OS-3

1) First fit algorithm :-

In these approach we allocate the first free portion.

Best fit :- It deals with allocating the smallest free partition which meets the requirement of the requesting process.

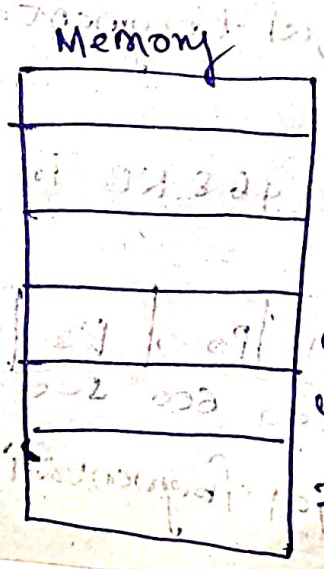
* This method first searches the smallest space that is adjustable

Worst fit :- Here we allocate the largest available free portion.

* It is reverse of best fit.

Here Memory portions of size are 200 KB, 400 KB, 600 KB, 500 KB ~~and~~, 300 KB, 250 KB

i.e



parent

child

P. M

Here allocated process are

$P_1 = 357$ KB

$P_2 = 210$ KB

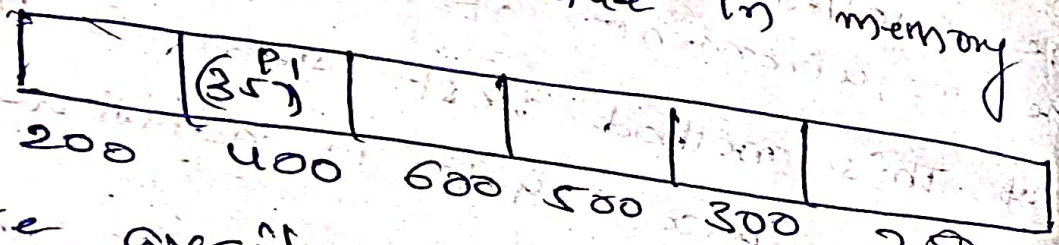
$P_3 = 468$ KB

$P_4 = 491$ KB.

(1) ~~First~~ fit :-

fitting P_1 process value in memory

i.e



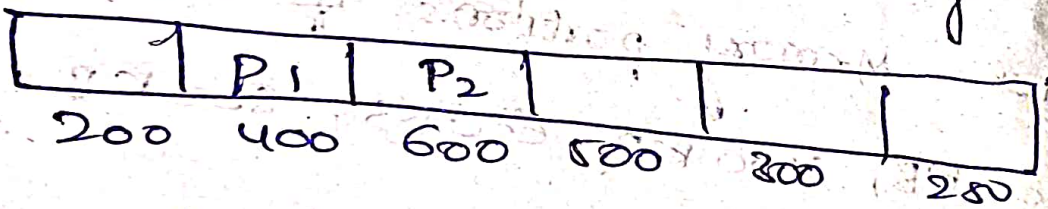
Here available internal fragmenta

= total space - occupied space

= 400 - 357 = 43 KB

Fitting $P_2 = 210$ KB in memory

i.e

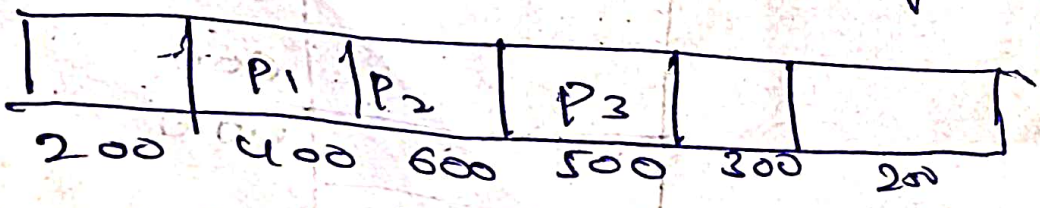


Here internal fragmentation = 600-

= 390

Fitting $P_3 = 468$ KB in memory

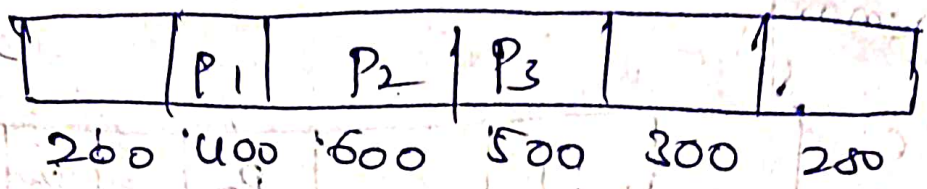
i.e



Here, internal fragmentation = 500-46

= 32 KB

Fitting $P_4 = 491$ KB,



it is not possible to fit P4 since no required space is available

So, external fragmentation required while allocating P_4, which means all the remaining space in filled space are gather at one place called Compactation.

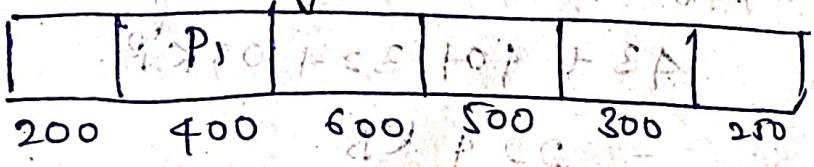
Total internal fragmentation

$= 43 + 390 + 32 = 465$ KB.

(ii) Best fit :-

fitting $P_1 = 357$ KB in smallest partition in memory

i.e



Here internal fragmentation $= 400 - 357$
$= 43$ KB.

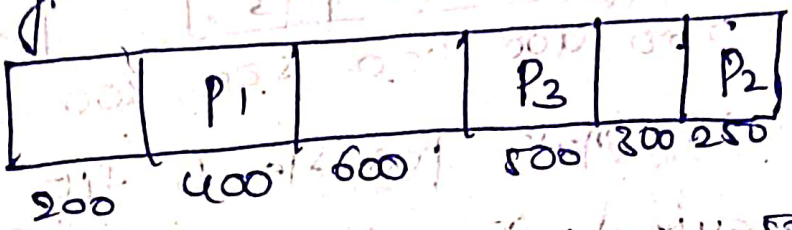
fitting $P_2 = 210$ KB in smallest po of memory

i.e

internal fragmentation = 2

Fitting $P_3 = 468$ KB in smallest post of memory.

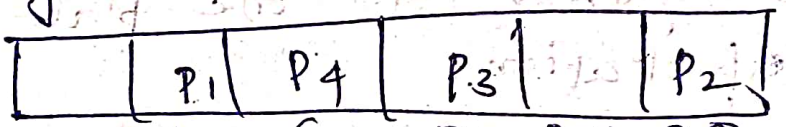
i.e



Here internal fragmentation = 500 −
= 32 KB

Fitting $P_4 = 491$ KB in smallest f
of memory.

i.e



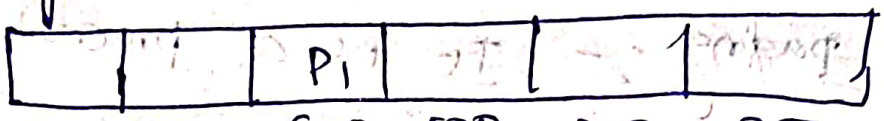
Here internal fragmentation = 600 −
= 109

Here all the proces are filled f
So, no other external fragmentation
Here, total internal fragmentation
93 + 40 + 32 + 109 KB
= 224 KB.

(iii) worst fit :-

fitting $P_1 = 357$ KB in high space in memory

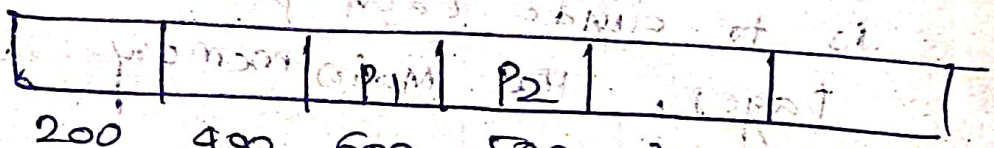
i.e



Here internal fragmentation = 600 - 357
= 243KB

fitting $P_2 = 210$KB in next high space in memory

i.e



here internal fragmentation
= 500 - 210 = 290 KB

for fitting P3, P4 external fragmentation is Required

Total internal fragmentation
= 243 + 290 = 533 KB

2) Explain briefly the concept of Segmentation with paging Scheme.

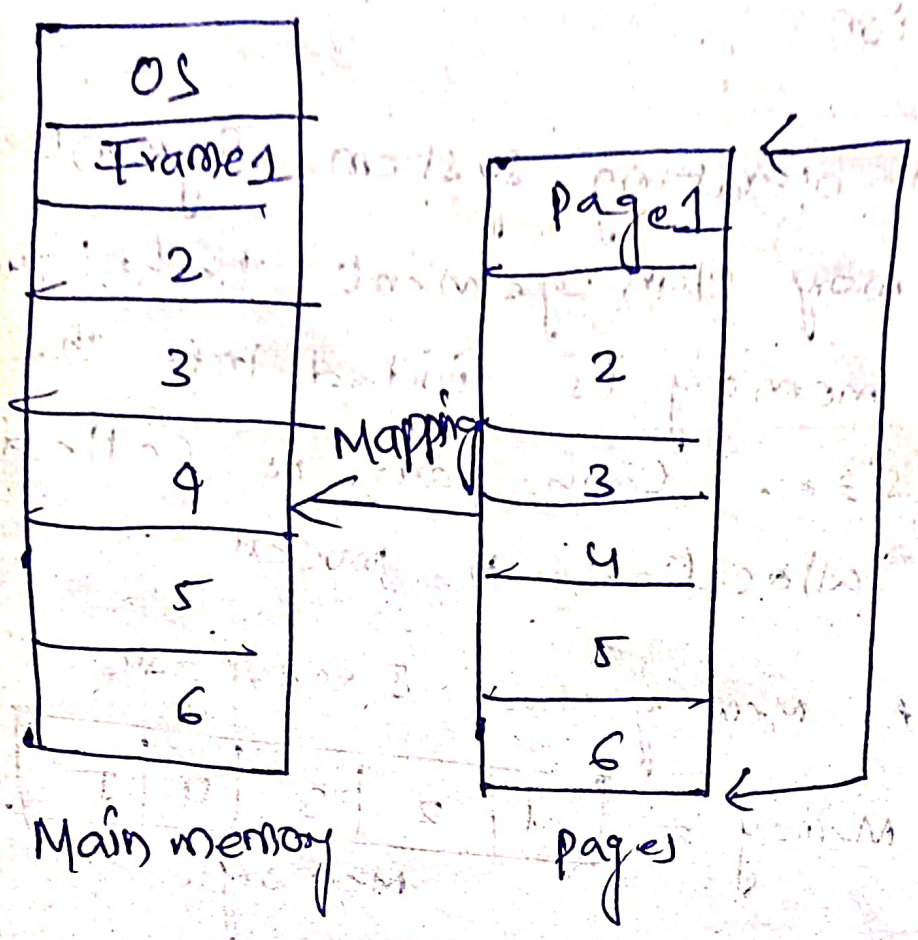
A) Paging :- It is a process of the Mechanism used to retrive process from the secondary storage to Main Memory in form of pages.

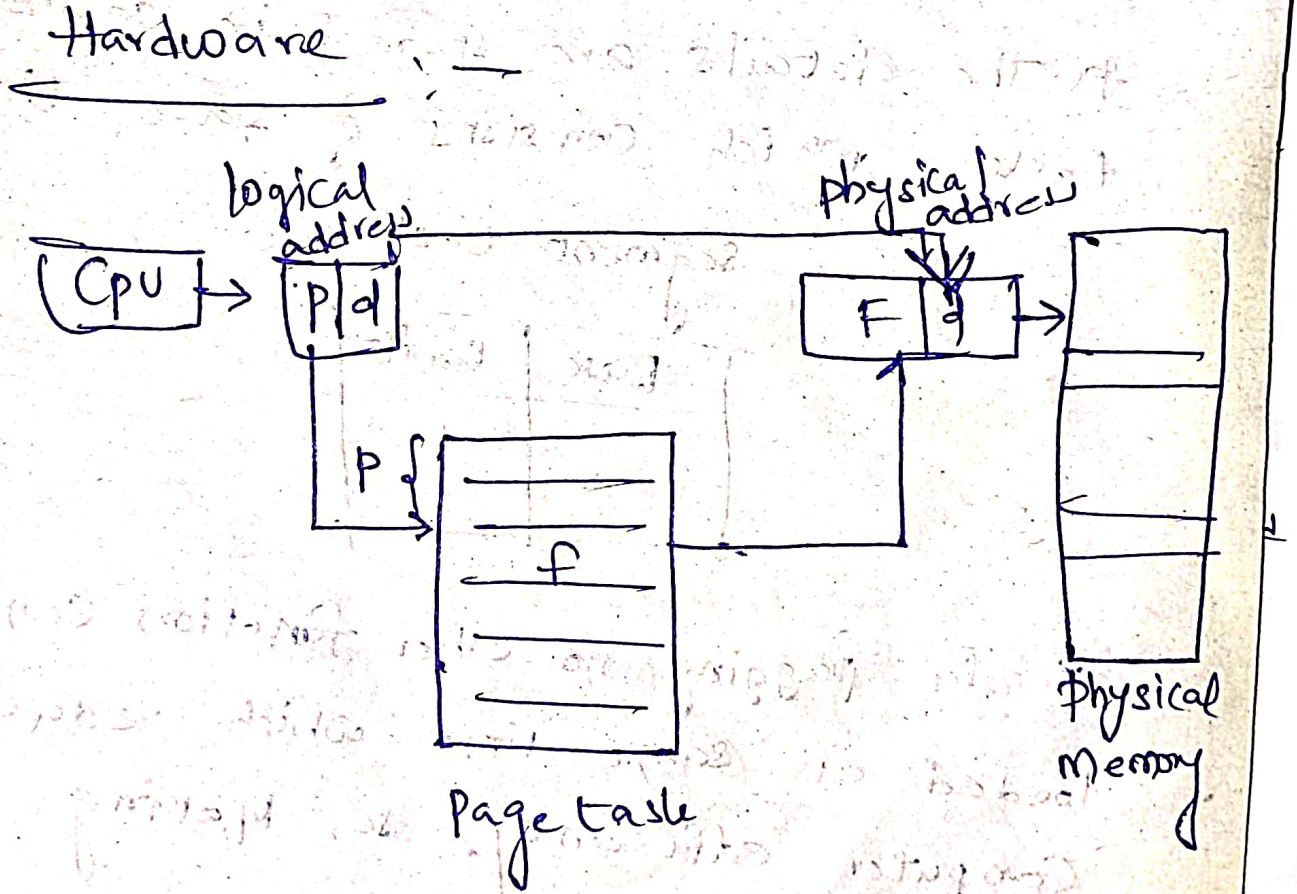
* The main idea behind the paging is to divide each process in form of pages, The main memory also is divided into frames.

* one page of the process is to be stored in one of the frames of the memory.

* Pages of the process are brought into main memory only when they are required otherwise they reside in secondary storage

* It is a non contiguous memory allocation technique


OS
Frame 1
2
3
4
5
6
Mapping
Page 1
2
3
4
5
6
process.
Main memory
Pages

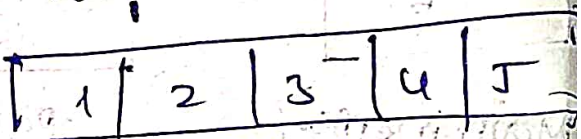

Hardware
Cpu
logical address
p
d
physical address
F
d
p
f
Page table
Physical Memory

Segmentation :-

In operating system, Segmentation is a memory management technique in which memory is divided into the variable size. Each part is called segment allocated to process.

Ex:- A Memory has 5 variable

Then Memory =



Memory

* The details are stored in segment table which consists of base, limit

Segment table

Base	Limit

* In pagging no other function can be loaded at same time which reduces Computer efficiency. So, Making process into segment can help to include all type of functions in one segment including main, other libaray into other segment

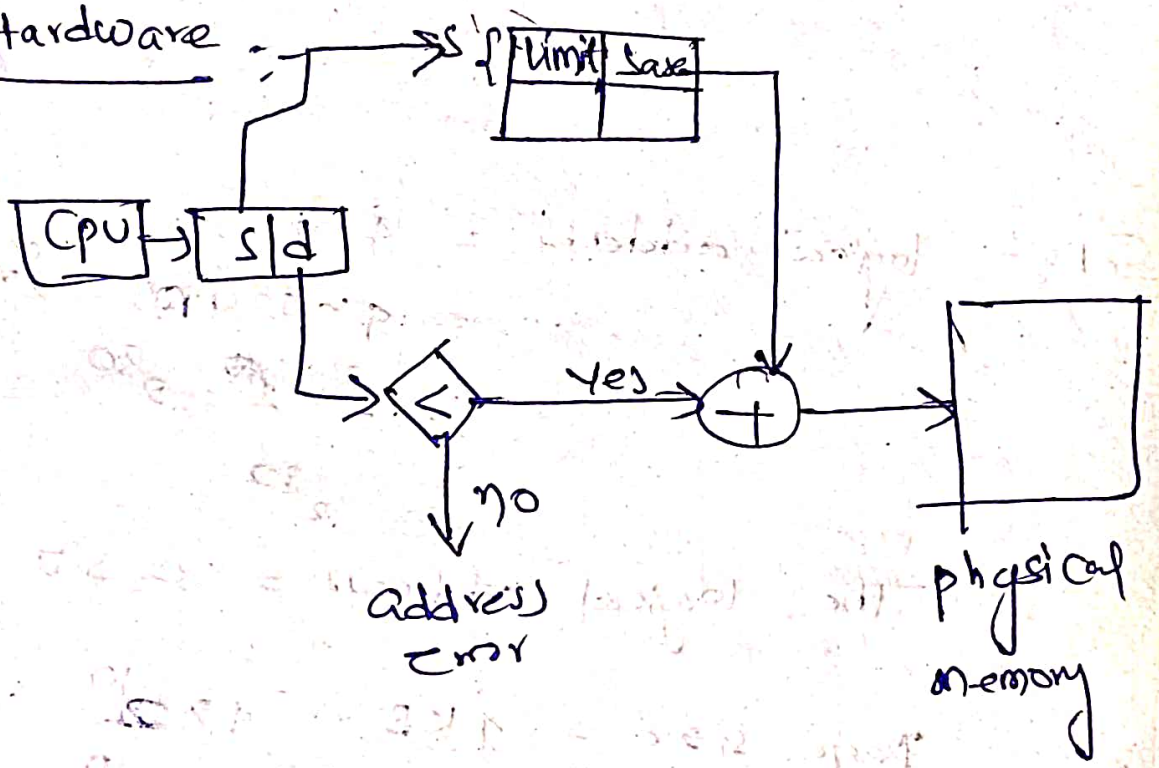
logical address :-

Cpu has two values in logical address they are :-

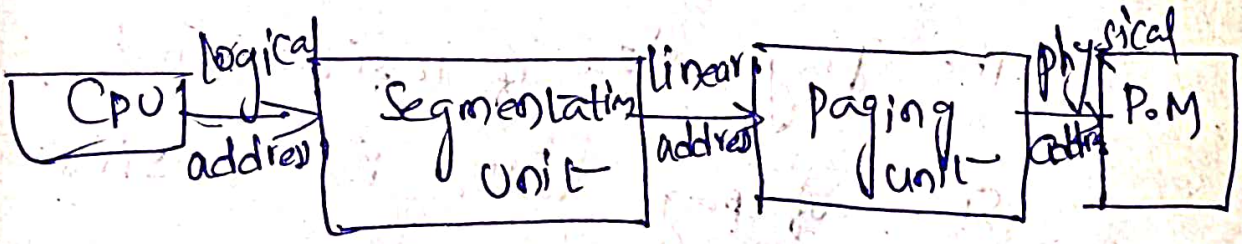
s : Segment number

d : offset

Hardware :-



pagging with segmentation :-



3) logical address space = 4 GB,
physical address space = 64 GB,
page size = 4 B,

(i) number of pages

Sol:- logical address = 4 GB
= 4×1024 MB = $2^{2} \times 2^{30}$
= 2^{32}

The logical address = 32 bit

Page size = 4 KB = 4×2^{10}
= $2^{2} \times 2^{10}$
= 2^{12}

The page size of 12 bits and size of page table is 32 bit

∴ number of pages = $\frac{\text{size of process}}{\text{page size}}$

= $\frac{2^{32}}{2^{12}}$ = $2^{32-12} = 2^{20}$

(ii) Number of frames :—

physical address = 64 GB

= 6 ex 1024 x MB

= 6 ex 1024 x 1024 x KB

= 6 ex 1024 x 1024 x 1024

= $2^6 \times 2^{10} \times 2^{10} \times 2^{10}$

= $2^6 \times 2^{30} = 2^{36}$

∴ size of physical address = 36 bits

frame offset = page offset

∴ frame offset = 4 KB = 4 x 1024

= 4×2^{10}

= $2^2 \times 2^{10}$

Number of frames = $\frac{2^{36}}{2^{12}} = 2^{36-12}$

= 2^{24}

(iii) Number of entries in page table

Number of entries in page table =

No. of pages in process = 2^{20}

parent

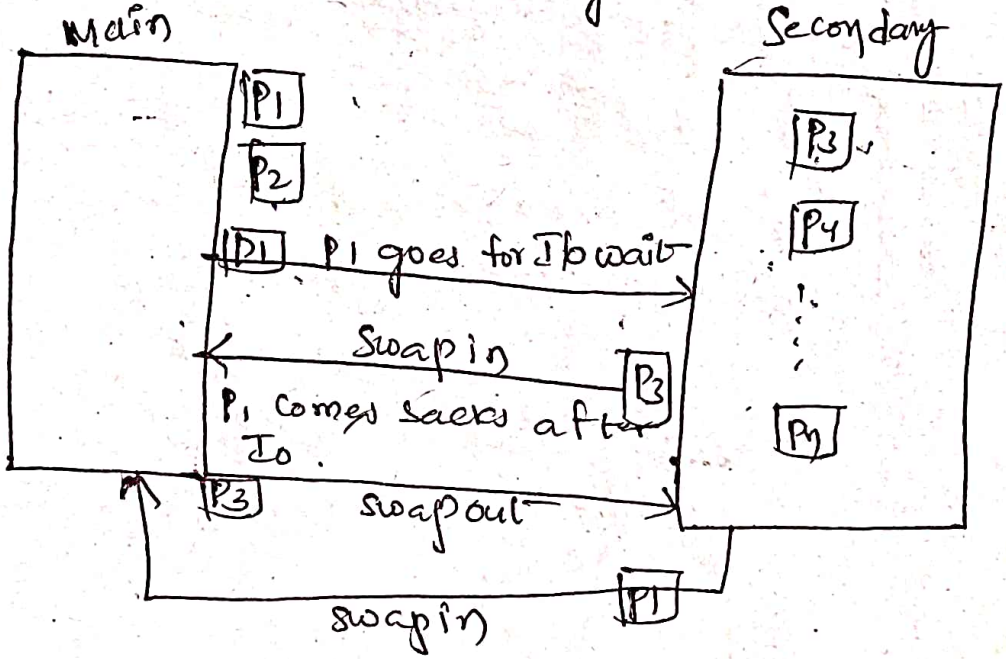
child

P.M

Unit-3

1) **Swapping** :- Swapp the data from Main memory to secondary memory and secondary memory to main.

" Swapping is a mechanism in which process can se swapped temporarly out of main memory to "secondary memory" later again swaps sacks to main memory



Variants of Swapping are

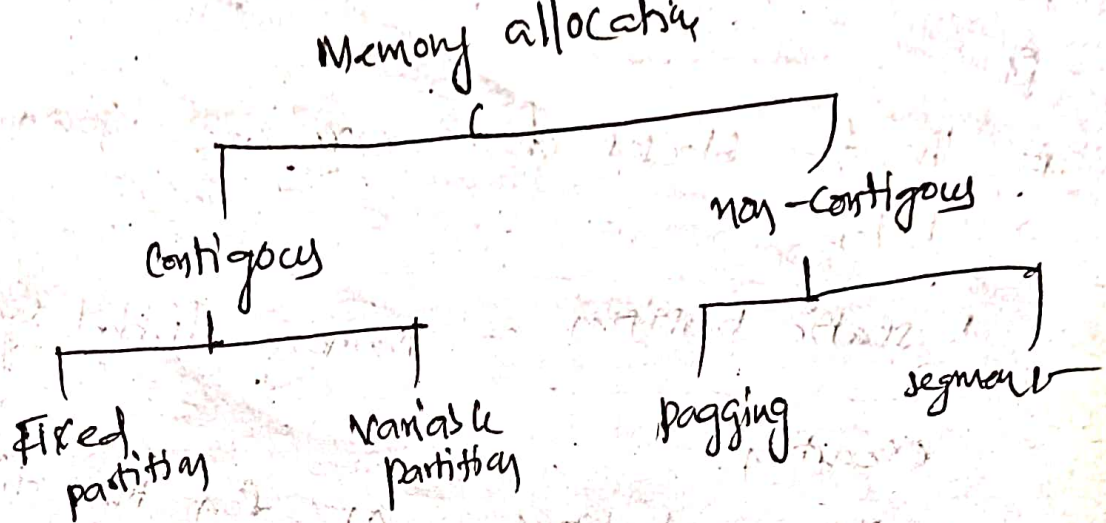
1. Swap in
2. Swap out

Whenever any Main memory swap in they it is temporarly stored in secondary due to size issue for execution after some time it swaps out.

parent — child

P.M

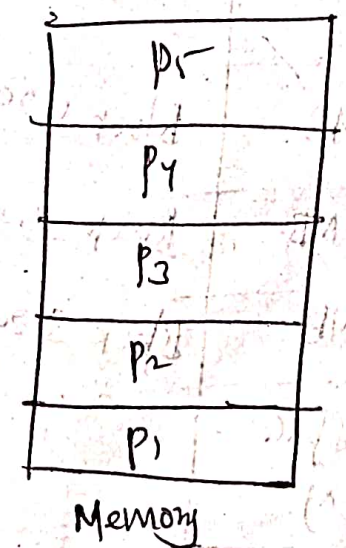
Contigous memory allocation



Contigous :- The storage of data/process in sequence called contigious memory allocation.

Fixed :- Memory is divided into several fixed size partitions. Each partition contains only one process.

let us consider 5 process are want to store in secondary memory. Then, Memory is divided into no. of process



When the process is completed then the memory is free space

whenever the partition is free, a proc
is selected from input queue & loaded
it.

* The free blocks of memory are known
holes.

1. static partitin :- Memory divided
execution

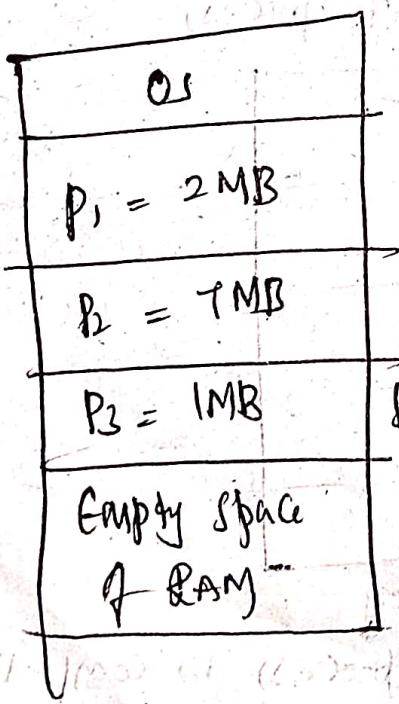
2. Dynamic partitin :- At run time
is dynamic partitin

Variable partitin :- at run time div
of storage called variable partition.

* Initially RAM is empty and memory
during run-time

* size = incoming processes

* No internal Fragmentation



3) Given page Reference string with 4 frames

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Compare the number of page faults, page fault Rate for LRU, FIFO, optimal page Replacement algorithm

Sol :— FIFO :- oldest will Replace

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
1		2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	3	3	3	3
2			3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6	6
3				4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2
					*	*					*				*			*	*	

Page Hit = "*"

No. of page HIT = 6

No. of page fault = 20 - 6 = 14.

Page fault Ratio = $\frac{14}{20} \times 100 = 70\%$.

page HIT Ratio = $\frac{6}{20} \times 100 = 30\%$.

LRU :- least Recently used :- Page which has not been used for longest time is Replace

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	1	1	1	1	6	6	6	6	6	7	7	7	7	1	1	1	1
1		2	2	2	2	2	5	5	5	5	5	3	3	3	3	3	3	3	3	3
2			3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2
3				4	4 (*)	4 (*)	4	4	4	1	1 (*)	1	1	6	6 (*)	6 (*)	6	6	6 (*)	6 (*)	6

No. of page Hit = 8, page Hit Ratio = $\frac{8}{20} \times 100 = 40\%$.

No. of page Fault = $20 - 8 = 12$, page fault Ratio = $\frac{12}{20} \times 100 = 60\%$.

Optimal page Replacement :- Replace the page that will not be called for long time. It has lowest page faults.

Frame	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
0	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
1		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2			3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3			4	4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6
					(*)	(*)			(*)	(*)	(*)	(*)		(*)	(*)	(*)	(*)	(*)	(*)	

No. of page Hit = 13 , page hit Ratio = $\frac{13}{20} \times 100 = 65\%$

No. of page fault = 20 - 13 = 7, page fault Ratio = $\frac{7}{20} \times 100 = 35\%$

Virtual Memory implemented using Demand paging

Whenever User wants to execute UGD pr
kar the strage in ~~secondary~~ Main memory. In UGD
the virtual memory takes place a crucia
role to execute the process.

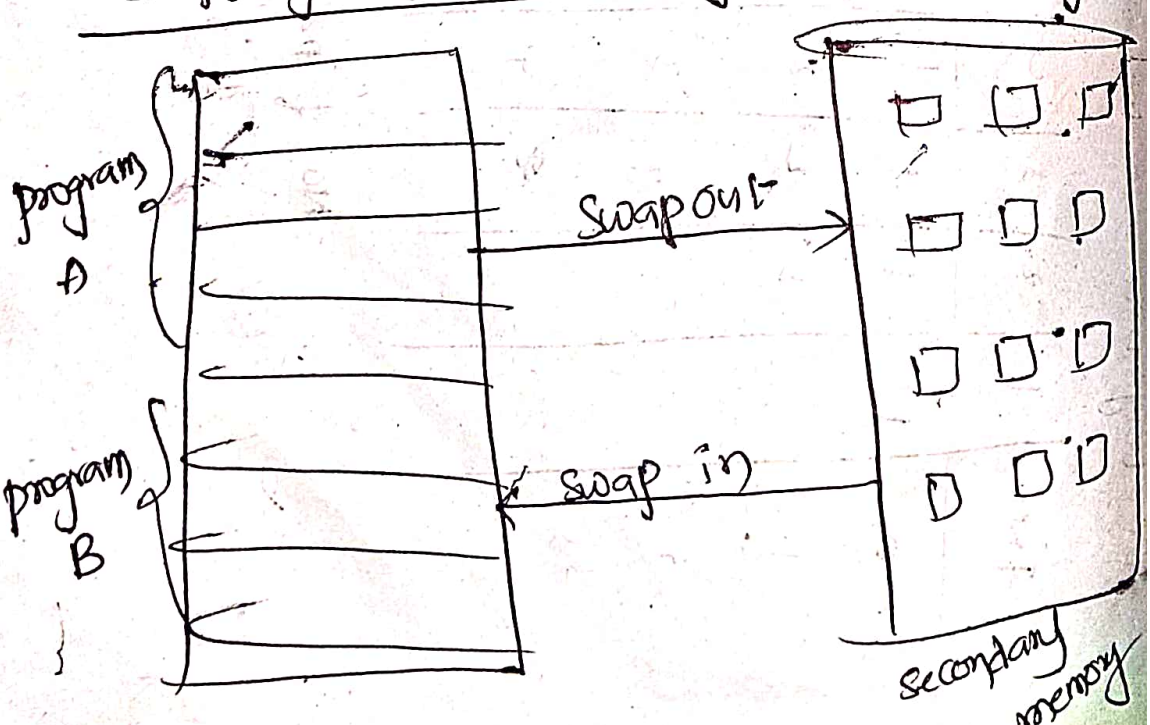
=) virtual Memory is implemented using
Demand paging

=) Demand paging is a paging In wh
the logical address are divided into e
number of partitions called pages.

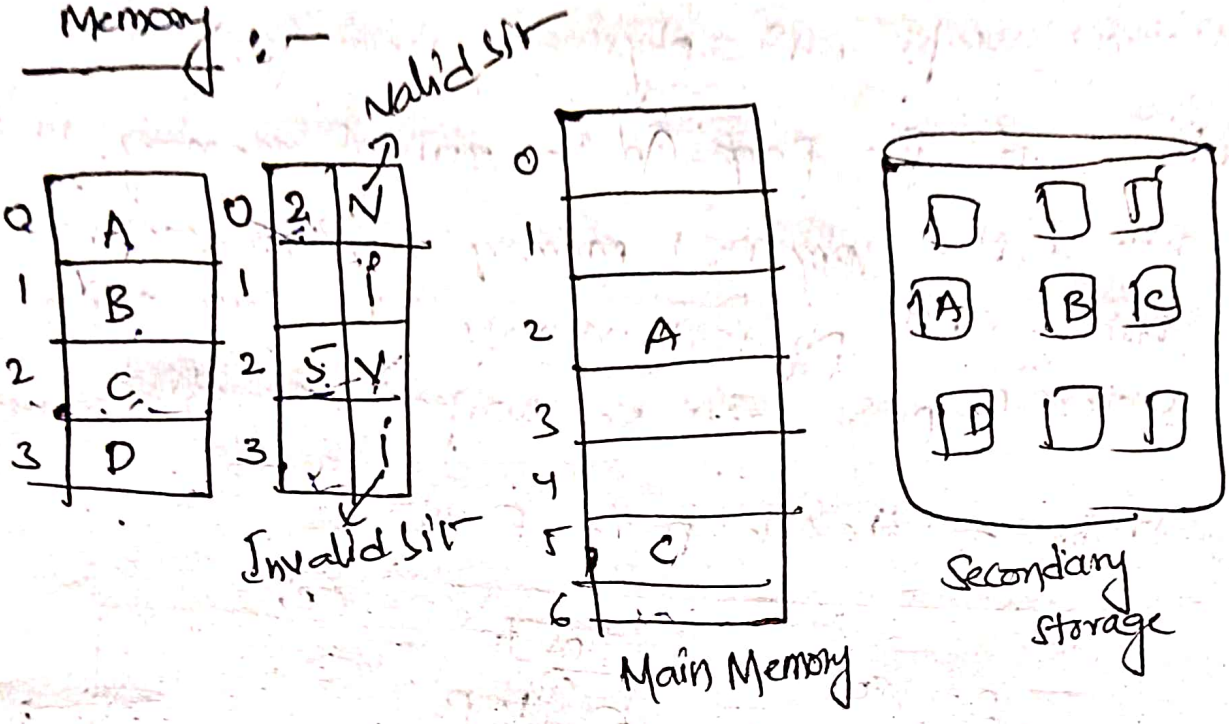
=) The Main Memory also divides equal
partitions called frames.

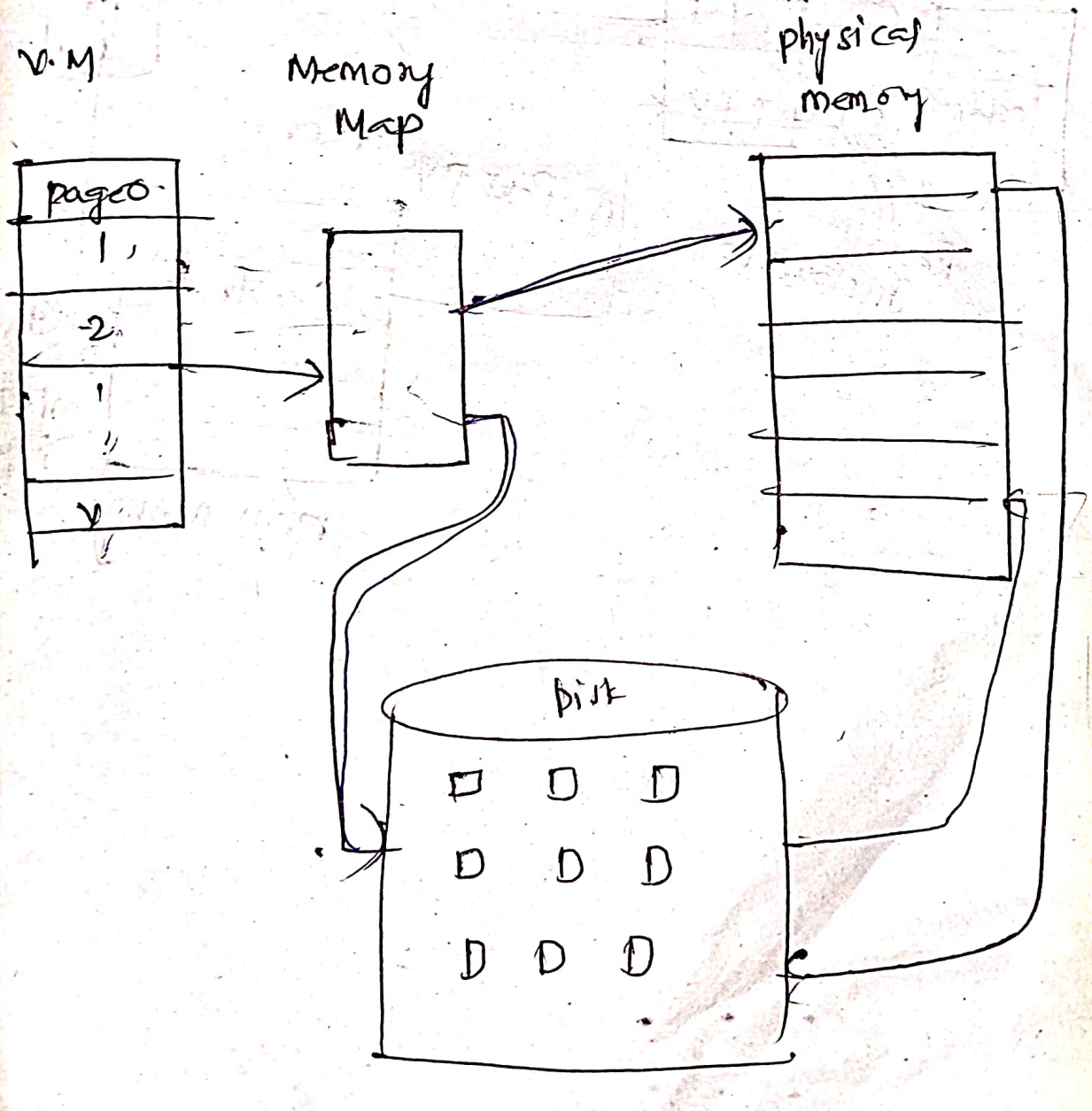
=) virtual memory is an imaginary memory
that is used to execute a process whose
capacity is greater the ~~secondary~~ Main memory

Swapping b/w Main memory and secondary M



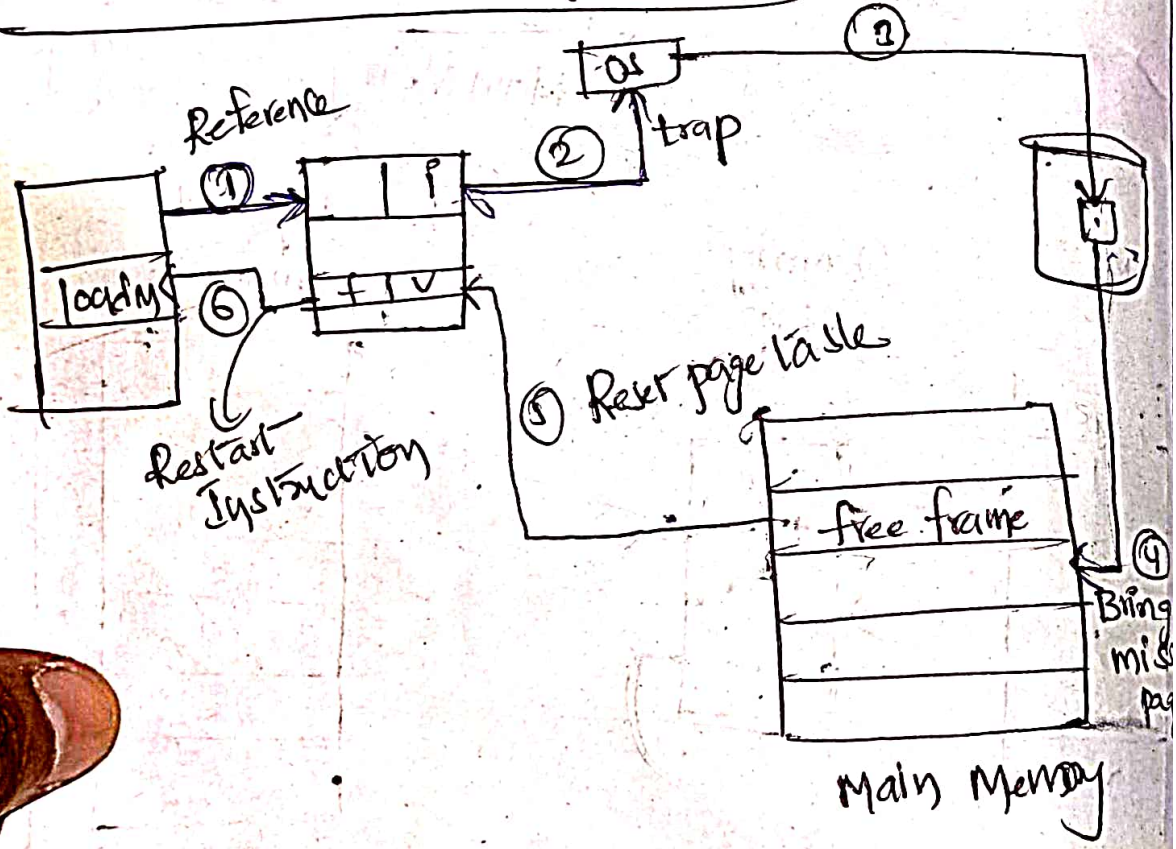
page table when some pages not in Main Memory :-





Virtual memory maps to memory and the memory maps with physical memory and disk. if the page in the virtual memory is not in the physical memory then it gets from disk. If physical memory is Full then unused page will be swaped to disk.

Steps to Handle page Fault :-



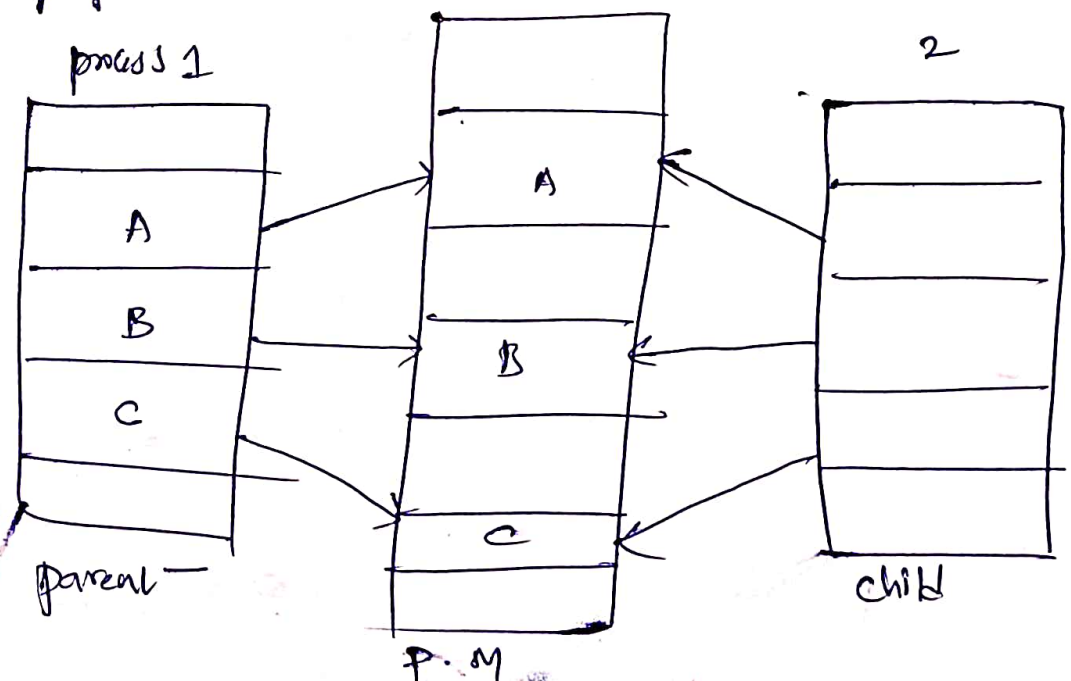
Copy on write

* fork() system call is used to create the child process from the parent process.
* copy of parent address is allocated to child address ss.
  * Duplicating pages belonging to parent.
* Many child process invoke exec() system call immediately after creation – copying of parent's address space may be unnecessary.

copy on write :–

allows parent and child processes initially to share same pages.

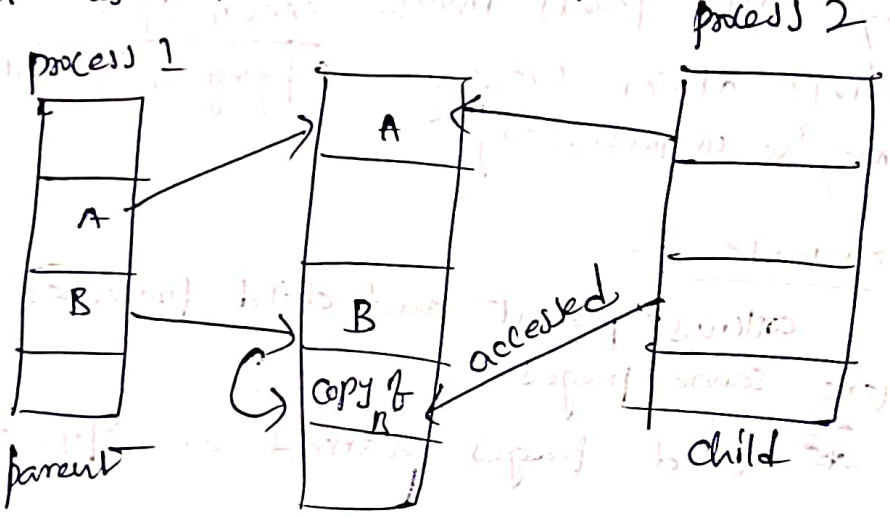
* shared pages marked as copy-on-write pages
* only pages that are modified are called as (Marked as Copy on write.
* pages that can't be modified – ex. pages containing executable code – can be shared by parent and child



* If either process writes to a shared page.
  - A copy of shared page is created
  - unmodified pages shared by parent and child process.

* If process want to create new page then new ~~page~~ frame want to create.

* Let us consider an example

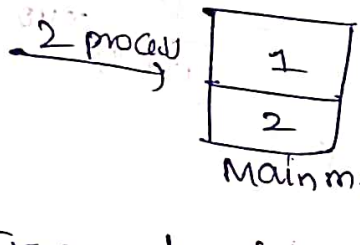


The process 2 Requires B then the process 1 contains B page and linked to physical memory that physical memory data OS will make a copy and shared to /accessed by process 2

Allocation of Frames

Whenever CPU wants to execute any set of process if it resides in main memory then the main memory is divided into equal no. of partions called Frames

Eg :- 2 process are executing by CPU then main memory is divided into two parts -



The Frames partition is divided into 3 approaches -

1. Equal allocation
2. propotional allocation
3. priority allocation
4. Glosal replacement

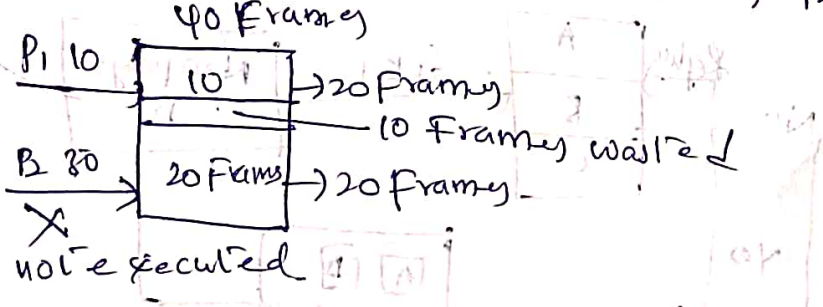
3. local Replacement -

1. Equal allocation :- The Frames are allocated equally in main memory

Eg :- $P_1 \rightarrow$ 10 Frames, $P_2 \rightarrow$ 20 Frames.

The size of main memory = 40 Frames

In Equal allocation P_1 allocated 20, P_2 allocated 20



Disadvantage :- wastage of frames can't be feeded for another one

2. Propotional Frame:- The Frames are allocated based on the size of the process

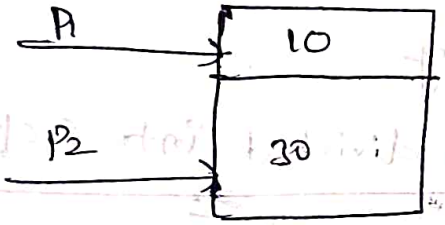
Eg:- P_1 = 10 Frames, P_2 = 30 frames, Size = 40 Fram

P_1 process Frames distribution = $\frac{P_1}{(P_1+P_2)} \times$ Si

$= \frac{10}{(10+30)} \times 40 = 10$ Frames

Similarly, P_2 Frames distribution = $\frac{P_2}{(P_1+P_2)} \times$ Size

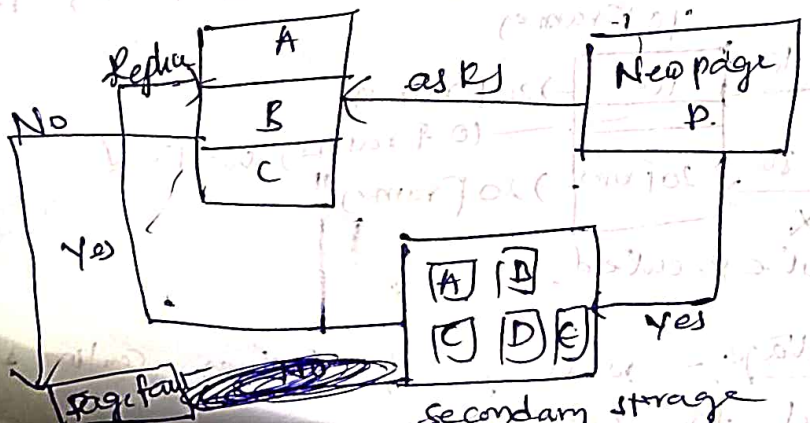
$= \frac{30}{(10+30)} \times 40 = 30$ Fram



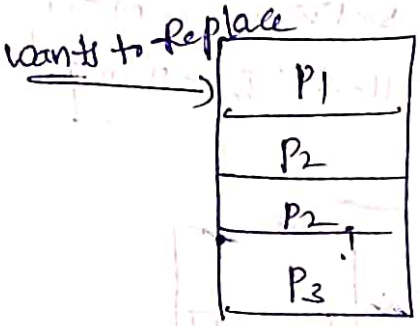
3. Priority allocation :- The high priority process wants to execute first then the low priority process. Based on priority the frames are allocated.

4. Glosal Replacement allocation :-

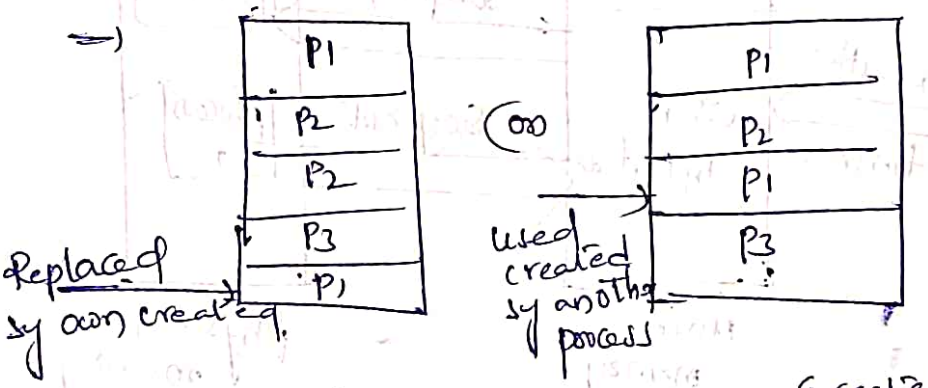
whenever the Frames the full then the new page wants to execute from the secondary storage if the page is available from secondary storage the Replacement happens else page fault occurs



process select a replacement frame from the set of all frames; or process can take a frame from another

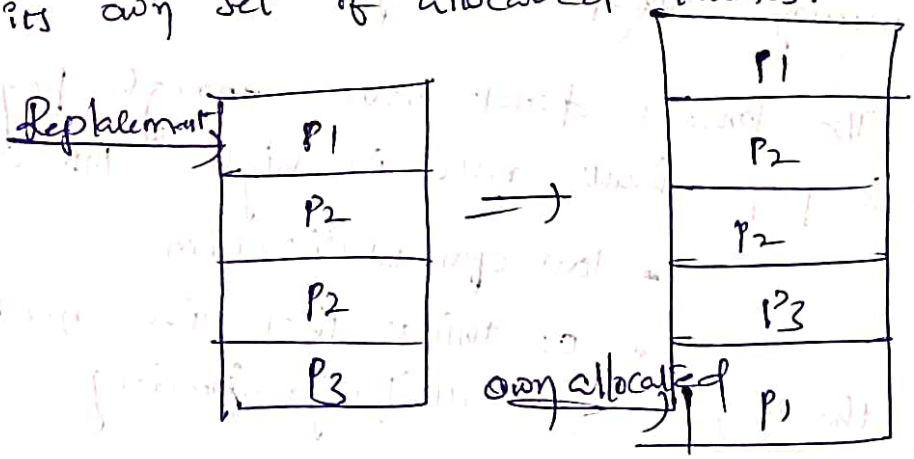


The P1 can take its own Frame creation (or already created frame



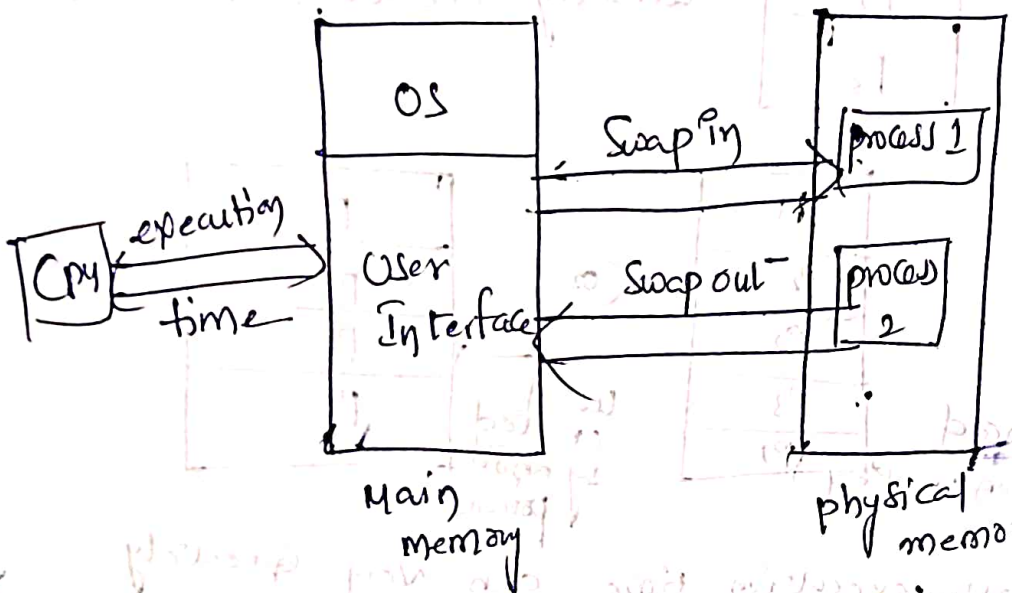
* process execution time can vary Greately
* Greater throughput are so common
  - The amount of work completed in unit time

local Replacement :- Each process selects from only its own set of allocated frames.



Thrashing ⇒ poor performance of virtual memory (or page

The cpu is executing the time more on swap in and swap out between main me and secondary memory called Thrashing

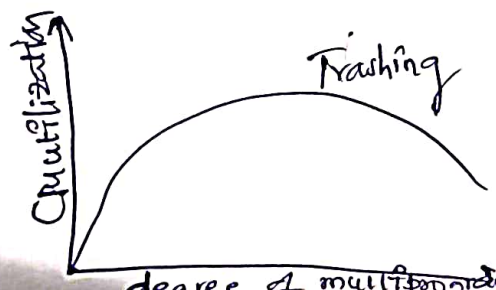


The above diagramtic Representation says abou thrashing

⇒ The processor spends more time on executi & swapping between main memory and secon memory called Trashing

⇒ The process don't have enough pages they page fault rate is high. This leads to

- low cpu utilization
- os thinks that it needs to incre the degree of multiprogramming
- another process added to the syste



The basic concept involved: if that if a process is allocated too few frames, then there will be too many and too frequent page fault. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.

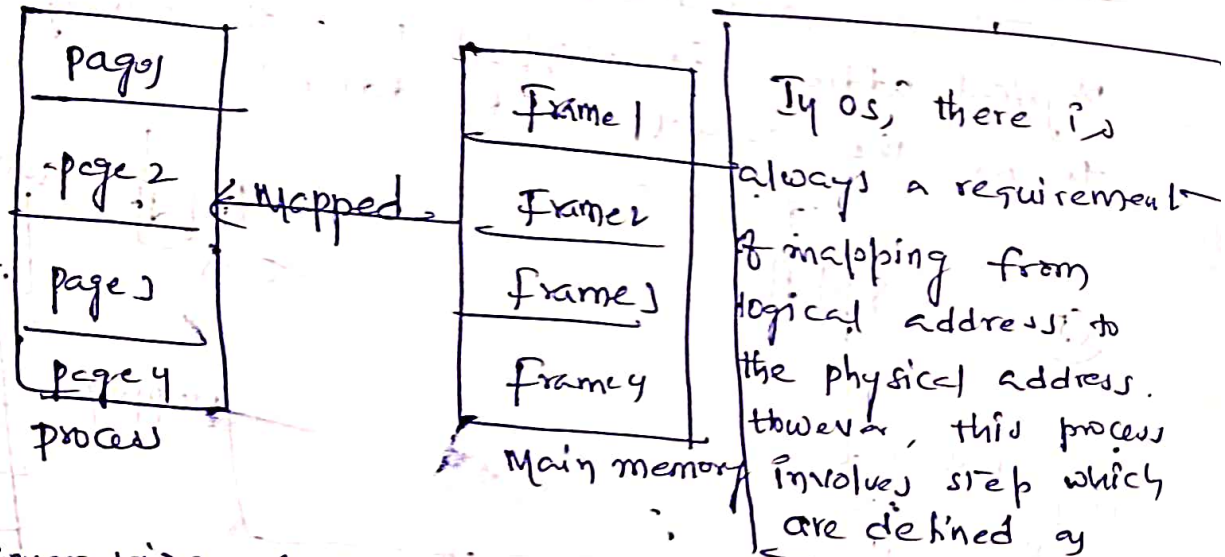
The long-term schedular would then try to improve the CPU utilization by loading so more processes into the memory, there by increasing the degree of multiprogramming.

Algorithm during Trashing :-

1. Global page Replacement :- Trashing will increase due to not enough Frames
2. Local page Replacement :- Thrashing will decrease due to selected Frames

Causes of Trashing :-

Memory Mapped :-



In OS, there is always a requirement of mapping from logical address to the physical address. However, this process involves step which are defined as

1. Generation of logic address :-

CPU Generates logical address for each page of process. This contains two parts

1. page number
2. offset

logical address

p	d

2. Scaling :- To determine the actual page numb. of the process, CPU stores the page table base in a register. Each time the address is generated, the value of page table base is added. to the page num to get the actual location of the page entry in the table. The process called as Scaling

3. Generating of physical address :-

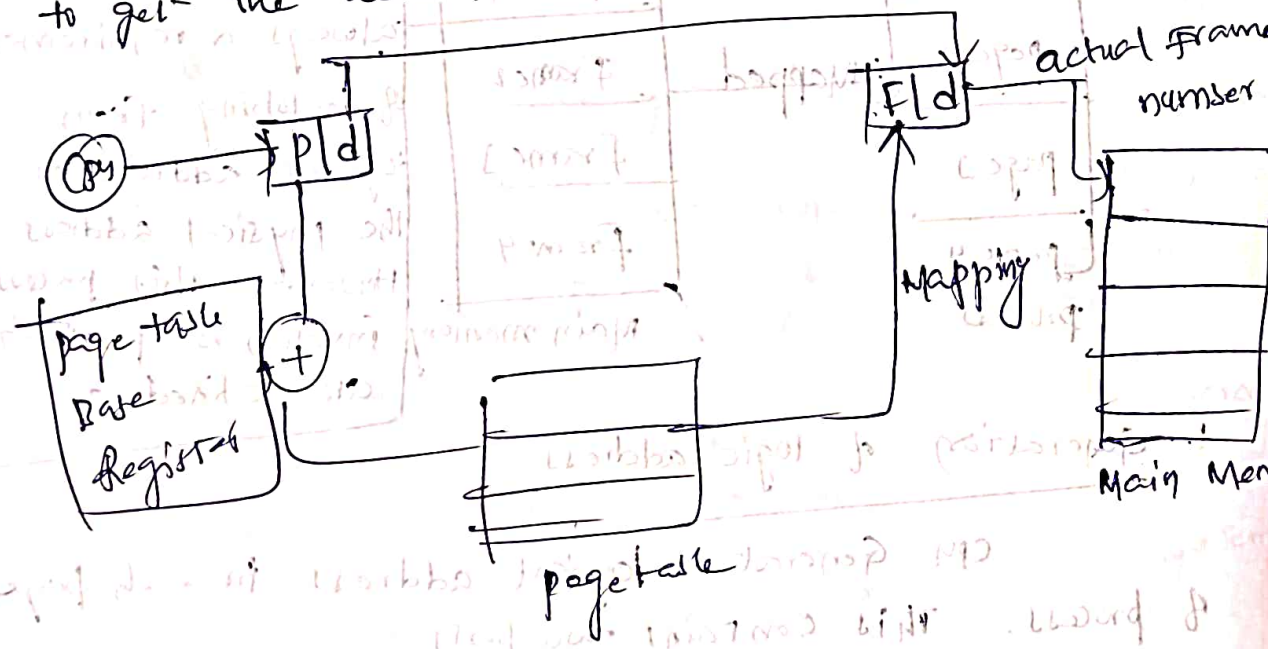
f	d

1. Frames
2. offset value

The page copied from logical address is str into page table. and that stored in Frame table

4. Generating actual Frame number :-

The frame number and the offset from the physical address is mapped to main memory in order to get the actual word address



Kernal memory allocation

User programs allocates in 2 approaches

1) Contigous Allocation
2) Non-contigous Allocation.

* The main memory is divided into 2 parts
  1) operating System (Kernel)
  2) User process

~~=> Every~~

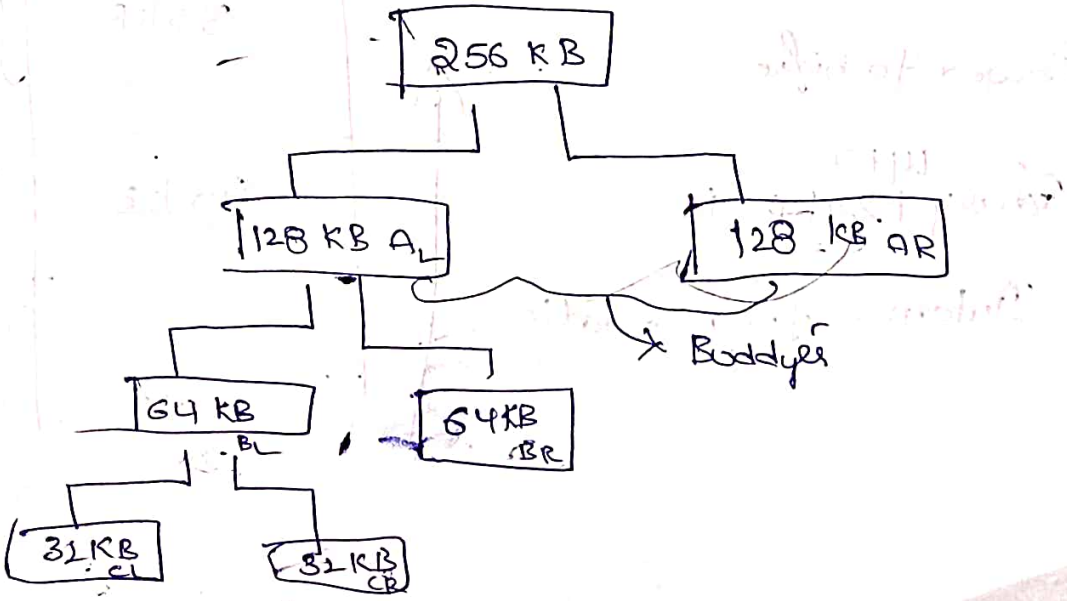
=> Kernel mainly stores process Descriptors, file objects, Semaphores

=> In order to allocate memory for the Kernal objects we have 2 approaches :-

1. Buddy System
2. Slab Allocation.

1. Buddy System :-

We use power of 2 allocation (Powers of 2)



=> Assume that the 21KB kernal object, and we have to allocate memory for it.

If we allocate 21 KB in 256 KB more memory

So it is classified then;

21KB in 128KB same process

like this it will goes upto 32 KB

$$32\,KB - 21\,KB = 11\,KB$$

less amount of memory is waste.

Advantage:- Coalescing.

=> Combine 2 into a large one.

disadvantage:- Internal fragmentation.

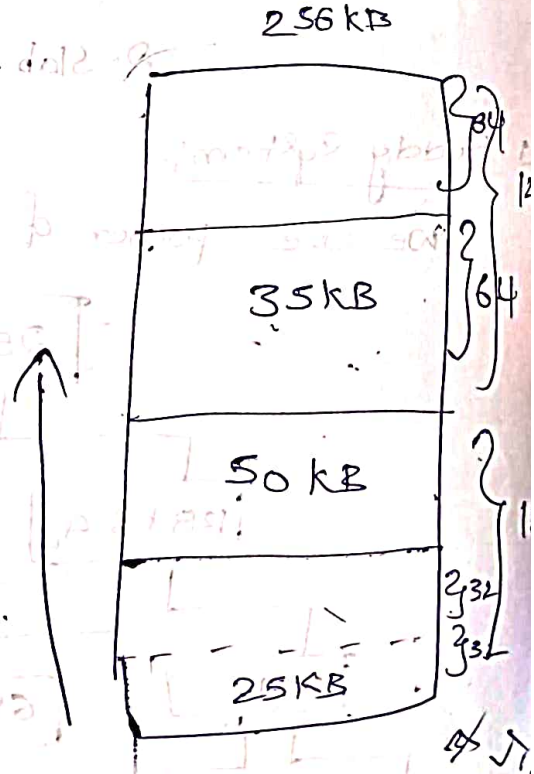
=> After allocating memory wastage of space form the above Example 11KB is Internal fragmentation.

Ex:-

Memory :- 25KB
50KB
35KB

Lower to higher

Only 29KB is 14+7 Internal fragmentation.



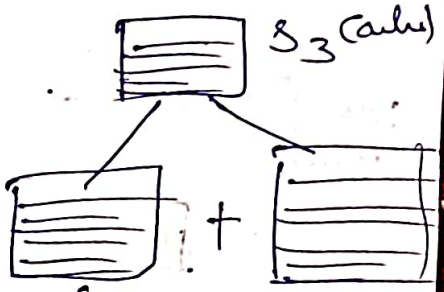
Slab Allocation:-

Kernal object

Semophores
file objects
process discripta

Slab means collection of contiguous of pages.

Cache means collection of Slabs =>



for each data structure there should be Separate cache.

for one cache process discriptors

another cache file objects

another cache Semaphore.

- Collections of different objects is not available to store in a single cache.

in cache

- Initially All the objects X are free.
- When ever req comes in the corresponding obj will stored.
- In Linux, Slab may be available in 1. Empty 2. full 3. partial. } States.

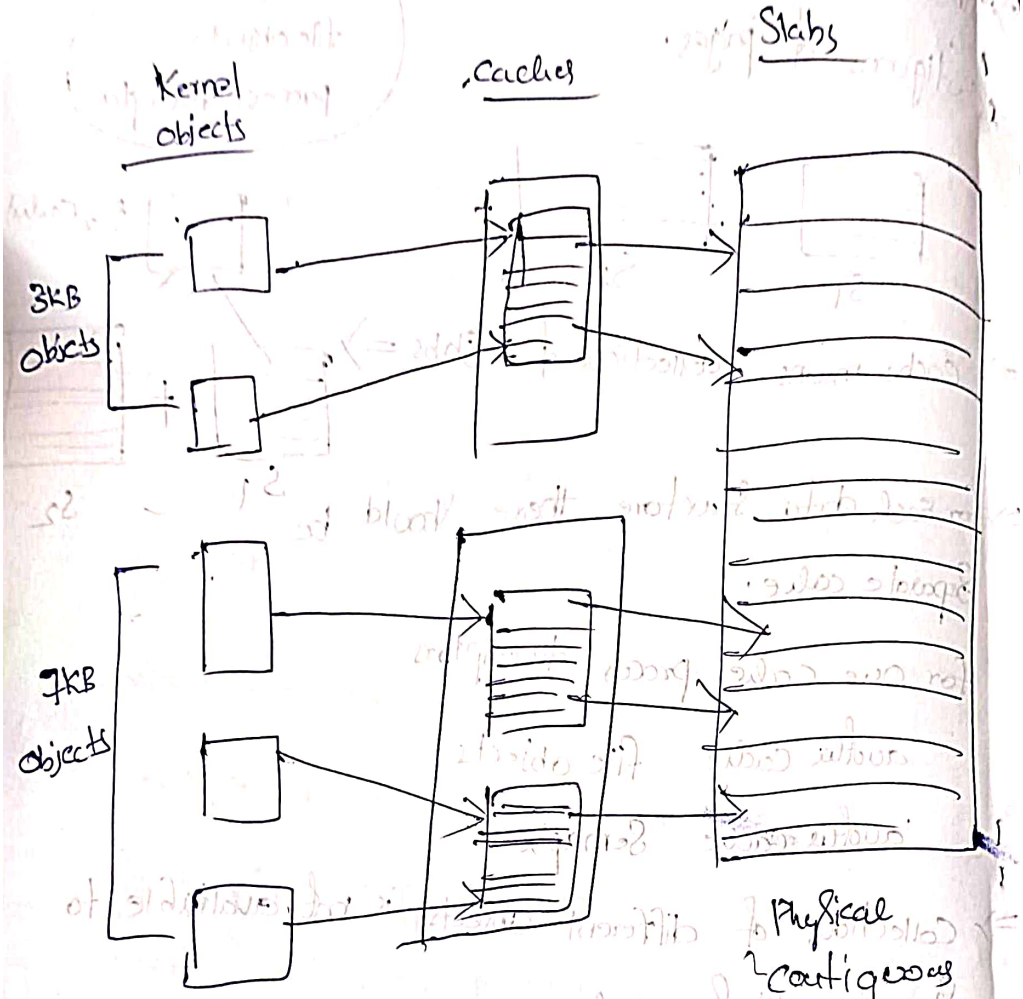
All the objects are free

All the objects are Used

~~All the~~

Some are free and some used.

When req arrives first it is ored to allocate in partial slab then Empty slab.



Advi- 1) No futernal fragmentation.

2) Kernel objects req allocates in Quicker manner.

OS - 4th unit

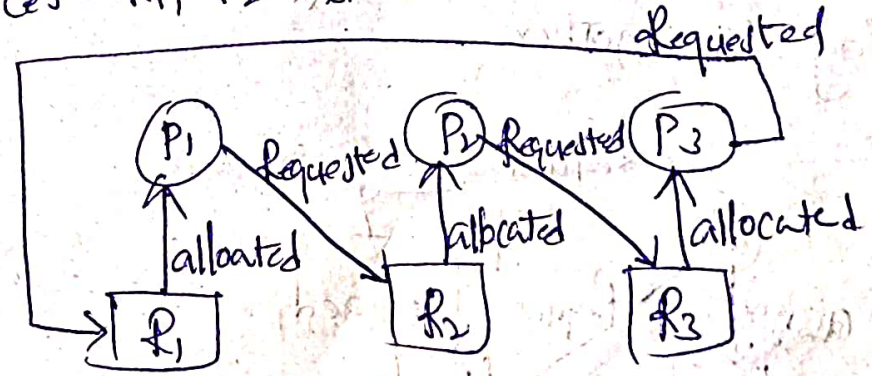
Deadlock :-

Every process needs a resource for its execution. However, the resource is granted in a sequential order.

1. The process requests for some resource
2. OS grants the resource if available otherwise let the process waits
3. The process uses it and release on the completion

Deadlock is a situation in which the computer processes waits for a resource which is being assigned to some other process. In these situation, none of the process is going to executed since the resource it needs.

let us assume that there are 3 processes P1, P2 and P3. There are three different resources R1, R2 and R3



This is called as deadlock

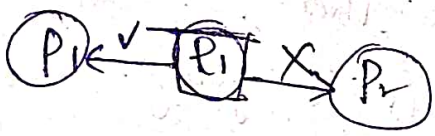
Conditions for deadlock :-

The processes are in

- Mutual exclusion
- Hold and wait
- No preemption
- Circular wait

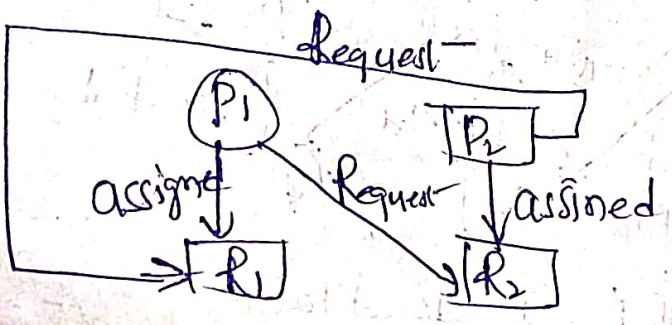
if all the 4 stages are happens simentaneo they deadlock will occur.

1. Mutual exclusion :- The only one proces can access and execute by one Resource a time. no other process can share the Res a process under execution.

Ex :- 

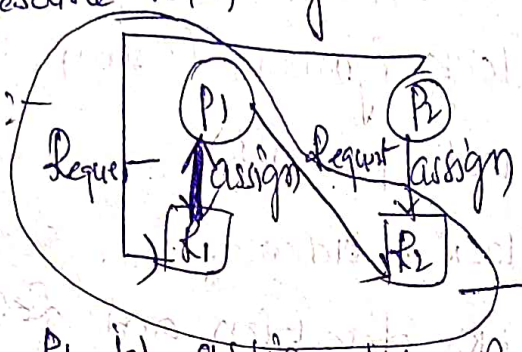
if P_1 is assigned by R_1 then P_2 can Request but not accessed.

2. Hold & wait :- Holding one resources an waiting for another.



3. No preemption :- A process can release the resource volunterly called as No preemption

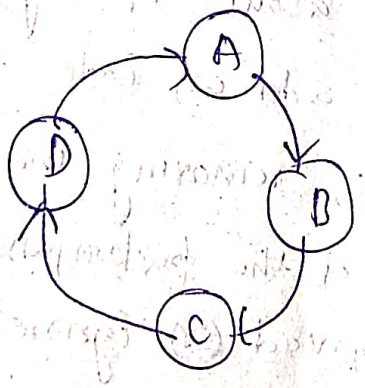
Eg :-



if P1 is assigned by Resource R1 and at time P1 request to R2 and P1 can't Release resource R1 then it is deadlock

4. Circular wait :- It is an process in which an process has assigned by Resources for execution and at same time the Resources of an particular process Requires to another processes

Eg :- Let us Consider 4 processes A, B, C, D



B Requires Resources which has assigned A, C Requires Resources which has assigned by B, D Requires Resources which has assigned by C. A Requires resource which has assigned by D

Various Methods to handle the dead lock

- Dead lock ignorance (Ostrich algorithm)
- Dead lock prevention
- Dead lock avoidence
- Dead lock detection and Recovery

1. Dead lock ignorance :-

The method says just ignore the dead lock. It is an rare condition if an dead lock occurs in sys then operating system automatically turn Restart and prevent deadlock.

* our OS like windows and linux, both use this strategy.

Eg :- we use windows OS, it consists of an certain code about management other than these we add an code to handle the dead lock, removing the dead it will be affected the performance. The OS can itself removed (or) ignore the deadlock itself if an additional co is written then it will be effected by the performance.

The deadlock is occurs then the OS automatically reboot the system. The deadlock happens 5 years - 10years at a single time.

→ This deadlock ignorance is achived from the Ostrich Method.

* Ostrich is a bird, whenever sandstorm it just put its neck into sand means it just simply say that there is no sandstorm. So it means trying to ignore the sandstorm.

→ Same as whenever a deadlock occurs then we simply ignore the data lock.

Why ignoring !-

→ we don't want to affect the performance.

Deadlock prevention !-

Before occuring the deadlock in an system we have to make an solution. There are 4 Mainly solutions are

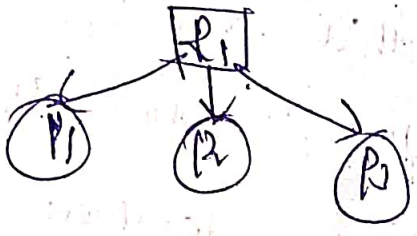
- Mutual exclusion
- No preemption
- Hold & wait
- Circular wait

If the system removes all the 4 conditions then deadlock can be prevented.

→ If above 4 conditions, any one the four can be fail then deadlock prevented.

1. Make Mutual exclusion false. –

Sharing an resource to differe processes at a same time

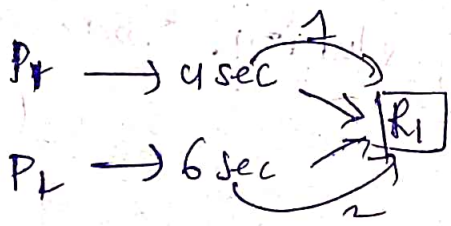


2. No preemption is false :–

By using time quantum Metho we make preemption is true

if P_1 = 4 sec time Quantum

P_2 = 6 sec time Quantum



3. Hold and wait is false :–

try to do no hold and w then deadlock can be prevented. i possible if all resources are alloc to the processes before execution start

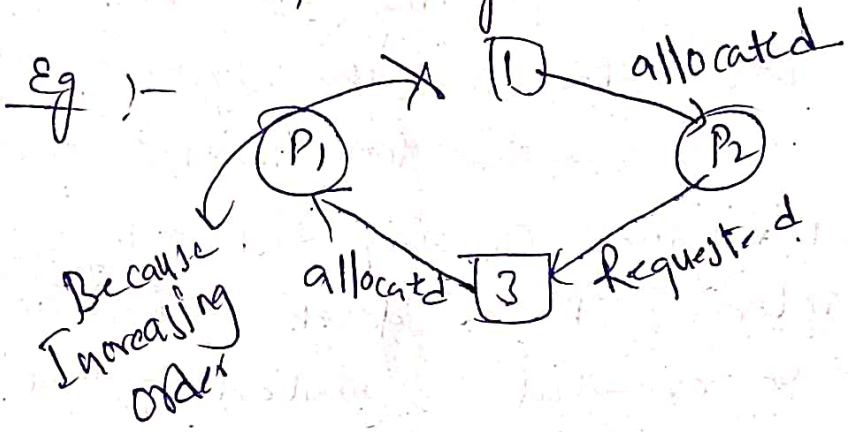
4. circular wait :-

To remove circular just give the numbering to all the resources.

Eg :- CPU - 1, printer - 2, Scanner - 3

So that the processor can Request the resources in increasing order

Eg :-



Deadlock avoidence (Banker's Algorithm) :-

It is a banker algorithm used to avoid deadlock and allocate resources safetly to each process in the computer system.

→ The 's-state' examines all the possible tests before deciding whether the allocation should be allowed to each process.

⇒ It also helps the OS to sucessfully share the resources between all the processors

→ The name bankers algorithm is named because it checks whether a person should be sanctioned a loan amount or not to help the bank system safely simulate allocation process.

Suppose the number of account holders in a particular bank is 'n', and the total money in a bank is 'T'. if an account holder applies for an loan; first, the bank susstracts the loan from full cas i-e available. if available is greater than loan then loan is approved

Similarly in operating system, when a new process is created in the system then it wants to give the full details about the Required resources, counting and delay. Based the creteria, the OS decides which process should be executed (or) waited so that no deadlock occurs in the system.

In Bankers algorithm, it requests to know about three things:-

1. How much each process can request for each resource in system. It is denoted by the [Max] system.

2. How much each process is currently holding each resource in the system. It is denoted [Allocated] system.

3. It Represents the number of each Currently available in the system. It denoted by [Available] system.

In Bankers algorithm, the major ste

1. process :- Required the proc

2. Allocation :- How many Resou to the process

3. Maximum :- The highest num are allocated to each process

4. Need :- How many Resou deadlock avoidence

Need = Max - alloc

5. Available (work) :- out of Need available resources

if Need ≤ Available

let us consider, There allocation of process are 3 resources A, B, C and

If A is 10, B is 5 an

Process	Allocation A B C	Maximu A B
P_0	0 1 0	
P_1	2 0 0	
P_2	3 0 2	
P_3	2 1 1	
P_4	0 0 2	

In Bankers algorithm, the major steps are

1. process :- Required the process for execution

2. Allocation :- How Many Resources are allocation to the process

3. Maximum :- The highest number of Resources that are allocated to each process

1. Need :- How Many Resources are need for deadlock avoidence

Need = Max − allocation

2. Availask :- out of Need how Many are availask resources

if Need ≤ availask Then Need = Need from

let us Consider, There are 5 processes and allocation of process are given and each 3 resources A, B, C and Maximum capacity of A is 10, B is 5 and C is 7

process	Allocation A B C	Maximum A B C	Availask Need	Need = Max − allocation
P0	0 1 0	7 5 3	3 3 2	7 4 3
P1	2 0 0	3 2 2		1 2 2
P2	3 0 2	9 0 2		6 0 0
P3	2 1 1	2 2 2		0 1 1
P4	0 0 2	4 3 3		4 3 1

Need is calculated by A consist of Maximum Resources = 10 and A occupies Resources for P_0 to P_5 = 0+2+3+2+0 = 7

∴ Available = 10 − 7 = 3

Similarly B = 5, B occupies = 1+0+0+1+0 = 2

∴ Available = 5 − 2 = 3

Similarly C = 7, C occupies = 0+0+2+1+2

∴ Available = 7 − 5 = 2

∴ Available for P_0 process =

A	B	C
3	3	2

process	Allocation A B C	Maximum A B C	Available	Need
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2	5 3 2	1 2 2
P_2	3 0 2	9 0 2	7 4 3	6 0 0
P_3	2 1 1	2 2 2	7 4 5	0 1 1
P_4	0 0 2	4 3 3		4 3 1

1) Need ≤ allocation

⇒ 332 ≤ 743 (X) check next

122 ≤ 332 (V) ⇒ Available = Available + Allocation

= 332 + 200

= 532

The execution at < P1, ... >

2) 532 ≤ 600 (X) Check next

532 ≤ 011 (V) ⇒ Available = 532 + 211

= 743

The execution at < P1, P3, ... >

3) 743 ≤ 431 (V) ⇒ Available = 743 + 002

= 745

The execution at < P1, P3, P4 ... >

4) 745 ≤ 743 (V) ⇒ Available = 745 + 010

= 755

The execution at < P1, P3, P4, P0 ... >

) 755 ≤ 122 (V) ⇒ available = 755 + 200

= 955

The execution at < P1, P3, P4, P0, P1, ... >

Safety algorithm :-

It is an algorithm which is used to check whether the system is in safety state (or) not.

⇒, There are two vectors work and finish of length m and n in algorithm.

1. initialize : work = available
   finish[i] = false (i=0,1,2,...

2. Check availability status of each resour
   Need[i] <= work
   finish[i] = false
   if the i does not exist then go step 4.

3. work = work + allocation(i)
   finish[i] = true
   Go to step 2 to check the status of resource availability for the next process.

4. if finish[i] = true ; it means that the system is in safety for all proces

Resource Request algorithm :-

1. if Request$_i \leq$ need$_i$ goes to (2)
2. if Request$_i \leq$ available, goto (3)
3. available = available - request

   allocation = ~~available~~ allocation + request

   need = need - request
4. check if new state is in safe state (or not

ex :-

process	allocation	Max	(work) Available	Need
P_0	0 1 0	7 5 5	3 3 2	7 4 3
P_1	2 0 0	3 2 2		~~1 2 2~~
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

if P_1 request (1,0,2), determine if it can (or not i.e $P_1 \rightarrow R(1,0,2)$

Need = Max - allocation = 322 - 200

= 122

① Request ≤ Need

102 ≤ 122 (V)

② Request ≤ available

102 ≤ 322 (V)

③ avail = avail - request

= 332 - 102 = 230

allocation = allocation + Request

= 200 + 102 = 302

need = need - Request

= 122 - 102 = 020

∴ The new table is

	Allocation	Max	Available	Need
P0	010	753	230	743
P1	302	322		020
P2	302	902		600
P3	211	222		011
P4	002	433		431

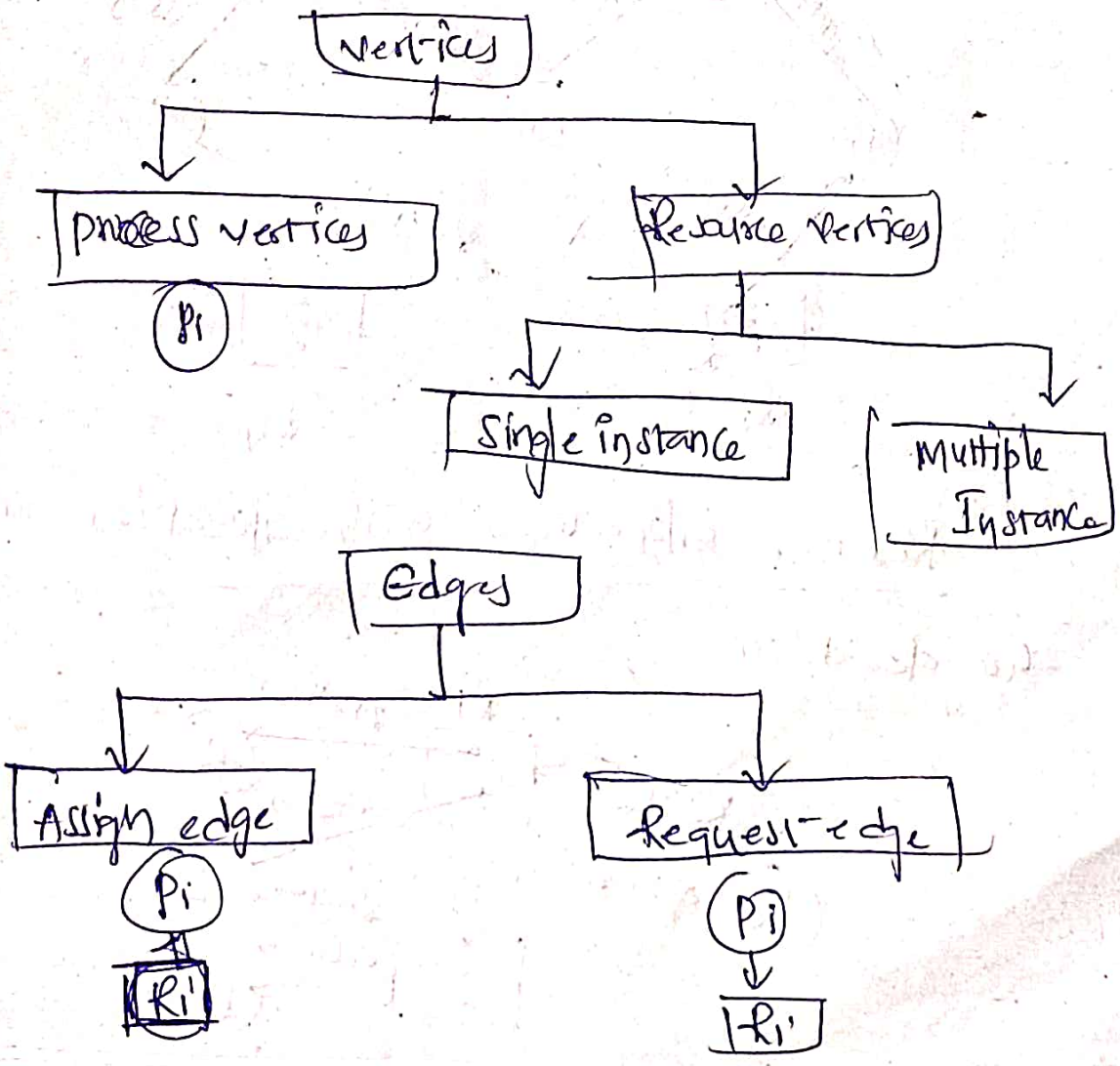
Next Continue with Sanjeev algorithm

(5) Resource allocation graph :-

The resource allocation graph is an pictorial representation of the state of a system. As its name suggests, the resource allocation graph is the complete ~~loop~~ information about all the processes which are holding some resources or waiting for some resources.

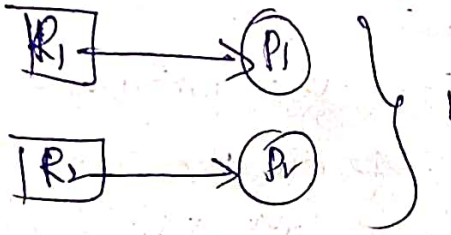
→ It also contains the information about all the instances of all the resources whether they are available or not.

→ In resource allocation graph, the process is represented by circle and the Resource by rectangle. Let us see types of vertices and edges.

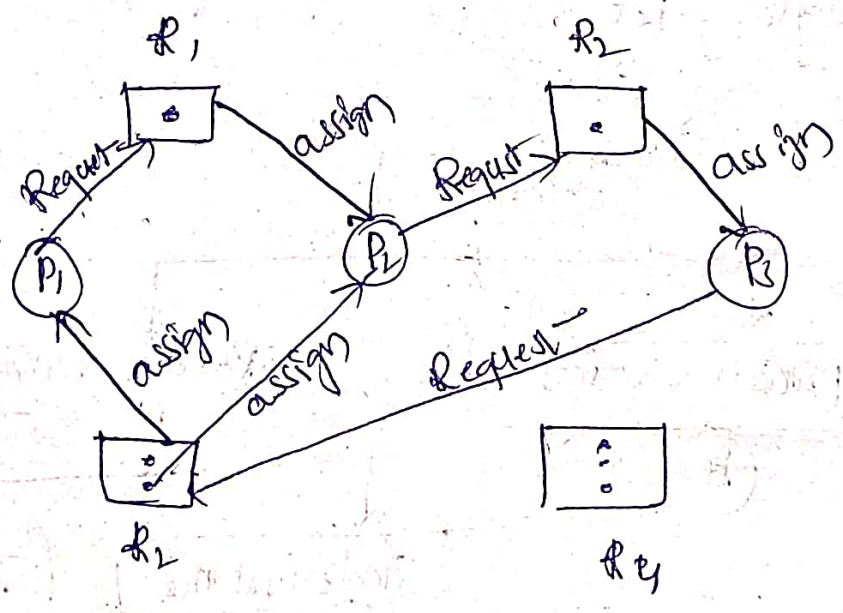


Resource allocation graph with deadlock

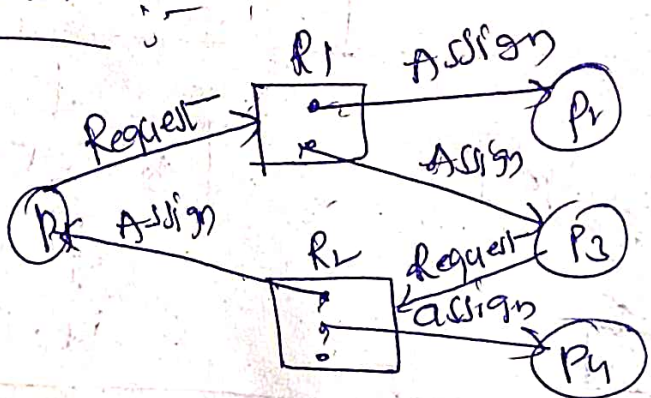
1. Whenever there is no cycle (Closed loop) in resource allocation graph, deadlock will occur.



2. Whenever there is an cycle then deadlock may (or) may not happen



Resource allocation with allocation but no deadlock :-



→ if no cycle no deadlock
→ if cycle deadlock may occur in single instance Resources
→ for multiple instance there is an possibility

Deadlock Recovery :-

- Process Termination
- Resource preemtion

- Process Termination :-

— Abort all the deadlock processes

Eg :- There are 5 processes out of 5 there are 3 processes are suffering of deadlock

i.e P_0 completes 95% of P_1 - execution P_1 - 92%, P_3 - 85% at this case too we detect deadlock then we abort

⇒ This make the time waste of CPU, execution processor etc., after abort we want to restart from starting

— Abort one process at a time until the deadlock cycle is eliminated

(i) we Abort/kill the processes based on priority

(ii) How much execution it does complete.

(ii) Resource preemption :-

* selecting a victim
* Roll back
* starvation

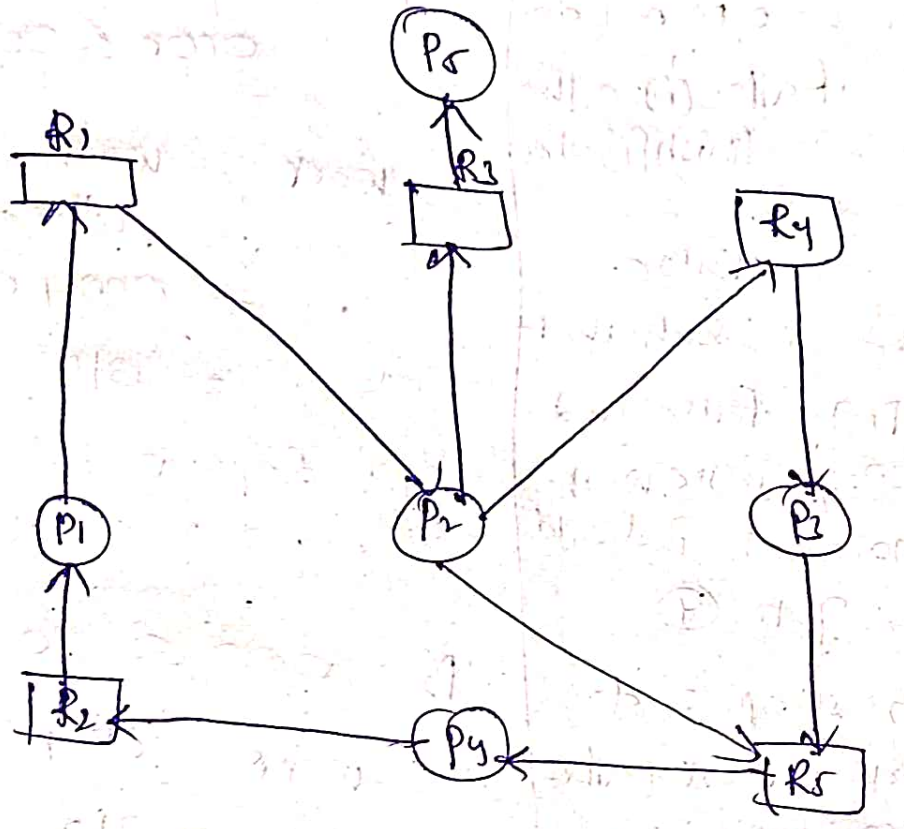
- Selecting a victim :- we need to ~~Correct~~ select correct process then we reduce the cost. we select based on less execution time and priority

- Rollback :- The Rollback is a process ~~Aborting~~ a process in deadlock and Rollbacking ~~start from~~ strather

- Starvation :- we selecting a same victim more time based on the execution and p makes starvation.

Deadlock Detection :-

Single Instance of each resource type :-



Multiple Instance of Resource type

A=7, B=2, C=6

process	Allocated A B C	Request A B C	Available
P0	0 1 0	0 0 0	0 0 0
P1	2 0 0	2 0 2	0 1 0
P2	3 0 3	0 0 0	
P3	2 1 1	1 0 0	
P4	0 0 2	0 0 2	

Steps

1. work = available
   for i = 0 to n-1 do
   if alloc(i) = 0 then
   finish[i] = true
   else
   false
2. Search an i such that
   finish[i] = false and
   Request(i) ≤ work
   if no such i should
   be found, goto (4)
3. for i in step 2 do:
   work = work + alloc
   finish[i] = true
   go to (2)
4. if finish[i] ≠ true for
   a some i then
   the system is in dead
   lock state
   else
   The system safe

000 = 000

P0 : Request ≤ wor
0000 ≤ 000

work = work +
allocat
= 000 + 01
= 010

P1 : Request ≤ wor
200 ≤ 010

P2 : 000 ≤ 010

work = 010 + 3
= 313

P3 : Continue th
process)

<P0, P2

File systems

Introduction :- File system is the part of the Operating system which is responsible for file Management.

⇒ It provides a Mechanism to store the data and access to the file contents including data and programs.

⇒ Some Operating systems treats everything as a file for example Ubuntu.

The File system takes care of the following issues

1. File structure :-

We have seen various data structures in which the file can be stored. The task of the file system is to maintain an optimal file structure

2. Recovering free space :-

Whenever a file gets deleted from the harddisk, there is an free space created in the disc. There can be many such spaces which need to be recovered in order to reallocate them to other files.

disk space assignment to the files :-

The major concern about the file is deciding where to store the files on the disk. There are various disk scheduling algorithms

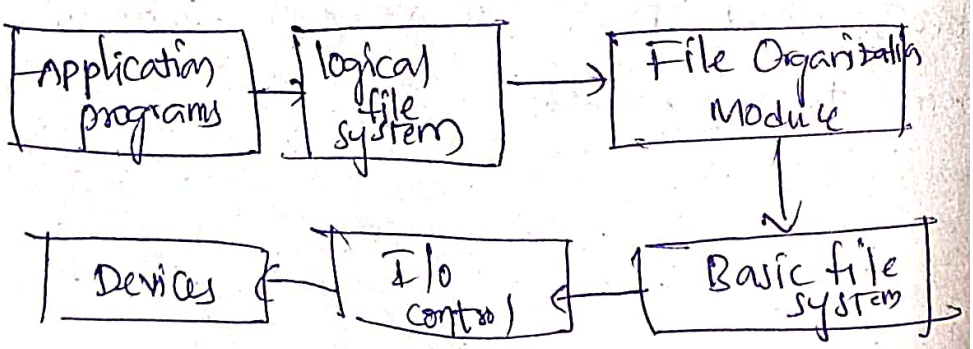
tracking data location :-

A file may (or may not be stored within only one block it can be stored in non contiguous blocks on the disk. We need keep track of all the blocks on which the part of the file reside.

File system structure :-

The file system work is to store files, locate the file and retrive the file

⇒ Most of the OS are used layer approach method to store the files. In each layer The activities of an file system happens.



→ Application program asks the file, the Request about the file goes into the logical file system and checks in it

→ logical file system consists of an file meta data and directory structure. if an application program requested data is not available at logical file system then it shows error

⇒ Generally, the files are divided as the logical blocks. These are stored in hardware to retrive in future

→ To store (or retrive the files in the hardware then the logical blocks must mapped with the physical block

→ These mapping is done by the File Organization module. These are responsible for free space management.

⇒ if files are created and deleted from harddisk then there is an gap/space at hard disk. if we created a new file and stores in harddisk at the position of gap and gives a path to store

→ once File Organization Module is decided then the information is passed to the basic file system.

→ The basic file system has commands that are sends to the input output control

→ The i/o control contains code which has to assign / access from the hard disk. The codes are called as drivers.

File :-

A file is an collection of related info stored on the secondary storage device

" A file was a collection of records a Each record consists of a field and e field represents some data"

Eg :- student data

SNO	SName	avgmarks
1	S	99
2	C	100

⇒ A file may be a object file, executa file, etc.

File attributes :-

1. Name :- Every file has name. inorder to perform an operation we must load file they we wanna specify the name. A name can't be same at same location. It is unique.

2. Identifier :- Whenever a file is created, interally a number is allocated to that file. Operating system operates the file using that identifier

3. Type :- There are so many types of files like doc file, jpeg file, texfile, obj file, c file, execultable file and other files. To specify type there is an extension.

4. location :-

The path of the file in which it is stored in the hard disk. So that we can retrive that file.

5. Size :- It specifies the size generally in bytes. if file is created it size is 0 bytes and then we add data into the files than the size increases

6. protection :- protection is an permission. Read permission, write permission, execatable permission

7. time, data & User identification

It Specifies that time, da
the file creation.

File operations :—

In order to perform operations
the file we uses various system Ca

1. create() :- To create a file we
create() system call. Generally the cr
file size is 0 KB and then we add da
init

2. Open() :- To open() a file from
Hard disk we want to create a file
stored at specific location and then
retrive that file and open()

3. close() :- In order to close the
we uses close() operation

4. write() :- To write an data int
file we uses write() operation

5. read() :- In order to read the file
we uses read() operation.

6. Reposition within file :- if any modification are take place in the file like changing the positions this are used. It is also called as seek() Method.

7. Delete() :- In addition to the content deletion, we use delete() Method. The attributes are deleted

8. Truncate() :- Truncate() Method is used to delete the file content only. The attributes are remains same

9. Append() :- To add the data at the last of the file we use Append() method.

10. Copy() :- This method is used to copy() the content from one file to another file if no protection is given.

11. Rename() :- This operation is used to Rename() the file name with other file name.

File types :- There are several types of files such as executable file, object file, text file and other others. These all files are differentiated via Name and extension to them. The extension is denoted via (.).

File types

1. executable :- The file is ready for execution. The extension is .exe. These file is loaded from harddisk to Main memory for execution.

2. Object file :- These file is created when we compile an C program. The extension of the file is .obj. The format is '0' and '1'

3. Batch file :- The batch is an example of shell programming which is written in unix and linux. The extension is .sh.

4. Text file :- The text file is created via notepad with extension .txt and via word with extension .doc.

5. word processor :- The extension of a word processor not only .doc but also .wp, .tex, .rtf etc.

6. Library file :- These files are useful in linking. The extensions are .lis, .dll

6. print (or view) :- In order to view the contents of the file we use .pdf (or) .jpg (or .png etc.)

7. Archieve :- These files consists of an group of files while we extract all we see the contents The extensions are .zip, .arc, etc.

8. Multimedia :- These files are used to listen (or) watch. The extension of the files are .MP3, .MPY, .avi

1) Sequential File Organization :-

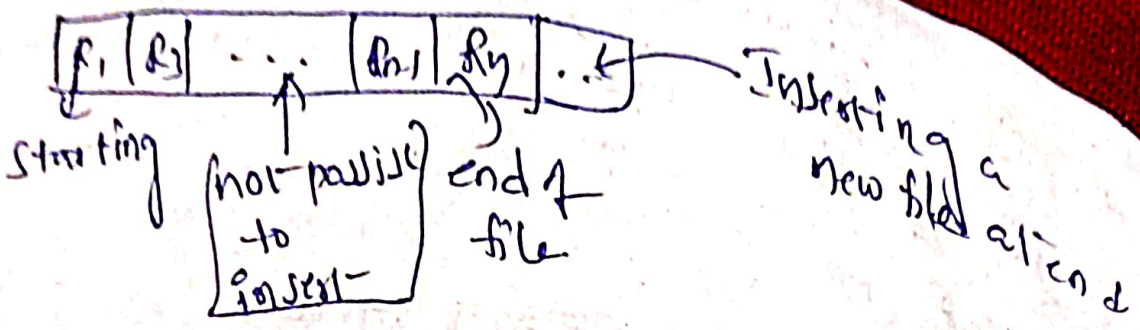
→ The easiest method for file Organization is sequential method.

→ In these method, the files are stored one after another in sequential manner.

→ The records are arranged in the ascending (or) descending order of a key field.

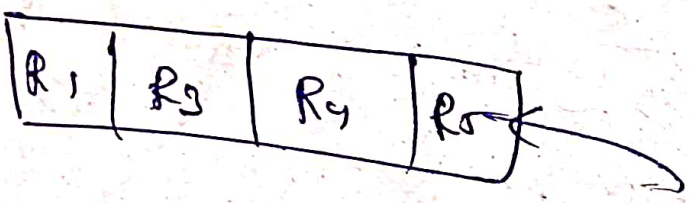
⇒ sequential file search starts from the beginning of the file and the records are added at the end of the file.

⇒ In sequential file allocation, it is not possible to add a record in the middle of the file without rewriting the file.

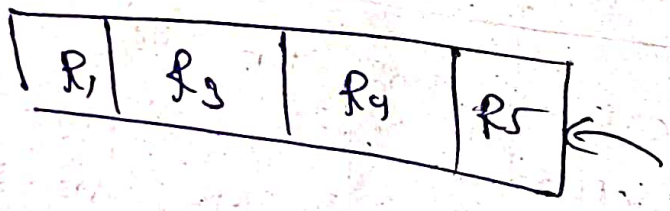


Inserting new Records in the file allocation

1. Inserting new Record



2. Sorting :-



⇒ Sorting is done on primary key (or any other key

id :-

Redundant array of independent disk

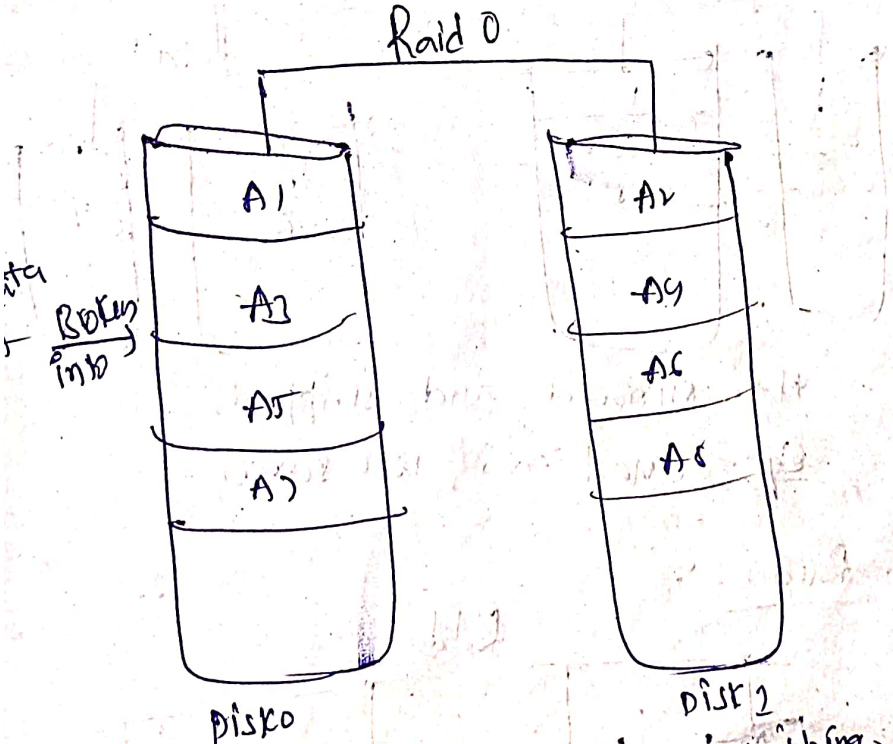
(or)

Redundant array of inexpensive disk

Here there is an duplicate files are ailable in the disk (stored the files). there e disk is an independent because we can ivide a single disk into 5 local disk (drives) and the cost of the disk is comparatively less

Different levels of RAID :-

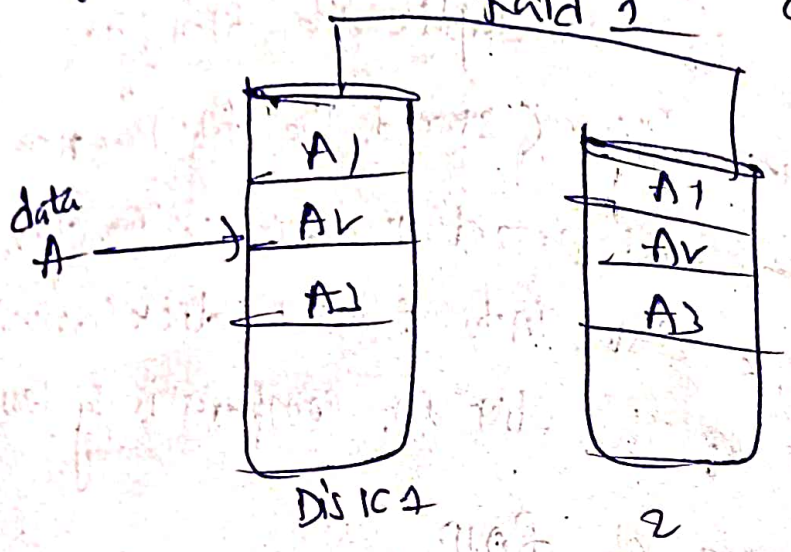
1. RAID 0 :-



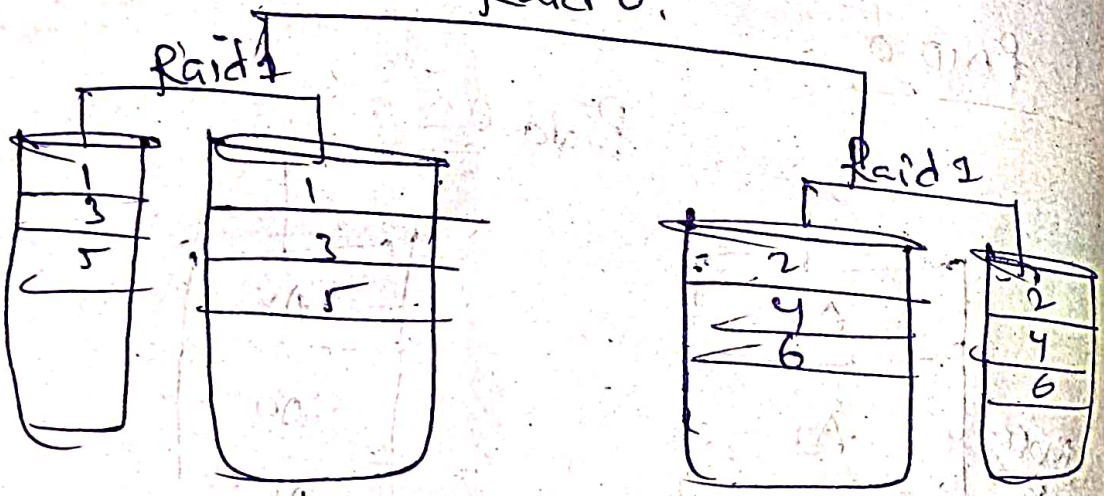
ita
- Rotation into

In RAID 0, we do data stripping, means data is going to divide into number of blocks. RAID 0 gives performance parallely we can Read and write the data.

Raid 1 :- It is called as **Mirroring** : This is the process of making a same copy of the different disk. It talks on the data availability.



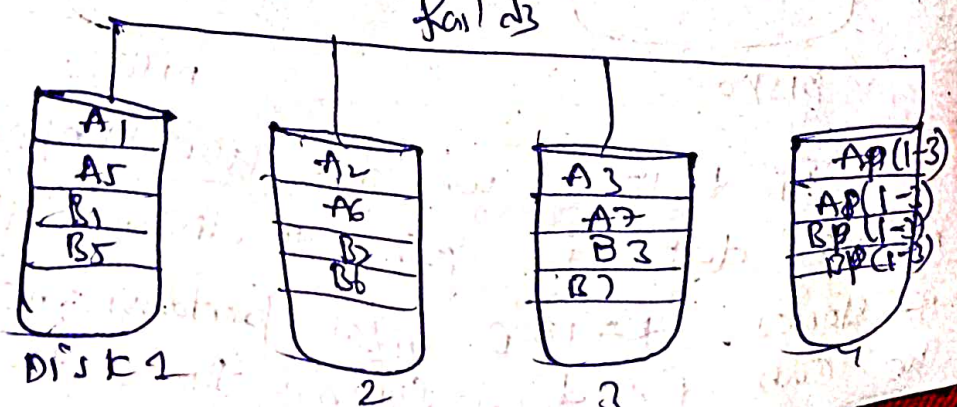
Raid 10 :-



Here Mirrored and stripped.

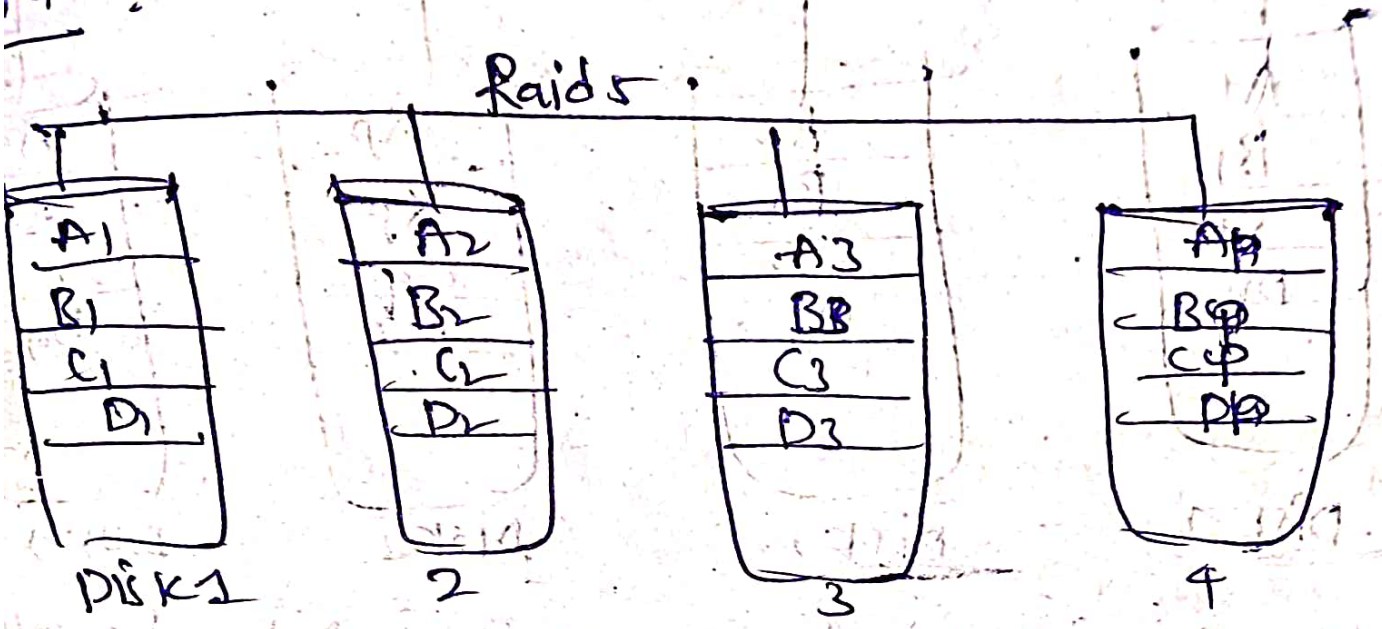
Eg :- Email server, web server.

Raid 3 :-



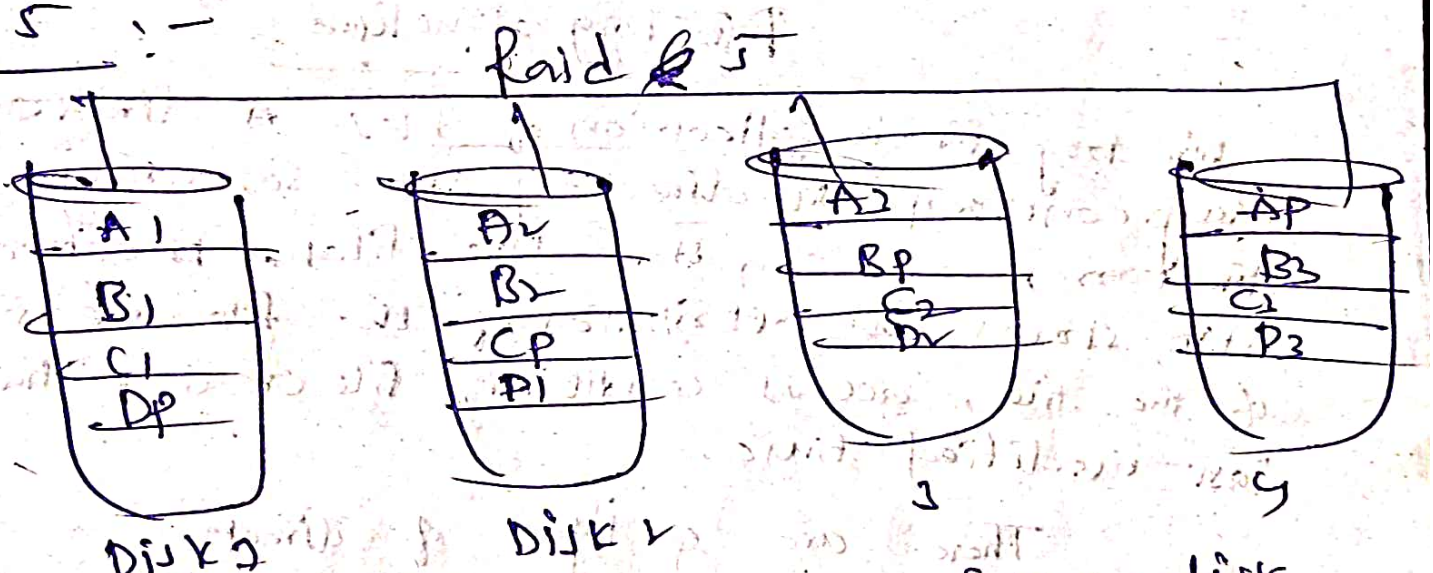
sit is used to store all the data one disk.

4 :-



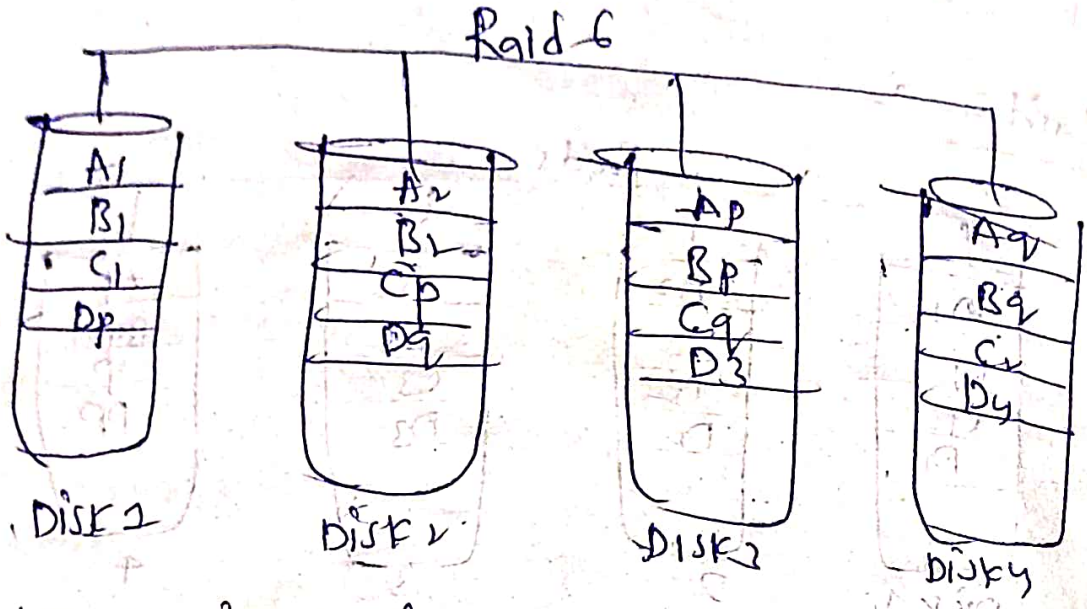
There is no difference in 3 and 4. They are solving problem by Raid 5

Raid 5 :-



all parity bits are not in same disk
The disks are available with same equality utilization.

Raid 6 :-



It consists of 2 parities bits because if one disk fails then one parity is sufficient if two fails then we need parity

Directory structure

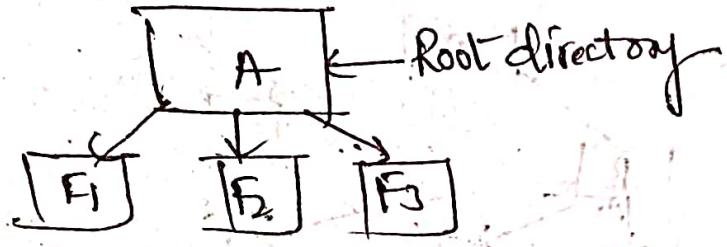
Directory is a collection of files. A directory may contain sub directory also. So, using directories we can store the files. Directories also stores the attributes of the file like size of the file, access condition, file created time, last-modified time.

There are 4 types of directory:

1. Single level
2. Two level
3. Tree structured
4. Acyclic graph directory

1. Single level directory :-

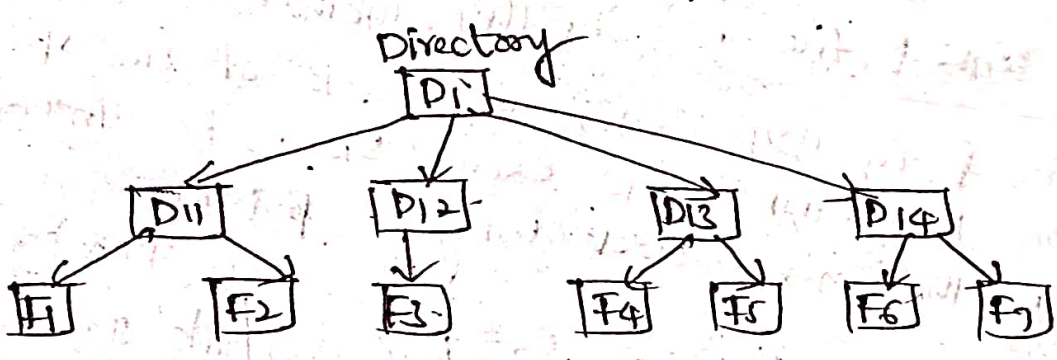
A single level directory contains of single node at ~~node~~ level 0. The directory contains of file



Simple to implement, Naming pblm, Grouping pblm

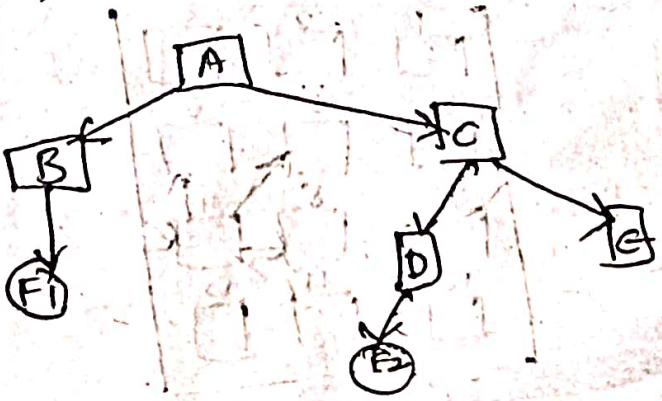
2. Two level directory :-

In the two-level directory structure each user has their own user files directory (UFD). The UFDs has similar structures,



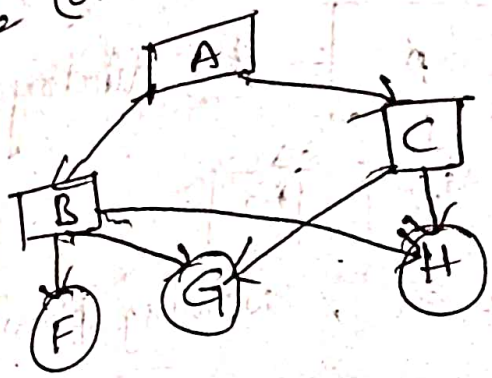
3. Tree structured directory :-

The directory is structured in the form of tree. It also has root directory, and every file in the system has a unique path. A directory within a tree-structured directory may contain files or subdirectories.



4. Acyclic Graph directory :-

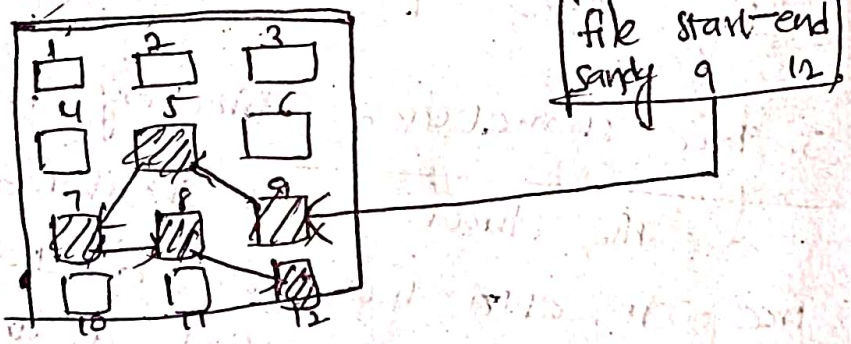
Two or more directory entry to a same file or sub directory



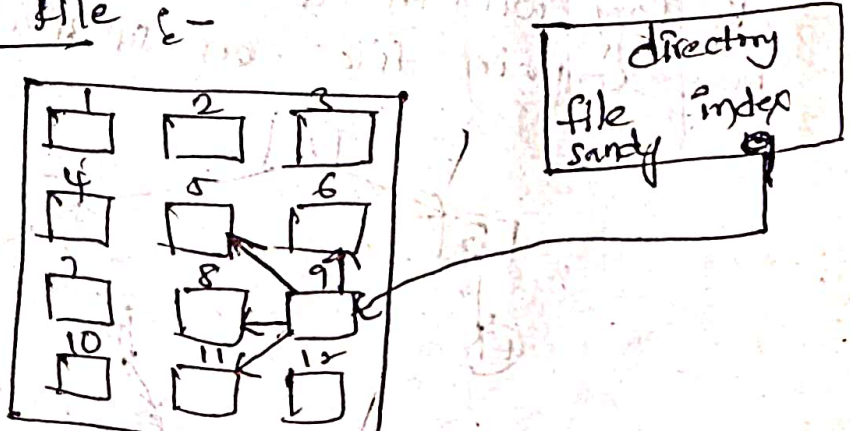
5) Allocation methods :-

1. Linked file
2. Indexed file

1. Linked file :- In this method, files are stored in non-contiguous block of free space on the disk, and each block is linked to the next block using a pointer.



2. Indexed File :-



Disk scheduling algorithm in Operating systems—

As we know, a process needs two type of time, CPU time and IO time. For IO, it request the OS to access the disk.

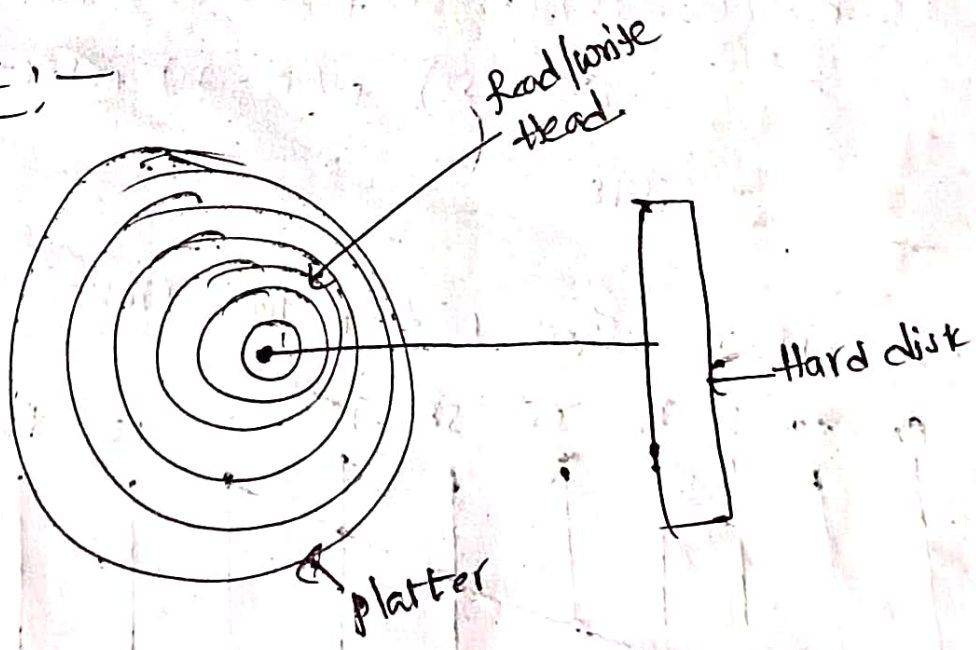
However, the OS must be fare enough to satisfy each request and at the same time, OS must contain the efficiency and speed of process execution.

The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Disk scheduling algorithm :—

FCFS, SSTF, SCAN, CSSCAN, LOOK, CLOOK

Disk :—



We assume a disk with 200 tracks and disk required queue has random request in it. The requested tracks in order 55, 58, 39, 18, 90, 160, 150, 38, 184 starting at track 100. Calcula

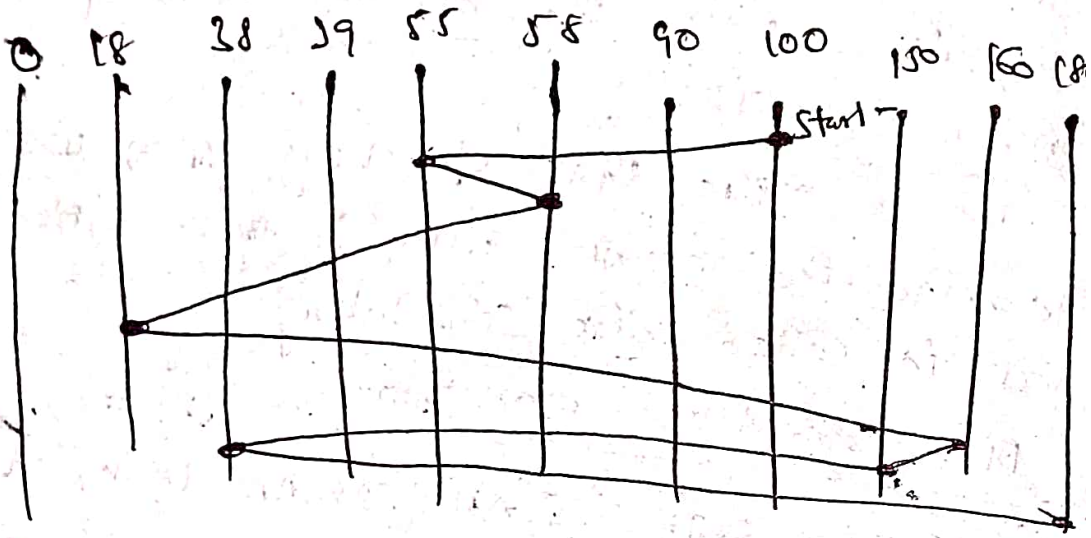
average seek length of fifo, SSTF, SCAN, C-S look and C-LOOK.

1. FCFS :- First come first serve.

Request (55, 58, 39, 18, 90, 160, 150, 38, 184)

Queue (0, 55, 58, 39, 18, 90, 160, 150, 38, 184)

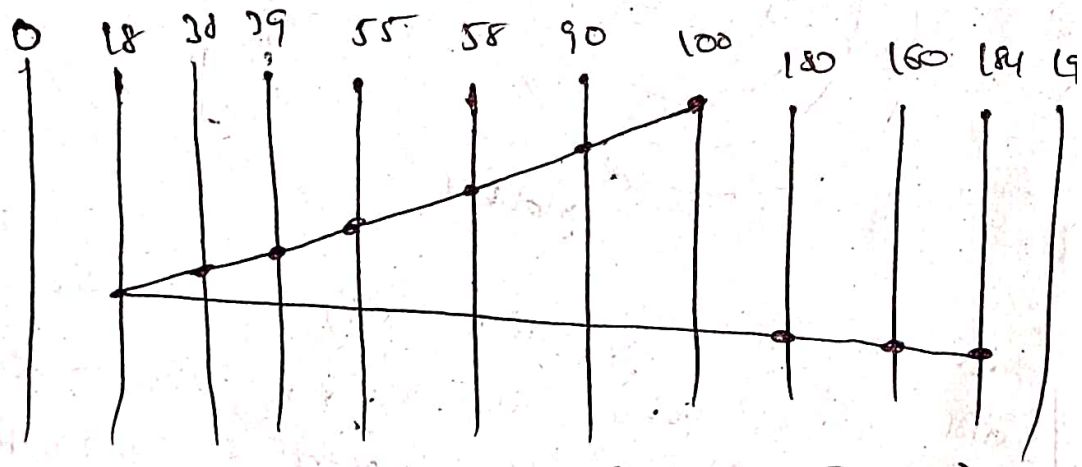
Head = 100



Seek = (100-55)+(58-55)+(58-18)+(160-18)+(160-150)+(150-38)+(184-38)

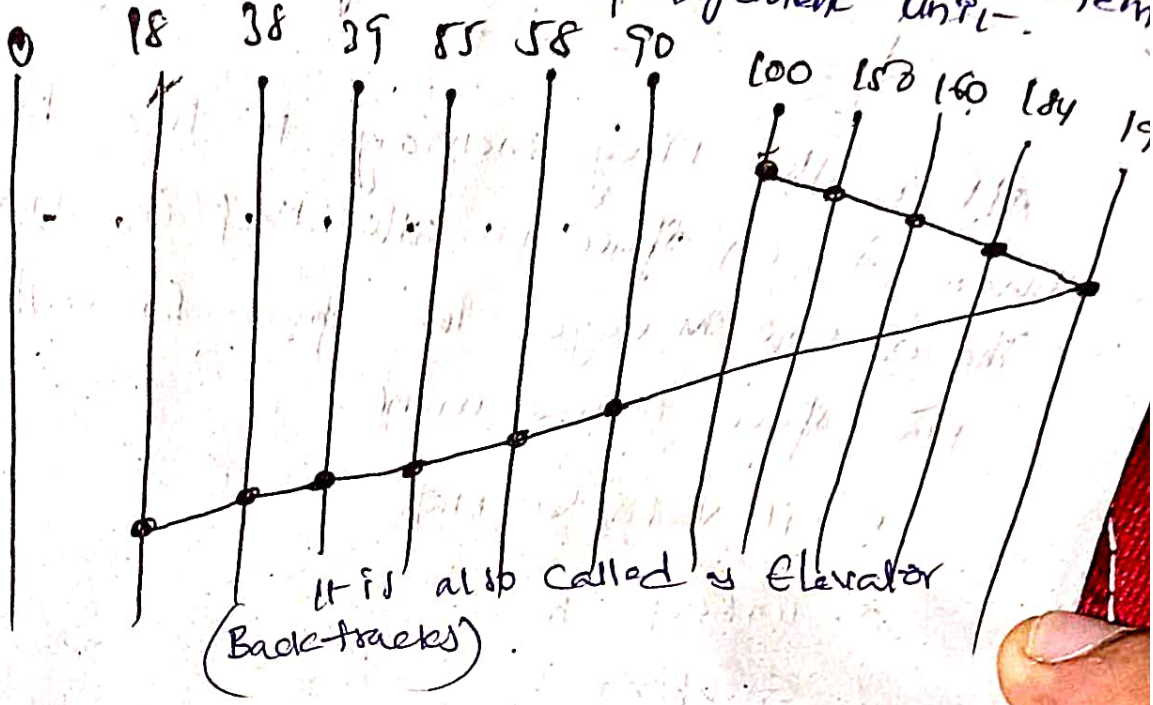
= 498

2. SSTF :- Shortest Seek Time First



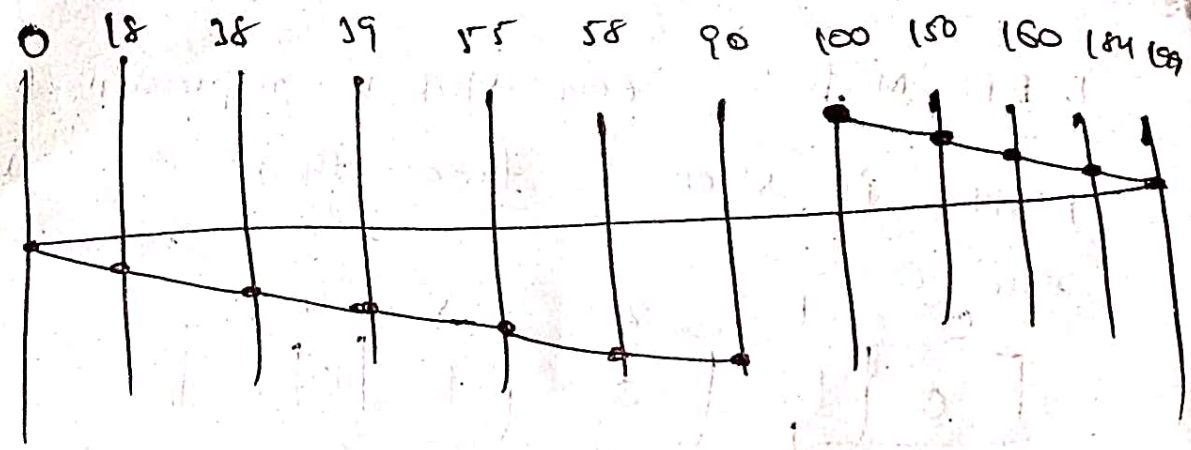
Seek = (100-90)+(90-58)+(58-55)+(55-39)+(39-38)+(38-18)+(18-0)+(184-160) = 248

3. Scan :- the request generated by the mem Management unit.

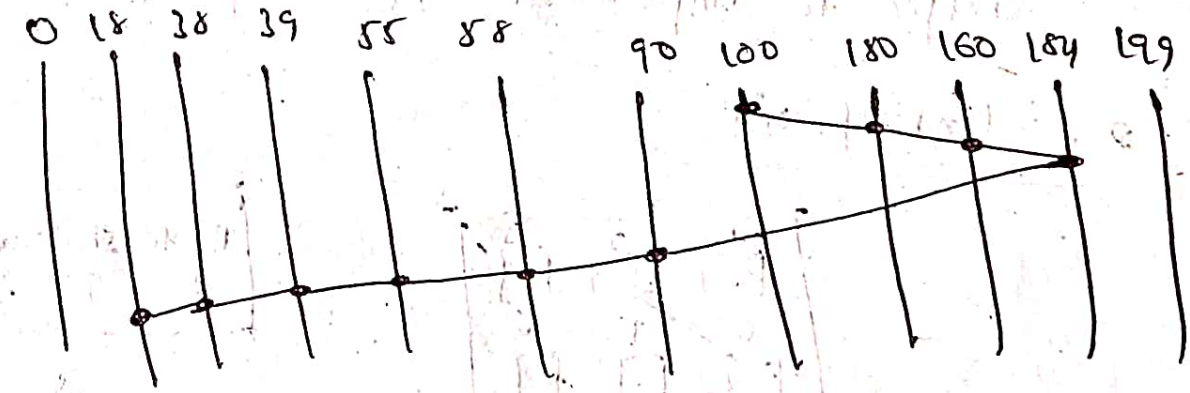


It is also called as Elevator (Back-tracks).

4. CScan :-



5. Look :-



6. Clook :-

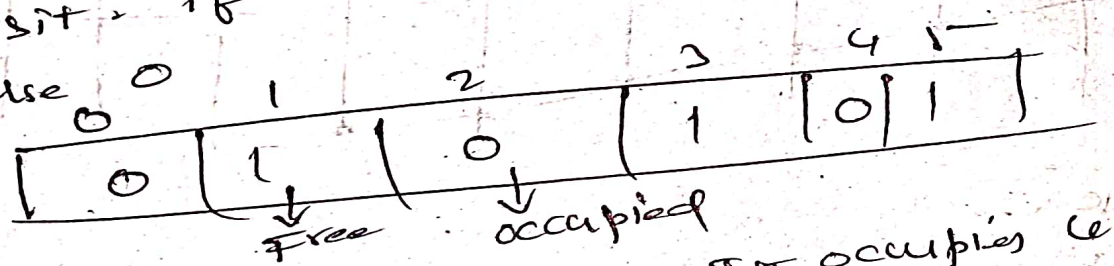
Free Space Management

After allocating memory to the files there is an space is allocated in memory. The OS how manages the space is called as Free space management.

1. Bit vector (or) map
2. Linked list
3. Grouping
4. counting

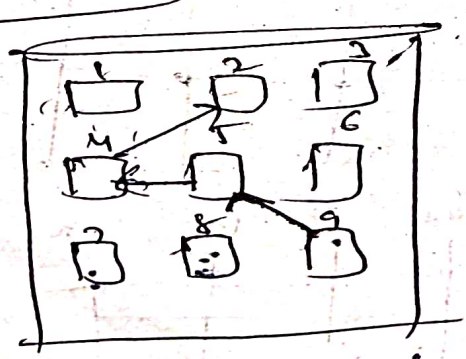
1. Bit Map :-

Each block is represented by 1 bit. if block is free then bit is 1 else 0



Simple to implement. It occupies less memory

2. Linked list :-



name	start	end
j	9	25

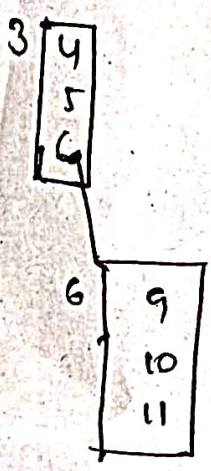
No wastage of space

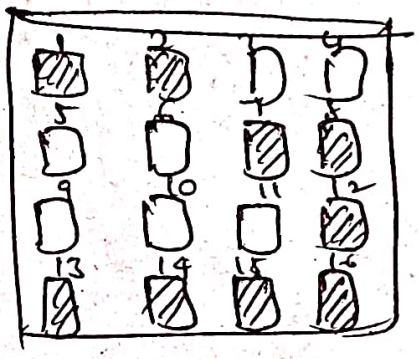
* For pointers need more memory
* Transferring more time

3. Grouping :-

Stored (~~stored~~) address of (n-1) free blocks in first free block

Adv :- address of large amount of blocks can be found quickly





4. Counting :-