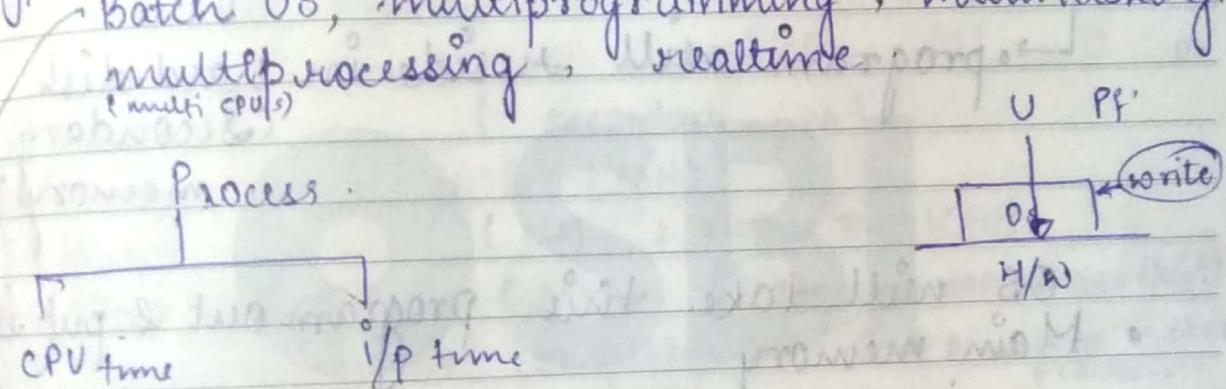
Operating System

FRIDAY 01 JUNE

→ OS is an interface b/w user and hardware.
→ Resource allocator (eg. CPU)
→ manager → memory, processes, files, security etc

Goals
primary → convenience. (PC)
secondary → efficiency

Types of OS:
Batch OS, multiprogramming, multitasking, multiprocessing (multi CPU's), realtime



→ throughput is less. (one by one)
↓
No. of jobs performed per unit time.

Process Management

Attributes of a process.

1.) process id
2.) program counter
3.) process state
4.) priority
5.) General purpose registers
6.) list of open files
7.) List of open devices
8.) protection (privacy)

2018

"There comes a time when silence is betrayal

02 SATURDAY JUNE (153-212) WK 22

C program

ex.c

⇓ compiler (convert high level language to low level language)

⇓ a.out

machine level lang. → understand by OS.

→ program will reside in hard disk (secondary memory)

OS will take this program out & put it in Main memory

In main memory it will create some data structure

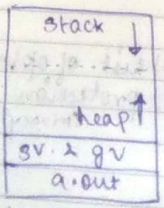
→ (process)

(body (real))

Process is something that is created by OS in order to execute some program.

03 SUNDAY (virtual) (2018)

process boundary



Static variables → created once & remain forever

global variable

Dynamic → heap

MM

Ten thousand fools proclaim themselves into obscurity, while one wise man forgets himself into immortality.

segmentation fault → if we go out of boundary.

WK 23 (155-210) MONDAY JUNE 04

heap will grow upward.

stack grow downward.

for every process → 1 process control block (PCB) (context)

contain all the info (attributes) → maintained by OS

→ States of Process

1.) New (secondary → pick)
2.) Ready (MM)
3.) Run (MM, 1 process run in CPU)
4.) Block or wait (MM, i/p state me daal na aur wait)
5.) Termination or completion (PCB will be deleted/ process → 0)
6.) Suspend ready (S.M)
7.) Suspend wait or suspend block

Operations on process

1.) Creation
2.) Scheduling
3.) Execution
4.) Killing / Delete

Multiprogramming → many processes to run

(with pre-emption) multitasking Time sharing

(without pre emption) CPU will not force it

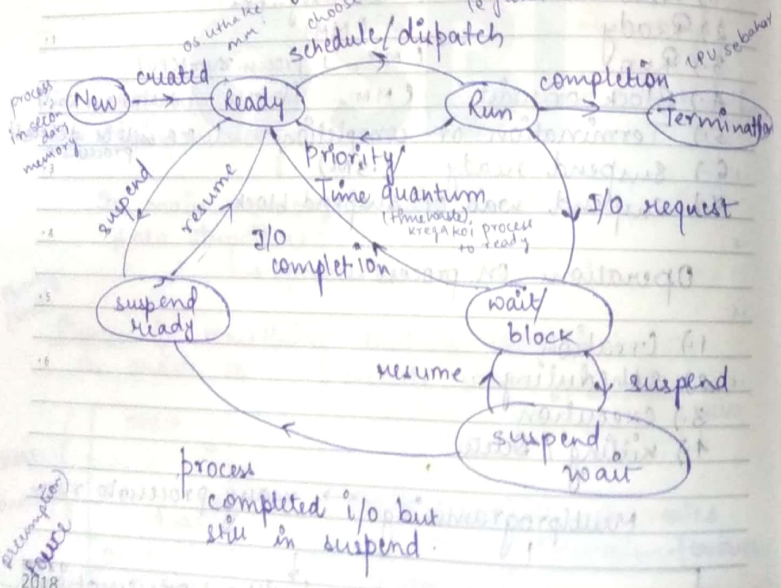
always right to do the right thing

context

05 TUESDAY JUNE

suspend ready → Agar koi new process aagaya aur main memory me space nahi h to ~~agar~~ hum mm ke kuch processes secondary memory me daal ke space banate h aur new process ko run kute h.

- Process state Transition Diagram & various scheduler



2018

"If you can't fly then run, if you can't run then walk, if you can't walk then crawl, but whatever you do you have to keep moving forward."

WEDNESDAY JUNE 06

A process contain two time
↙ ↘
CPU time i/p time

① Long term scheduler
② Short term "
③ MT Medium term

→ Question on Process state.

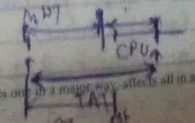
consider a system with 'N' CPU processors and 'M' processes, then

	min	max	
ready	0	M	(depend on long term scheduler)
running	0	N	
block	0 (if CPU bound)	M (I/O)	

→ Various times related to process.

Imp. parameters of processes:

1.) Arrival time → time at which a process enter the ready state (Ready queue)
2.) Burst time
3.) Completion time
4.) Turn around time TAT = CT − AT
5.) Waiting time WT = TAT − BT
6.) Response time → (1st time CPU hit) − AT



2018

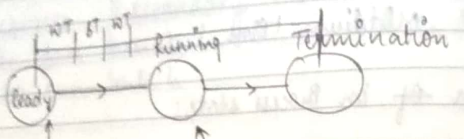
"What affects one in a major way affects all in a minor way."

07 THURSDAY

(before executing it)

Burst time — The amt. of time that is required by a process i.e. CPU time to finish

- completion time — The time at which the process finishes



Spend lot of time in ready state

time spend here depend on the burst time

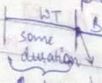
2 types → wait
work → execute

(assuming no i/o time required)

wait time, burst time
(WT) (BT)

$$CT - AT = BT + WT$$

- Turn around time → The difference b/w arrival time and turnaround time.
- waiting time → WT = TAT - BT
- Response time → what is the first time at which the process hits the CPU



taken up for processing (Response time)

2018 The 1st time at which a process get schedule — the arrival time

I have decided to stick to love... Hate is too great a burden to bear.

preempted → pulled out

FRIDAY 08

CPU Scheduling → taking up the process from ready state & giving it to CPU

who → short term scheduler

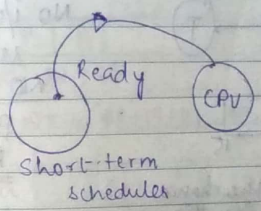
where → Ready state to running

when → when a process moves from —

1.) Run → Termination
Run → wait
Run → Ready

2.) New → Ready i.e. when a process is just created

3.) wait → ready.



pick → scheduler
Run state me dalne ka kaam → dispatcher ka hota h

RT = CT - AT
WT = TAT - BT

2018

Nothing in all the world is more dangerous than sincere ignorance and conscientious stupidity

09 SATURDAY JUNE

FCFS: First come first serve.

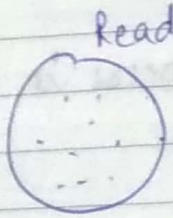
criteria: Arrival time.

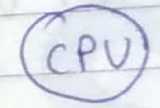
mode: Non-preemptive → Once you choose a job & give it to CPU then you are not going to pull out until it finishes.

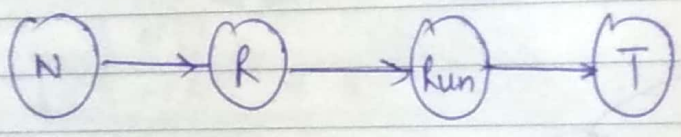
waiting time (WT) = RT	P No.	AT	BT	CT	TAT	WT
0	1	0	4	4	4	0
3	2	1	3	7	6	3
5	3	2	1	8	6	5
5	4	3	2	10	7	5
6	5	4	5	15	11	6

completion time

(CT − AT) → Turn around time

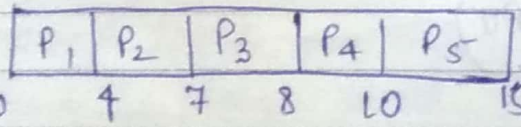






assume
No i/o time required
no need of pre emption
RT = WT.

snapshot



t=0 P1

* waiting to get the chance first time.

10 SUNDAY

t=4 P5 P3 P2 P4

TAT = WT + BT

WT = TAT − BT

Th' = no. of / To

2018

"Be The Peace You Wish To See In The World!

Response Time

After a process gets into the system how much time does it require to get the CPU for the first time.

MONDAY 11 JUNE

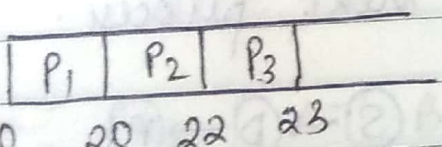
average turn around time —
avg. amount of time that each process is spending. $= \frac{\Sigma TAT}{n}$

avg waiting time $= \frac{\Sigma WT}{n}$.

disadv.

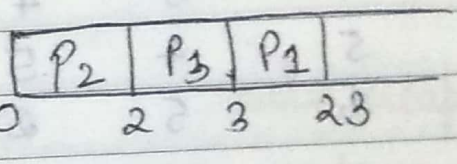
Convoy effect

PNO.	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	2	22	21	19
3	1	1	23	22	21



avg WT $= \frac{0+19+21}{3} = \frac{40}{3}$ ✓

PNO	AT	BT	CT	TAT	WT
P1	1	20	23	22	2
P2	0	2	2	2	0
P3	0	1	3	3	2



avg WT $= \frac{4}{3}$.

⇒ when Big process (high BT) arrives early, it takes lots of time to finish, so during the entire time all other process are going to start & has to wait for the completion of 1st process this is convoy effect.

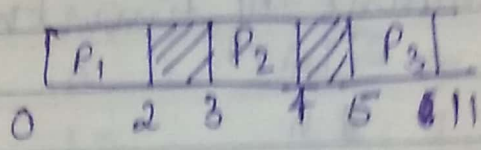
processes are not waiting too long.

2018

12 TUESDAY
JUNE

eg

PNo	AT	BT	CT	TAT	WT
1	0	2	2	2	0
2	3	1	4	1	0
3	5	6	11	6	0



CPU is waiting for process to arrive.

queue → ds.

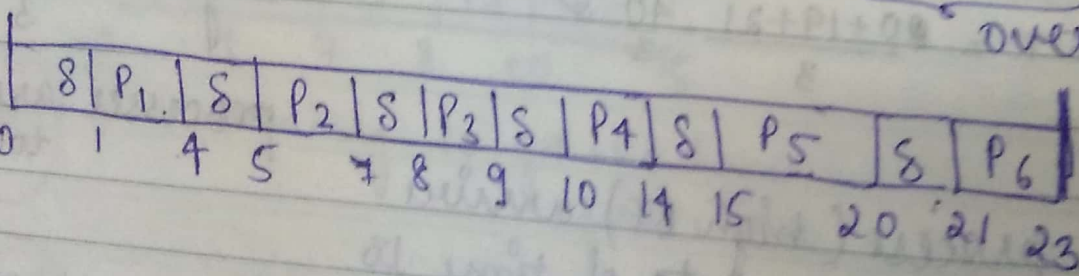
eg. FCFS with overhead.

$\delta = 1$

PNo	AT	BT
1	0	3
2	1	2
3	2	1
4	3	4
5	4	5
6	5	2

context switching time
when a process is running we are going to stop it & then schedule the next process.

overhead.



$$\eta = \left(1 - \frac{6}{23}\right) \times 100$$

Shortest remaining job first (preemptive)

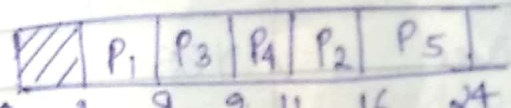
WEDNESDAY
JUNE 13

Shortest Job First –

criteria – Burst time
mode – Non-preemptive → (process does not stop until it finishes)

PNO	AT	BT	CT	TAT	WT
1	1	7	8	7	0
2	2	5	16	14	9
3	3	1	9	6	5
4	4	2	11	7	5
5	5	8	24	19	11

→ Least BT is processed first



convoy effect

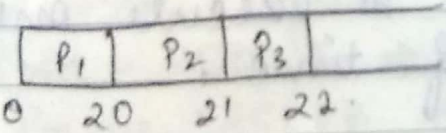
} among the available funcn (process) shortest select

min heap → dsa nlogn time complexity

eg ①

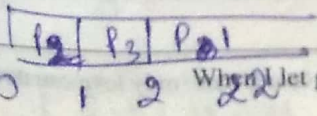
PNO	AT	BT	CT	TAT	WT
1	0	20	20	20	0
2	1	1	21	20	19
3	2	1	22	20	18

avg $\frac{39}{3} = 13$



② 1 2 20 22 20 0
2 0 1 1 1 0
3 1 1 2 1 0

avg WT = 0

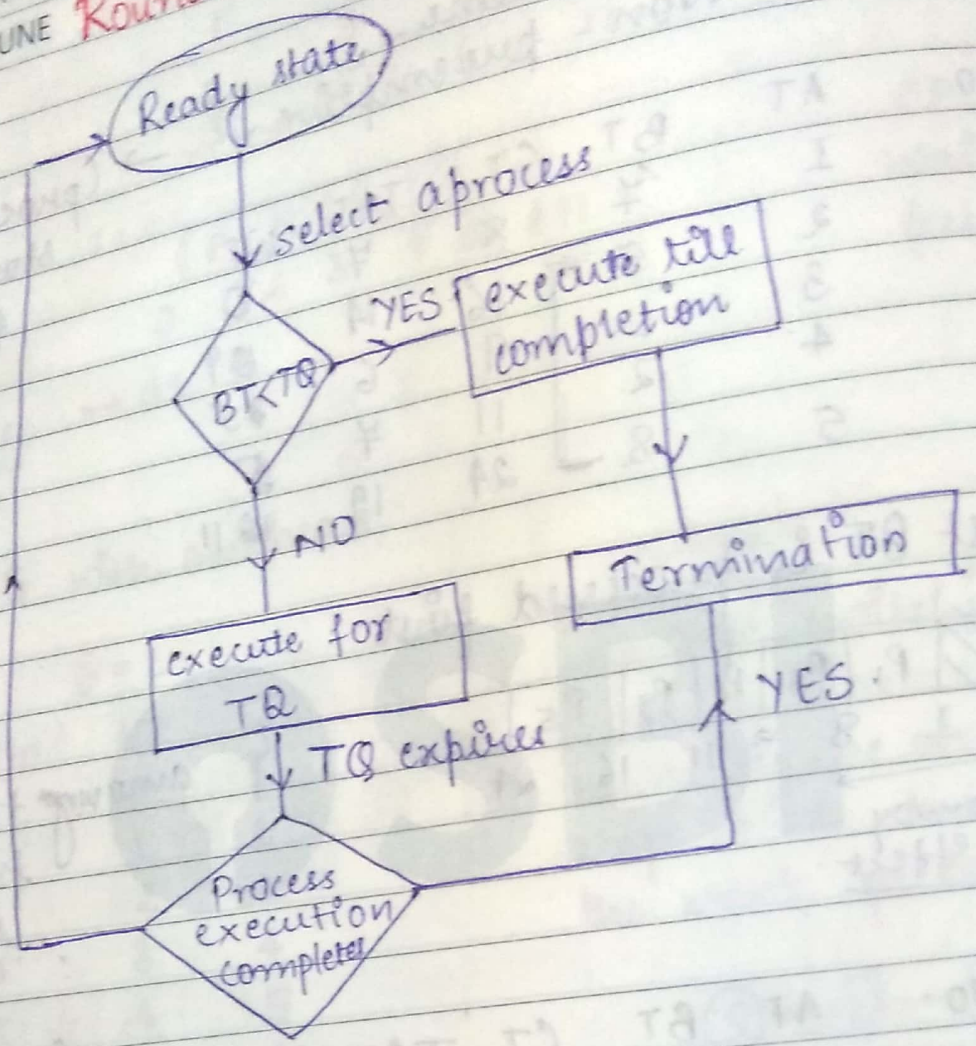


Throughput = $\frac{\text{No. of process}}{\text{total time}}$

When I let go of what I am, I become what I might be -Lao Tzu

14 THURSDAY JUNE

Round Robin Algorithm —



- practically implementation. (no BT) prev

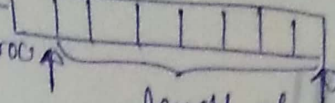
TQ → Time quantum. (we fix)

every process is going to execute only for small amount of time.

- easy to implement
- no starvation problem.

2018

$tp = \frac{\text{no. of pro.}}{\text{length of Proc}}$



length of process.

FRIDAY 15 JUNE

SJF with prediction of BT

SJF

- Advantages
  - → maximum throughput.
  - → minimum avg WT & TAT
- Disadvantages.
  - → starvation to longer jobs
  - → It is not implementable big BT of processes cannot be known ahead.

soln : SJF with predicted BT.

Prediction techniques.

- static
  - process size
  - process type
- Dynamic.
  - simple averaging.
  - Exponential averaging / aging

process size (Bytes)

Pold = 200 KB ⇒ 20 units

∴ Pnew = 201 KB ⇒ 20 units

process type.

Process

- OS ⇓ (3-5) units
- User
  - inter active Gaming (5-8)
  - fore ground (10-15)
  - Back ground (15-20)

"The difference between a dreamer and a visionary is that a dreamer has his eyes closed and a visionary has his eyes open" 2018

16 JUNE Simple average

→ Given n - processes
(P1 - - - - Pn)
→ Let ti be the actual BT
→ Let Ti denote the predicted BT

$$T_{n+1} = \frac{1}{n} \sum_{i=1}^{n} t_i$$

Exponential average / aging

$$T_{n+1} = \alpha t_n + (1-\alpha) T_n \quad -①$$

$0 \le \alpha \le 1$

α → smoothing factor

$$T_n = \alpha t_{n-1} + (1-\alpha) T_{n-1} \quad -②$$

② in ①

$$T_{n+1} = \alpha t_n + (1-\alpha)\left[\alpha t_{n-1} + (1-\alpha) T_{n-1}\right]$$

↓
current value depend on all the previous history

Ex: $\alpha = 0.5$ $T_1 = 10$, actual BT (t_1, t_2, t_3, t_4) SUNDAY
$= (4, 8, 6, 7)$
then T_5

$T_2 = \alpha t_1 + (1-\alpha) T_1$
$= 4 \times 0.5 + 0.5 \times 10 = 0.5 \times 14 = 7$.

2018 $T_3 = \alpha t_2 + (1-\alpha) T_2$
$T_3 = 0.5 \times 8 + 0.5 \times 7$

Lightning makes no sound until it strikes

MONDAY 18 JUNE

RR

TQ = 2 (max allowable time a process can run)

Eg:

PNO	AT	BT	CT
1	0	~~4~~ ~~2~~ 0	8
2	1	~~5~~ ~~3~~ ~~1~~ 0	~~13~~ AT + TQ
3	2	~~2~~ 0	6
4	3	~~1~~ 0	9
5	4	~~6~~ ~~4~~ ~~2~~ 0	21
6	6	~~3~~ ~~1~~ 0	19

FCFS + preemption

queue

P1	P2	P3	P1	P4	P5	P2	P6	P5	P2	P6	P5
0	2	4	6	8	9	11	13	15	17	18	19	21

2

P5

snapshot

context switching

TQ ↓ no. of context switches ↑

TQ ↑ → starvation ↓

	CT	TAT	WT
	8	8	4
	17	17	12
	4	4	2
	6	6	5
	17	17	11
	13	13	10

Eg: TQ = 3

PNO	AT	BT	CT	TAT	WT	RT
1	5	~~5~~ ~~2~~ 0	32	27	22	10 (15-5)
2	4	~~6~~ ~~3~~ 0	27	23	17	5
3	3	~~7~~ ~~4~~ ~~1~~ 0	33	30	23	3
4	1	~~9~~ ~~6~~ ~~3~~ 0	30	29	20	0
5	2	~~2~~ 0	6	4	2	2
6	6	~~3~~ 0	21	15	12	12

P4	P5	P3	P2	P4	P1	P6	P3	P2	P4	P1	P3

TQ ↑ CS ↑ RT ↑

	P4	P5	P3	P2	P4	P1	P6	P3	P2	P4	P1	P3
0	1	4	6	9	12	15	18	21	24	27	30	32	33

TQ ↓ CS ↑ RT ↓

2018

RT

TQ → ∞ RR → FCFS

Change does not roll in on the wheels of inevitability, but comes through continuous struggle.

19 TUESDAY JUNE (170-195) WK 25

Q. Consider 4 jobs P_1, P_2, P_3 and P_4 arriving in ready queue in the same order at time 0. If BT requirements of these jobs are 4, 1, 8, 1 resp., what is completion time of P_1, assuming RR with TQ = 1

	AT	BT	CT
P_1	0	~~4~~ 3 2	9
P_2	0	~~1~~ 0	2
P_3	0	~~8~~ 7	13
P_4	0	~~1~~ 0	4

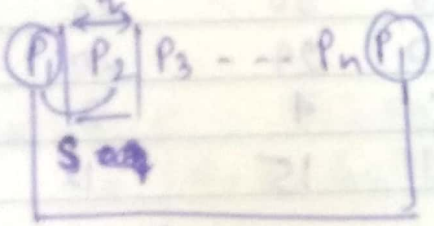
TQ = 1

~~P1 P2 P3 P4 P1 P3 P1 P3~~

P_1	P_2	P_3	P_4	P_1	P_3	P_1	P_3	P_1	P_3

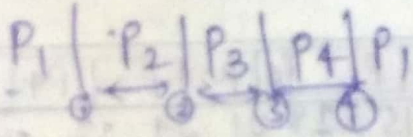
0 1 2 3 4 5 6 7 8 9 ~~14~~

Q. Consider 'n' processes sharing the CPU in RR fashion. If the context switching time is 's' units what must be the time quantum 'q' such that the no. of context switches are ~~requ~~ reduced, but at the same time each process is guaranteed to get the turn at the CPU for every t sec.



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

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



2018

Courage faces fear and thereby masters it

Longest

Process
crite
mod

P NO.
1
2
3
4
5

Q.

Q.

WEDNESDAY JUNE 20

Longest Job first algorithm

Process having longest BT gets scheduled first.
criteria : BT
mode : non-premptive.

PNo.	AT	BT	CT	TAT	WT	RT
1	0	3	3	3	0	0
2	1	2	20	19	17	17
3	2	4 (3)	18	16	12	12
4	3	5 (1)	8	5	0	0
5	4	6 (2)	14	10	4	4 (8-4)

P1	P4	P5	P3	P2
0	3	8	14	20

Longest Remaining time first.

Q2

PNo	AT	BT	CT	TAT	WT	RT
1	1	2 1	18	17	15	0
2	2	4 3 2 1	19	17	13	0
3	3	8 5 4 3 2 1	20	17	11	0
4	4	8 7 5 4 3 2 1	21	17	9	0

	P1	P2	P3	P4	P3	P4	P5	P4	P2	P3	P4	P3	P4	P1
0	1	2	3	4	6	7	8	9	10	11	12	13	14	15	16	17	18

P2	P3	P4
1	9	20	21

Q1

PNo.	AT	BT	CT	TAT
1	0	2 1	12	12
2	0	4 3 2 1	13	13
3	0	8 4 3 2 1	14	14

avg TAT?

$\frac{12+13+14}{3}$

P3	P2	P3	P2	P3	P1	P2	P3	P1	P2	P3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2018

"Faith is taking the first step even when you can't see the whole staircase"

21 THURSDAY JUNE

Highest Response Ratio next (HRRN)

Criteria: Response ratio (RR) = $\frac{w+s}{s}$

w: waiting time for a process so far

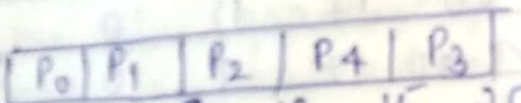
s: service time of a process or BT.

→ HRRN not only favours shortest jobs but also limits the waiting time of longer jobs.

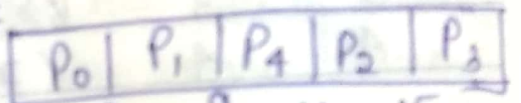
→ mode: non-preemptive.

PNo	AT	BT	CT	TAT	RT
0	0	3			0
1	2	6			1
2	4	4			5
3	6	5			9
4	8	2			5

HRRN



SJF



at 9 RR: $P_2 = \frac{(9-4)+4}{4} = \frac{9}{4}$, 2.25

RR: $P_3 = \frac{3+5}{5} = \frac{8}{5}$, 1.6

RR: $P_4 = \frac{1+2}{2} = \frac{3}{2}$, 1.5

at 13 RR: $P_3 = \frac{7+5}{5} = \frac{12}{5}$, 2.4

RR: $P_4 = \frac{5+2}{2} = \frac{7}{2}$, 3.5

2018

Everything has beauty, but not everyone can see. - Confucius

Priority Scheduling

FRIDAY JUNE 22

Priority

- Static
  - Doesn't change throughout the execution of the process
- Dynamic
  - changes at regular interval of time.

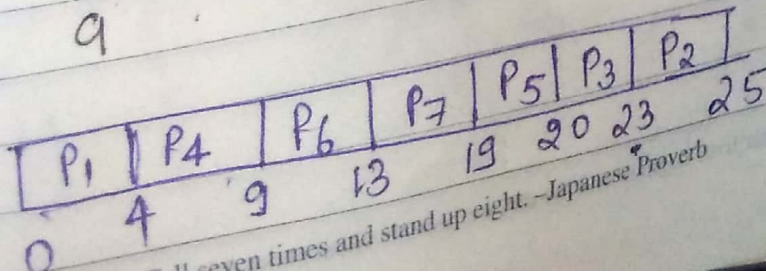
process comes with some priority.

priority scheduling algorithm

- Preemptive
- Non-preemptive

Non pre-emptive priority scheduling

PNo.	priority (L)	AT	BT	CT	TAT	WT	RT
1	2	0	4	4	4	0	0
2	4	1	2	25	24	22	22
3	6	2	3	23	21	18	18
4	10	3	5	9	6	1	1
5	8	4	1	20	16	15	15
6	12 (H)	5	4	13	8	4	4
7	9	6	6	19	13	7	7



2018

stop when high priority comes.

23 SATURDAY
JUNE

Pre-emptive priority scheduling

PNo.	Priority	AT	BT	CT	TAT	WT	RT
1	2	0	3				
2	4	1	1				
3	6	2	2				
4	10	3	8				
5	8	4	4				
6	12	5	5				
7	9	6	6				

P_1	P_2	P_3	P_4	P_6	P_4	P_7	P_5	P_3	P_2	P_1
0	1	2	3	5	9	12	18	19	21	22	25

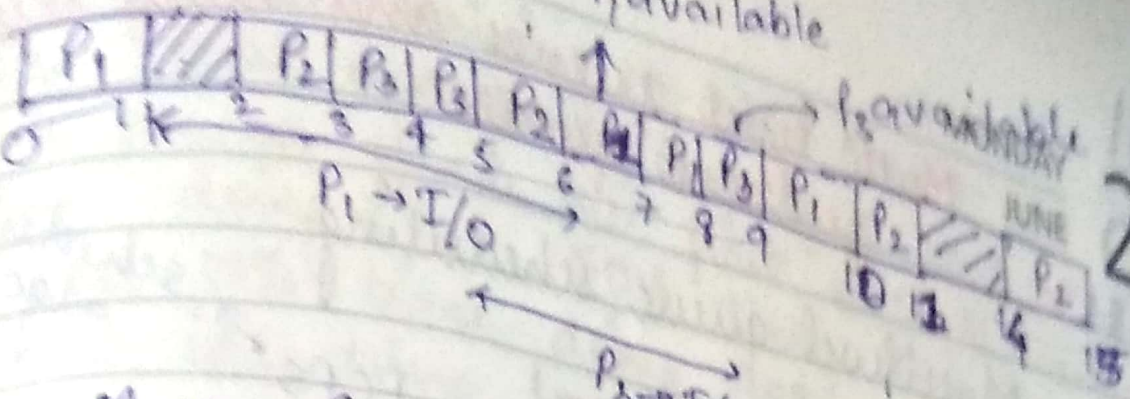
stop if next arrived ones have high priority

At some point the pre emptive will become non-preemptive.

Pre-emptive priority with processes contains CPU and IO time.

PNo.	AT	priority	CPU	I/O	CPU	CT	TAT	WT
P_1	0	2	10	5	3	SUNDAY	16	6
P_2	2	3(L)	3	3	1	15	13	9
P_3	3	1(H)	2	3	1	9	6	3

2018



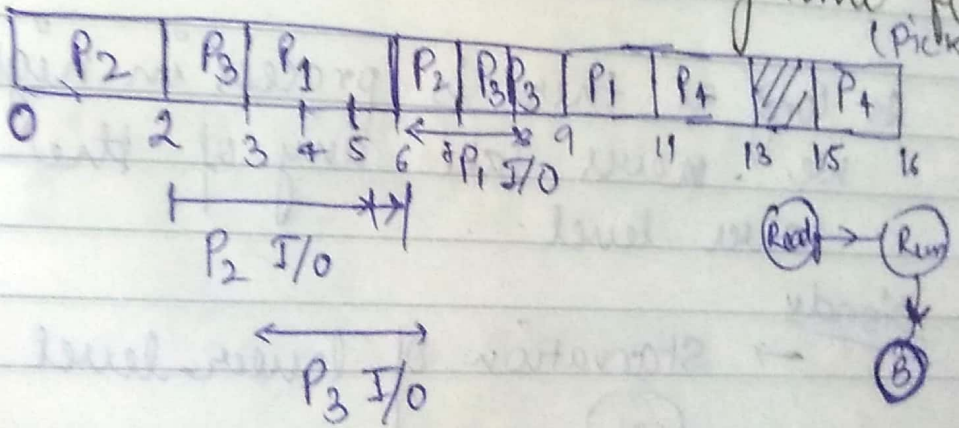
$$\eta_{CPU} = \left(1 - \frac{4}{15}\right) \times 100$$

SRTF with processes contains CPU & I/O time

PNo	AT	CPU BT	IO BT	CPU BT	CPU BT
1	0	3	2	2	2
2	0	2	4		1
3	2	1	3	2	3
4	5	2	2	1	3

CT	IO
11	6
7	4
9	4
16	8

shortest remaining time first. (Pick least BT)



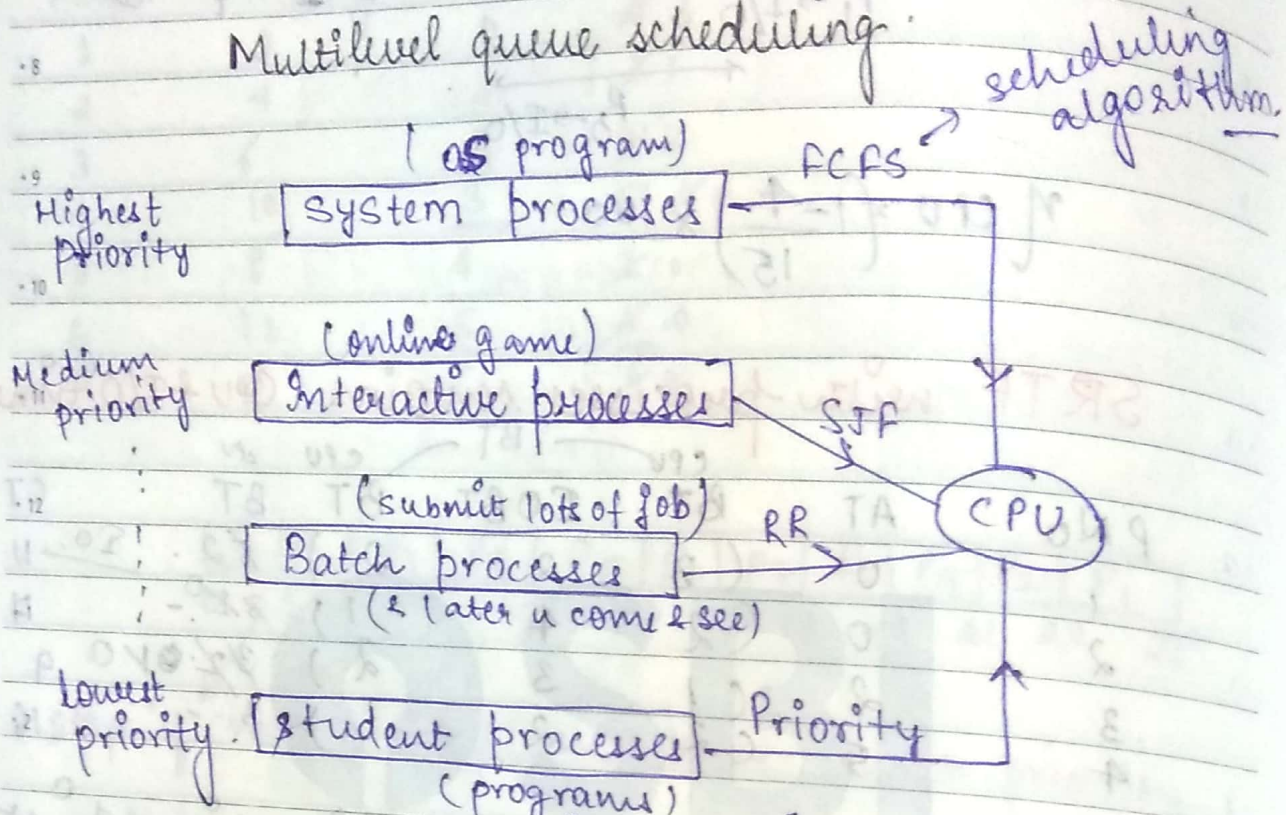
TAT = CT − AT

WT = TAT − BT (CPU) ← process

26 TUESDAY JUNE

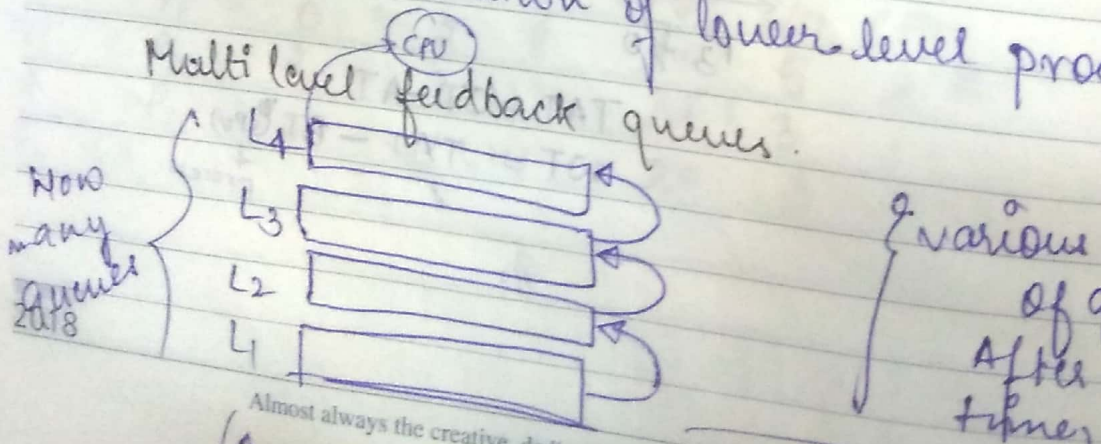
Multi level queues and multi level feedback queues.

Multilevel queue scheduling.



As long as there is process in higher level we never take any of the process of lower level.

Disadv → Starvation of lower level processes.



Almost always the creative, dedicated minority has made the world better.

19 THURSDAY JULY

File Allocation

- File system is the most visible part of the OS.
- Files are stored in disks.
- Allocation of space for files in disks.
- Keep track of free disk block.

Allocation Methods.

- An allocation method refers to how disk blocks are allocated for file.

Disk comprise of disk blocks.

- Disk space should be allocated so that disk space is utilized effectively & files can be accessed quickly.
  - Contiguous allocation.
  - Linked allocation.
  - Indexed allocation.

Contiguous Allocation.

- Each file occupies a set of contiguous blocks on the disk.
- If only one job is accessing the disk, accessing block $b+1$ after block b requires no head movement.
  - Disk seeks are minimal.
- Simple — only starting location (block#) & length (no. of blocks) are required.

2018

FRIDAY
JULY 20

• Sequential & direct access.

directory

Name	start add.	length

• Wasteful of space. (dynamic storage allocation problem)
  – How to satisfy a request of size n from a list of free holes.
  – First fit or best fit.
    (smallest space that is enough)

• External fragmentation – compaction. (solve)
• Difficult to determine how much space is needed for a file (before the creation of file).
• Files cannot grow.
  – can overestimate the space needed, results in wastage of space.
  – Find a larger hole, copy contents to new space, free old space.

• Even if file size is known in advance, file may grow over a long period of time – internal fragmentation.
• Modified contiguous allocation.
  – Initially, contiguous chunk of space is allocated.
  – If this is not large enough, another chunk of space (extent) is allocated.
  – Ex. Veritas file system.

2018

The surest way to be happy is to seek happiness for others.

Linked Allocation.

- solves all problems of contiguous allocation.
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- Directory entry has a pointer to the first & the last block of file.

directory

Name	start add	end add
		25 | -1 → pointer

- No external fragmentation.
- Size of a file need not be declared when the file is created.
- File can continue to grow as long as free blocks are available.
- Compaction not needed.

Disadv.

- No Random access (only sequential access)
- space required for pointers.
  - collect blocks into multiples called clusters.
  - less no. of pointers
  - fewer disk-head seeks
  - Increased internal fragmentation.

MONDAY JULY 23

- Reliability
  - If pointers are lost or damaged?
  - use doubly linked list.
  - Store the file name & relative block number in each block.
- Variation of linked allocation — File — allocation table (FAT) — disk-space allocation used by MS-DOS
- A section disk at the beginning of each volume contains this table.

Indexed Allocation

- Solves the external fragmentation & size-declaration problem of contiguous allocation.
- Solves the direct ~~pro~~ access prob. in linked allocation.
- Brings all pointers together into one block, the index block.
- Each file has an index block, which is an array of disk-block addresses

directory

Name	Index block
jeep	19

- ith entry in the index block points to the i^{th} block of the file.
- Supports direct access

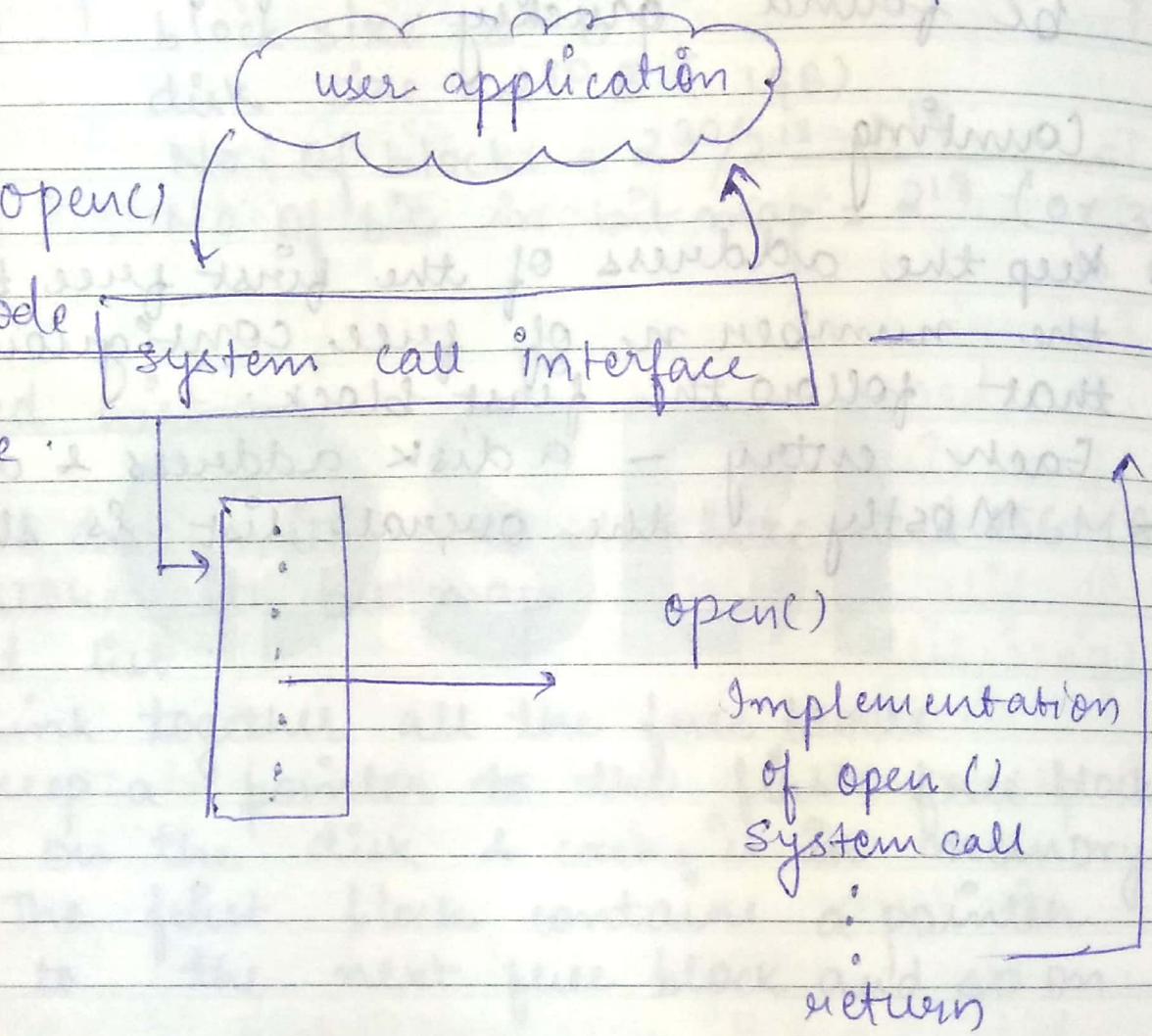
2018

"Talk to yourself like you would to someone you love." Brene Brown

System Calls

- Available as assembly lang. instruction

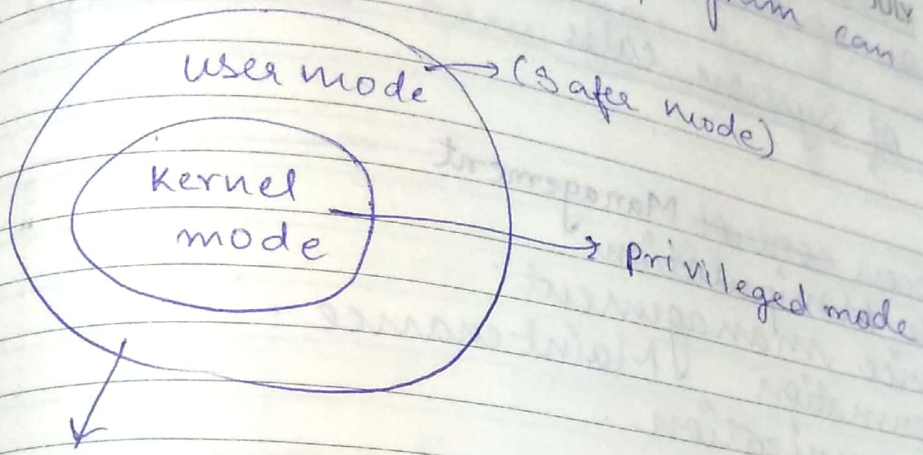
API – system call – OS Relationship



System calls provide an interface to the services made available by an OS.

MONDAY JULY 30

user mode & kernel mode are two modes in which a program can execute



does not have direct access to memory to hardware.

If the program executing kernel mode ~~crash~~ crash then the entire system will crash.

→ when user mode ~~who~~ wants to access to resources then it makes a call to OS telling that it need resources.

* when a program switches from user mode ~~and~~ to kernel mode it is called context switching

→ System call is the programmatic way in which a comp program req a service from the kernel of the OS.

2018

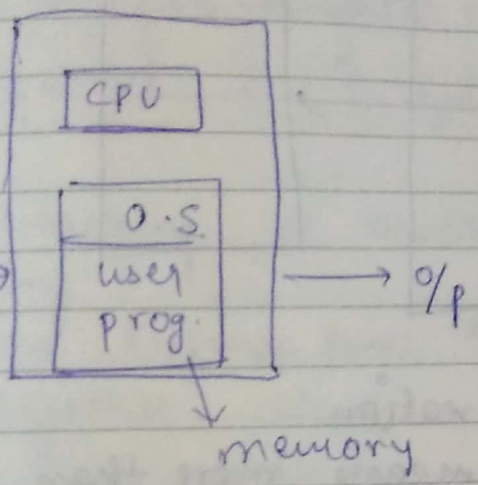
"We must substitute courage for caution

WEDNESDAY
AUGUST 08

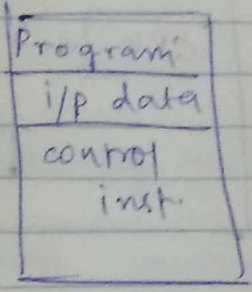
Types of OS

→ mainframe comp.

punch card

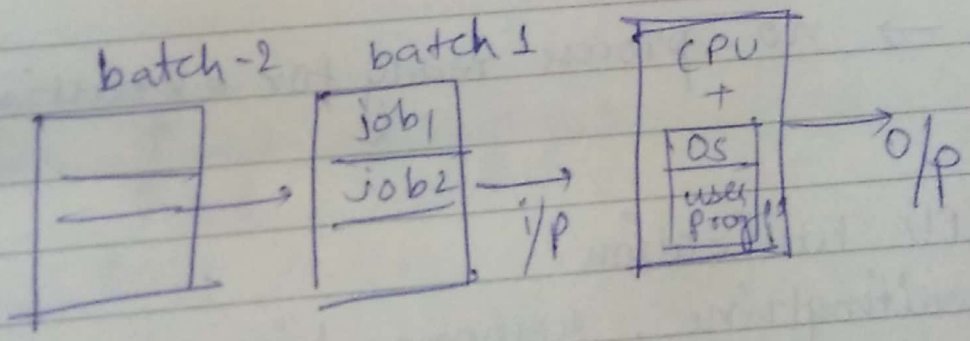


job



→ Batch Processing OS.

jobs with similar needs are batched together & executed through the processor as a group.



Adv.

- need same
- no manual intervention needed.
- less time (loading compiler)

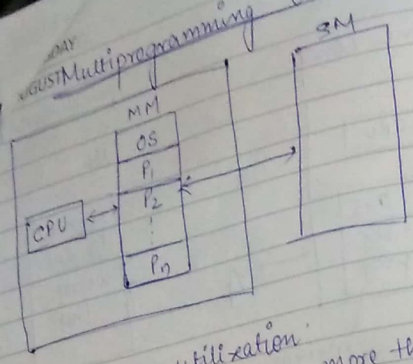
Dis

- memory limited
- interaction of I/P & O/P devices directly with CPU

Today knowledge has power. It controls access to opportunity and advancement. - Peter Drucker

07 AUGUST

Multiprogramming OS.



→ Maximize CPU utilization.
→ Multiprogramming means more than 1 process in MM which are ready to execute.
→ Process require CPU time & I/O time, so if running process perform I/O or some other event which do not require CPU instead of sitting idle. CPU make a context switch & pick some other process.

→ CPU idle → no process ready for execution.

Adv
- → High CPU utilization
- → less waiting time, response time etc.
- → may be ~~executed~~ extended to multiple users.

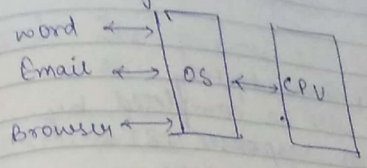
Dis
- → Difficult scheduling
- → MM management is req.
- → Memory fragmentation
- → Paging

2018

As people are walking all the time, in the same spot, a path appears. - John Locke

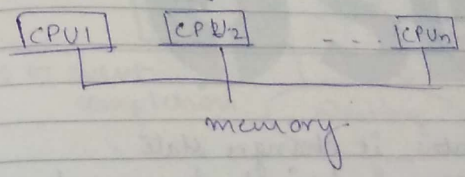
FRIDAY AUGUST 10

Multitasking



→ Multitasking is multiprogramming with time sharing.
→ Better Response time & executing multiple process together.

Multiprocessing.



Adv.
- ↑sed throughput
- ↑sed Reliability
- cost saving.
- parallel processing

Dis
- more complex
- Overhead reduce throughput.
- large main memory

"The way to develop self-confidence is to do the thing you fear and get a record of successful experien

11 SATURDAY AUGUST

Real Time

→ RTOS response to i/p immediately.
→ In RTOS, the task will be completed by specified time & its responses in a predictable way or unpredictable way.

Hard RT

* failed if RT too long
* Sec. storage is limited

Soft RT.

* less accurate if response time is too long
* useful in application such as multimedia

Process State

* As a process executes, it changes state.
* The state of a process is defined in part by the current activity of that process.

Each process may be in one of the following states —

12 SUNDAY

NEW → The process is being created.

RUNNING → Instruction are being executed.

WAITING → The process is waiting for some event to occur (such as an I/O

2018

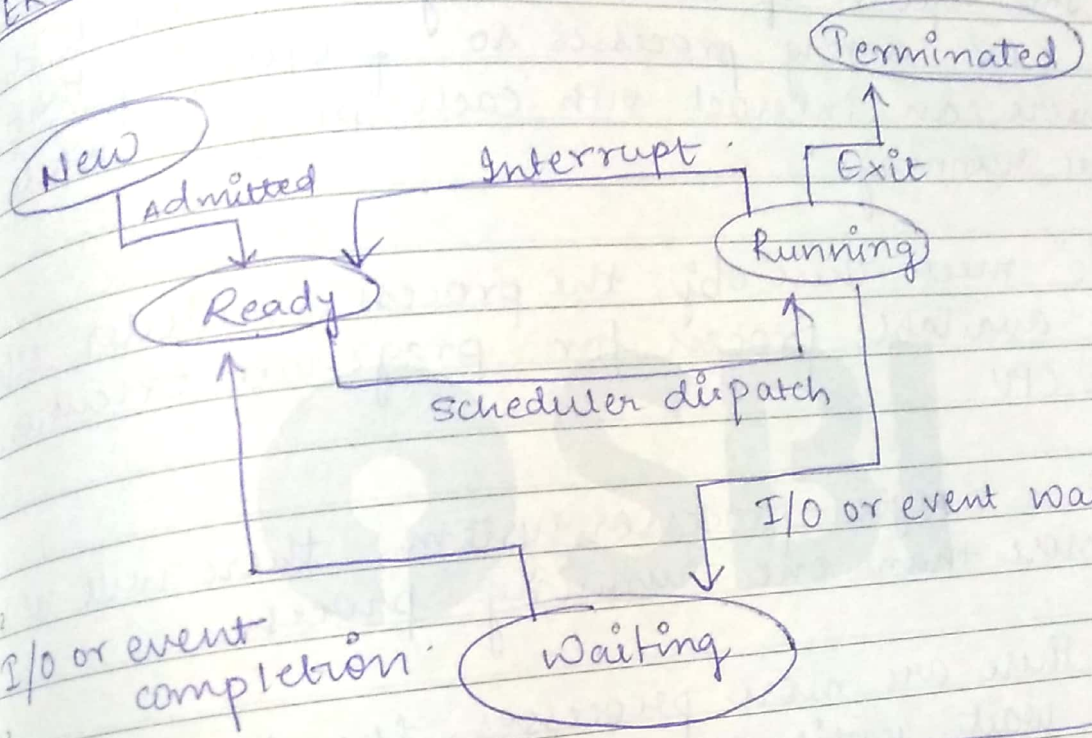
That some achieve great success, is proof to all that others can achieve it as well." - Abraham Lincoln

MONDAY AUGUST 13

completion or reception of a signal.

READY → The process is waiting to be assigned to a processor.

TERMINATED → The process has finished execution d



Process Control Block

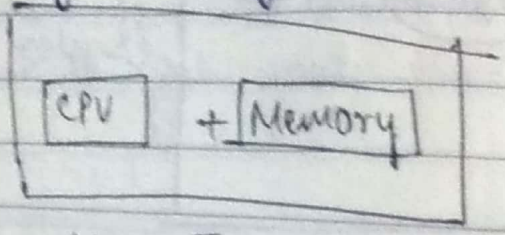
Each process is represented in OS by a PCB.
— also called Task CB.

- process state → ready, run.
- " no → process ID.
- Prog. counter → add. of next inst.
- CPU Registers.
- Memory Units → CPU scheduling inf. Memory manag. inf. Acc. inf.
- List of open files → I/O status inf → which I/O is assigned to process that are used by process.
- . . .

ing

18 SATURDAY AUGUST OS

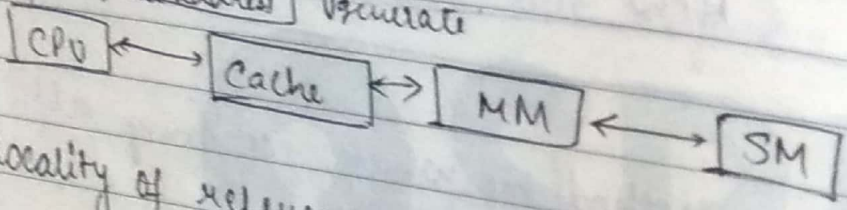
Memory Management.



computer

1.) Size (↑)
2.) Access time (↓)
3.) per unit cost (↓)

Hirachy of memory



locality of reference.

- execution sequencial nature.

To access MM → physical address.

LA → PA.

↳ is used to access secondary memory.

OS → LA → PA

which part of sec memory is stored in MM & where.

→ Contiguous & Non-contiguous.

(array) ↓ (linked list)

- External fragmentation.
- fast access.

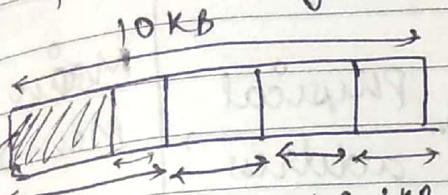
19 SUNDAY

2018

"I've finally stopped running away from myself. Who el

contigious Memory Allocation

fixed size partitioning

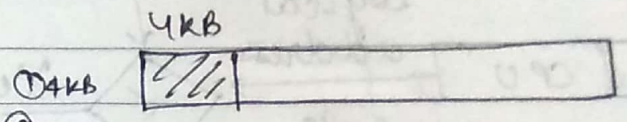


1KB file
1KB is wasted

- In 1 partition only 1 process.
- internal frag.

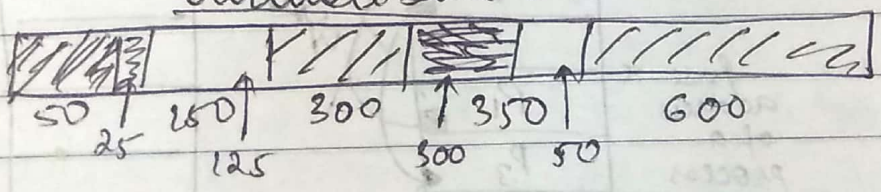
(paging)
(modified fixed size partitioning)

variable size partitioning



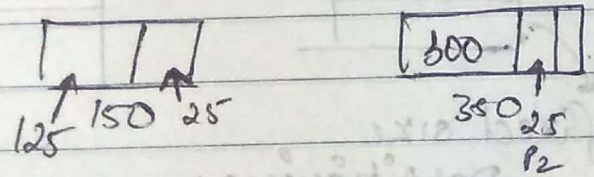
- no internal fragmentation
- jaise-jaise process aayegi hum memory allocate karenge.

variable Size Parti



$P_1 = 300$
$P_2 \rightarrow 25$
$P_3 \rightarrow 125$
$P_4 \rightarrow 50$

First



Best

P_4 left.
↳ cannot be satisfied

Worst

fixed size Parti

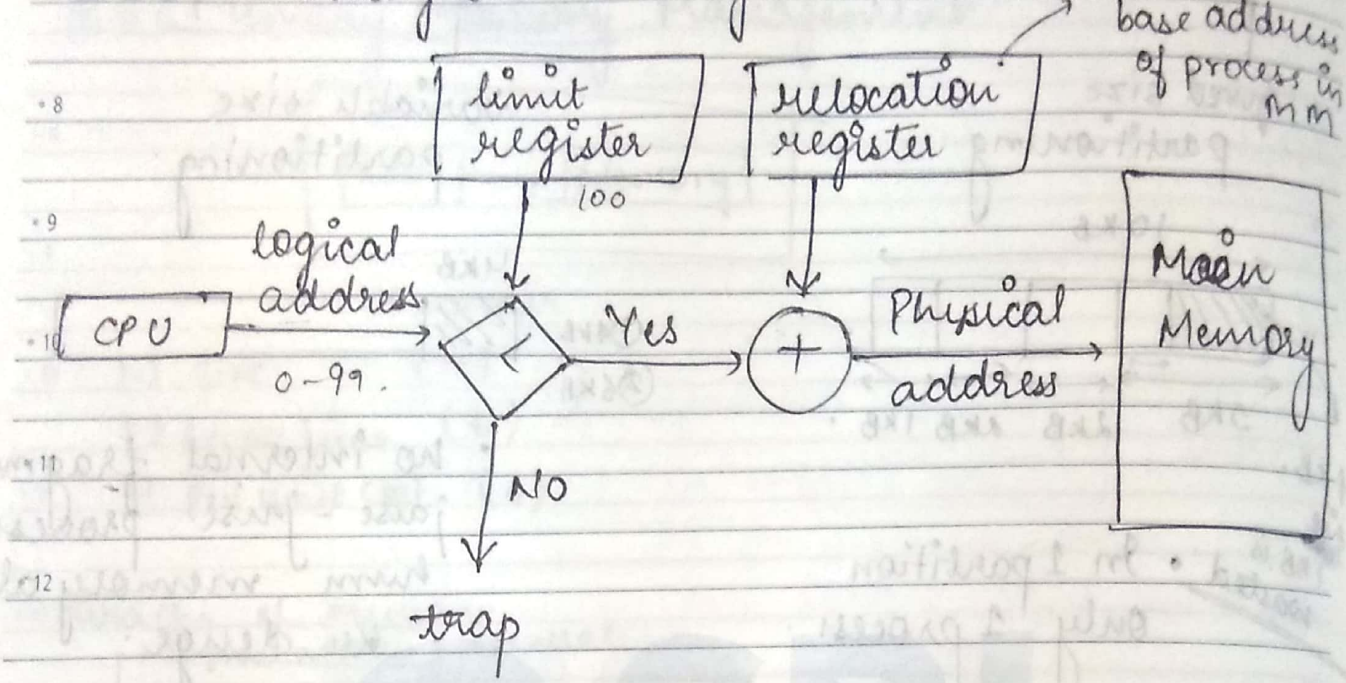
first

Best

Worst

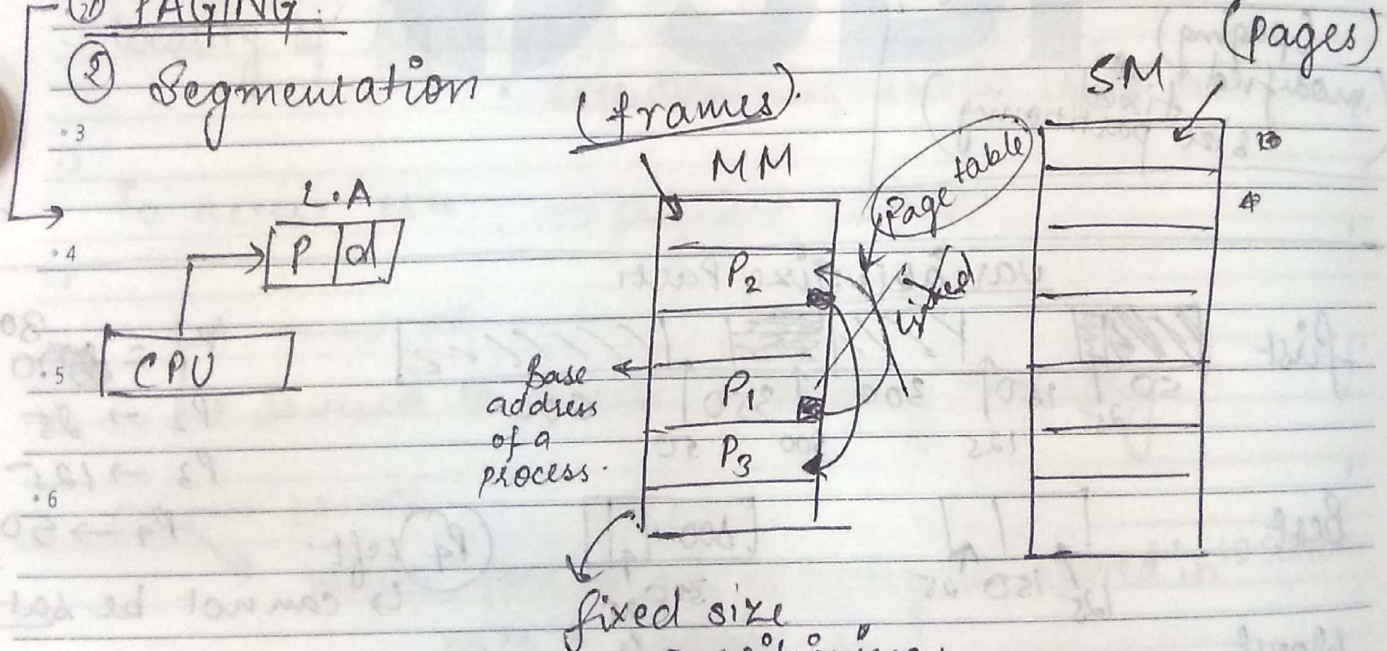
2018

Contigious Memory allocation



Non - Contigious

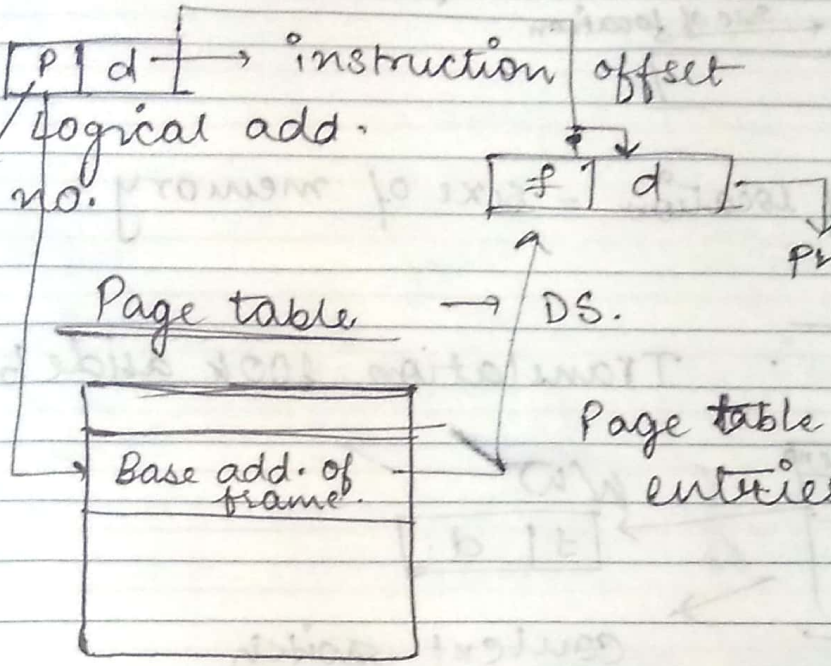
1. PAGING
2. Segmentation



Fixed size partitioning.
(Size of each partition should be same).

size of page = size of frame.

Process pages were already divided in pages & pages can be placed into frames. in non contigious fashion.



Page table contain no. of entries = no. of pages process having in S.M.

Every process have its independent page tab

Page table Base register.

holds page table base address.

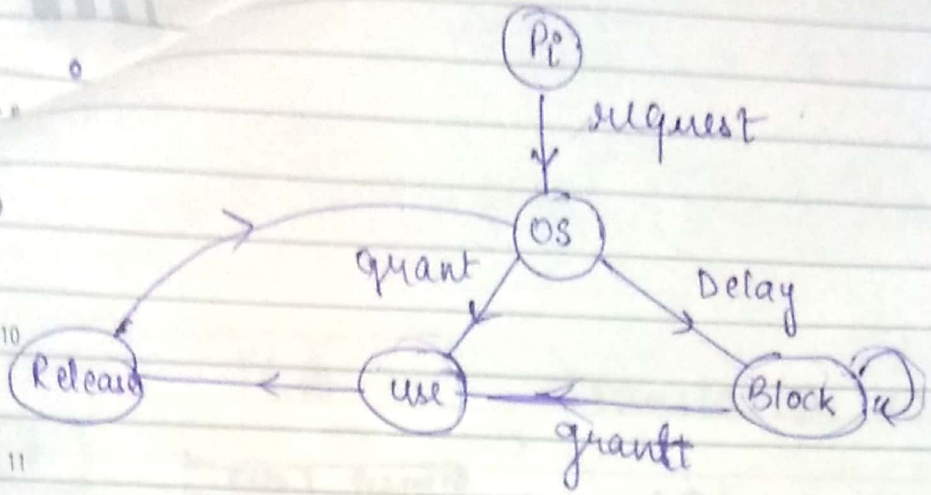
PT is not stored in PCB.

PTBR stored in PCB.

PT is in MM.

25 SATURDAY AUGUST

Deadlock.



- multiprogramming system a no. of process competes for limited resources.
- If resources are not available at that instance the process enter into waiting state
- If a process unable to change its waiting state indefinitely (because the res requested by it are held by another waiting process) the system is said to be deadlock.

System Model

- Every process req for reso.
- if entertained → process use the res
- process must release the res. after use.

26 SUNDAY

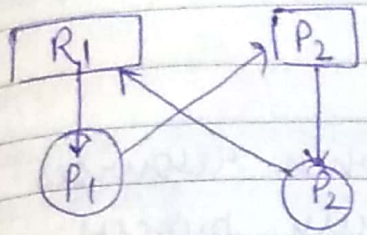
4 cond^n.

- Mutual exclusion → atleast 1 res. in sys. which can be used in non sharable mode
- Hold & Wait → A process is holding at least 1 res. & requesting another which is being held by other process
- No preemption → if the work is not over resource cannot be taken.

2018

Quiet and silence is a form of power. Thoughtful and wise people are not talkative people. – Dr T.P.Chia

- Circular Wait → P1 is waiting for R2 & P2 for R1.



Strategies to avoid deadlock.

- ~~condⁿ~~ ~~Appro~~
- Deadlock ignorance. (reboot)
- Deadlock Prevention (not let the system into dl state)
- Deadlock avoidance (S+U) (") .....
- Deadlock Detection & Recovery (U2)

Deadlock prevention (safe)

condⁿ	Approach
Mutual exclu.	Spool everything (not possible to dis-satisfy)
Hold & wait	Request all res. initially
No preemption	take res. away
Circular	wait

~~Avoid~~ eliminate dl

(H&NO)

- Conservation approach – process is allowed to start execution if & only if it has all the required resources.
- Do not hold – process will acquire the desired res., but before making any fresh req., it must release all the res. currently holding
- Wait timeout. – we place a max. time upto which a process can wait after which process must release all the holding res. earlier

2018

He who does not understand your silence will probably not understand your words. – Elbert Hubbard

28 TUESDAY AUGUST

Eliminate No preemption —
- preempt res. from process when resources required by other high priority process.
- Forceful preemption
- waiting process se chheno.

Eliminate Circular wait.

- Each resource will be assigned with a numerical numbers. A process can req. the resources only in increasing order of numbering.
- req lesser no. reso. the release all the res. of larger no.

P_1	P_2	DL
R_1	R_2	
R_2	R_1	

R_1	R_1	Req order same No DL
R_2	R_2	

Deadlock Avoidance

System maintain a set of ds using which it takes a decision whether to entertain a req. of or not, to be in safe state.

Irreducible cycle → will not give the res. to that process.

RAG Resource Allocation Graph.

- Graphical representation of resource state.
- with the help of this, we try to find out whether the system is in deadlock state or not.
- cycle → possibility of deadlock.
- Irreducible cycle → DL.

2018

Silence is the best way to react while angry. Matovu joseph

OS does not directly give the res. it presents to give it, check if there is cycle or not,

WK 35 (241-124)

WEDNESDAY
AUGUST 29

Banker's Algo.

DS is maintained & apply the safety algo, if it is in unsafe state, then we undo the state & reach back to the previous state.

Available: Available[j] = k , k instances of R_j type
Max:
Allocation: m x n matrix
Need: ↑ P ↑ R

Max[i][j] = k, Process P_i may request k instances of resource type R_j.
Allocation[i][j] = k P_i currently allocated k instances of resource type R_j.
Need[i][j] = k P_i need k more instances of R_j to complete its task.

Need[i][j] = Max[i][j] - Allocation[i][j]

Safety Algo

Step 1. Work & Finish vector of length m & n.
Work = Available.
Finish[i] = False. i = 0, 1, ... n-1

Step 2. a.) finish[i] == false } find index i
b.) Need[i] ≤ Work }
if exist go to Step 4.

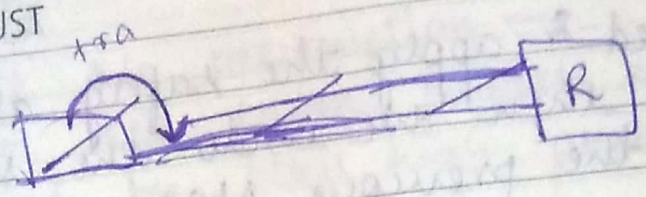
Step 3 — Work = Work + Allocation i
& Finish[i] = true. (Go to 2)

Step 4 — if finish[i] = true, for all i then system is in safe state.

2018

"If you're presenting yourself with confidence, you can pull off pretty much anything." Katy Perry

30 AUGUST



Resource Request Algo

Request$_i$[j] = k. Pi wants k instances of Rj

(1) Req$_i$ ≤ Need$_i$
goto step 2;
otherwise
raise an error cond^n
∵ the process has exceeded its maximum claim

(2) if req$_i$ ≤ available.
goto step 3.
otherwise
Pi must wait.

(3) Have the system pretend to have allocated the req. to resources to process Pi, by modifying the state as follow —

Avail = Avai − Req$_i$
Allocation = Allo$_i$ + Req$_i$
Needi = Needi − Req$_i$

if state found is safe
PP allocate.
not ✗ restore

2018

FRIDAY
AUGUST 31

Safety

After applying safety algo.

if some process req. for more res. then add that in in allocation & find available & Need.

(Deadlock Detection & Recovery)

RAG.

→ kill one by one → WT, RT & TAT ↑ & throughput is suffered.

→ kill all processes. → it will not help in resolving the prob(no guarantee that there will be no deadlock again.

to break the cycle.

Process to be killed is choosen on the basis of priority of process

- how much % of work the process has done.
- how many resources the process has already used.

* Rollback rather than terminating any process. to previous state.

Ostrich Algo.

Until & unless no deadlock occurs it will allow the process to use the resources it want & when deadlock occurs it will find recovery algo. to recover from dl.

2018

The deepest feeling always shows itself in silence. — Marianne Moore

I see ignore the prob. as if it does not exist

13 TUESDAY NOVEMBER

OS.

(317-048) WK 46

Disk & Drum Scheduling

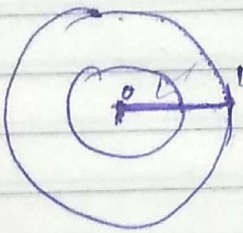
- File allocation method.
- File Access method.

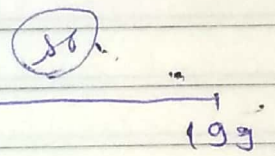
↓ reduce the seek time

in case of multiple i/o requests, ds algo must decide which req. must be executed first.

The process requesting the disk access - there will be a queue of track no.

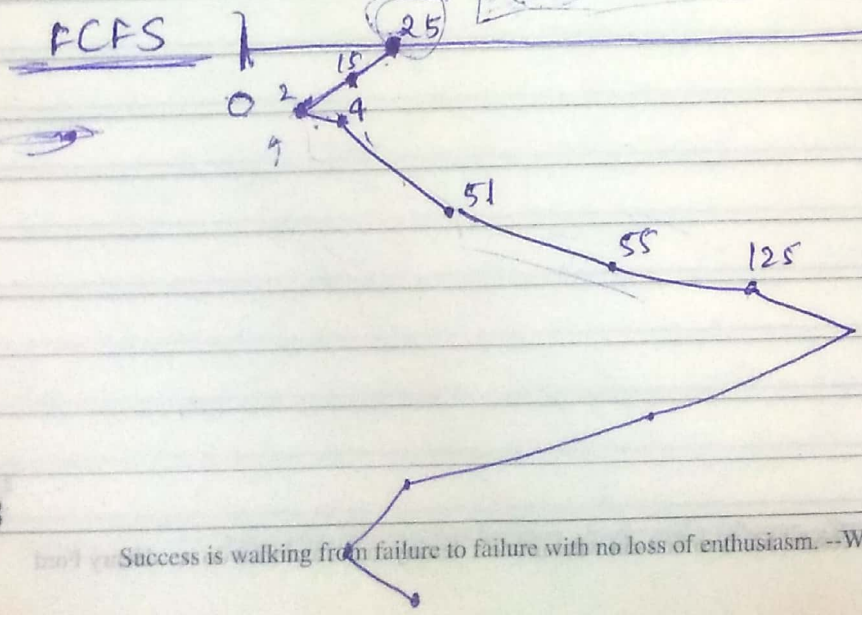
* Time req. to move the r/w head on the desired track = seek time.
* Time taken by disc to rotate to reach a particular sector is rotational latency.





Queue (15, 2, 4, 55, 180, 190, 23, 35)

(a) FCFS



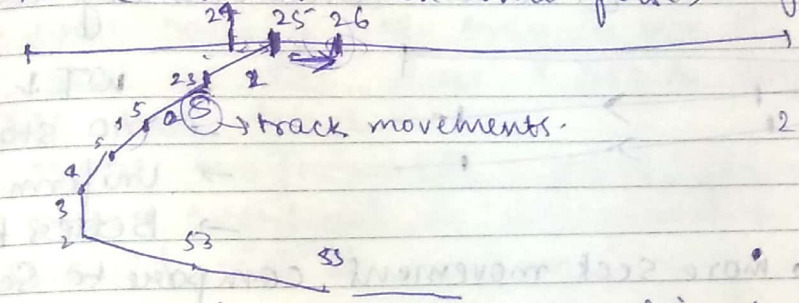
- Simplest disk sch method.
- no starvation (may suffer from convoy effect.
- Easy to implement

2018

Success is walking from failure to failure with no loss of enthusiasm. --Winston Churchill

→ more seek time
→ more WT & RT
→ inefficient

(b) • SSTF (Shortest seek time first) greedy soln



In case of tie req. in the dir of head movement is processed first.

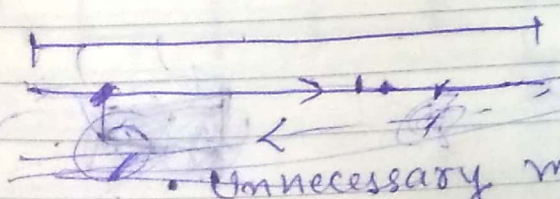
• track movement has reduced as compare to FCFS. seek time also.
• less avg WT & ↑ throughput.
• reduce RT

* Similar to SJF, problem of starvation.
→ overhead to find closest request.
→ high variance in RT & WT.

(c) SCAN (Elevator Algo)

it scans in one dir at a time the entire track & all the processes that come in b/w will be approved.

No starvation. (WOT may ↑)
• (bounded wait)
low avg WT.
• long WT for locations just visited by head.
• Unnecessary move till the end of disc even if there is no req.



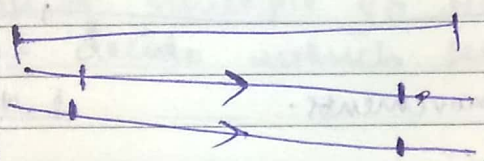
2018

Successful entrepreneurs are givers and not takers of positive energy. --Anonymous

15 THURSDAY NOVEMBER

C SCAN
Circular Scan.

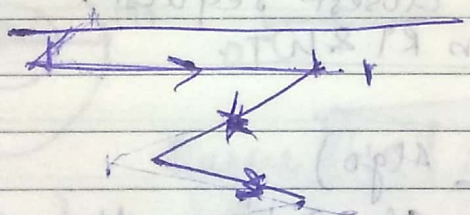
↳ always scan in 1 dir (not in reverse) so that the once arrived first will be scanned first
(199 is considered adjacent to 0)



→ WT ↓
→ no starvation
→ Uniform WT.
→ Better RT.

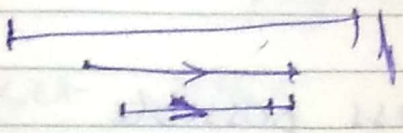
→ more seek movement compare to Scan.

© LOOK
will look whether the req is there or not (scan in the given range)
instead of going till last track, we go till end req & then change dir.



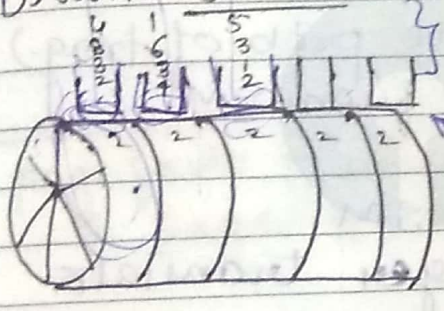
→ used in case of less load.
→ better performance than SCAN
→ overhead to find last req.

CLOOK (best disk scheduling)



→ takes adv of both C-SCAN & LOOK
→ will satisfy req. only in one dir.
→ will go till last req & return but not till last track.
→ more uniform WT, more efficient
⇒ more overhead in calculations.
→ should not be used in case of more load.

Drum Scheduling

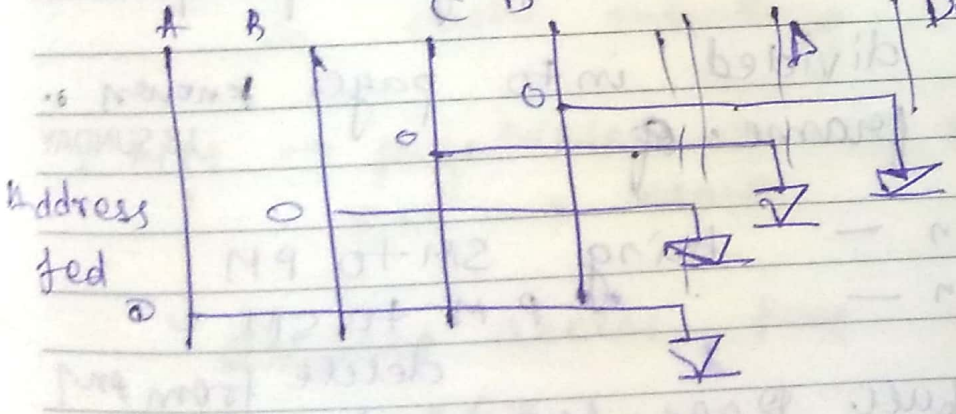


→ arrange the req. in the order of sector below, so that max requests are fulfilled at a time.

sector queuing

USB (Pendrives) → Electronic semi-conductor
FAT-32 use.

• NAND — Flash Memory



* Buffer capacity — data is stored in the circuitary
↓
stored parallelly

2018