* CONCURRENCY CONTROL

i) Process/thread Syncronization and Mutual Exclusion:

i) Process syncronization → Independent procen
→ cooperative process

Independent

i) Execution of 1 procen doesnot affects the execuh of other procen

Cooperative

i) Execution of 1 procen affects the execution of other procen

A procen that can affect or be affected by other Procenes executing in the Sys.

Ps problem arises in the case of cooperative process also because resource are shared in cooperative procen.

◉ Process syncronisation is the task of coordinating the execution of processes can in a way that no two processes access the same shared resources and data.

◉ Thread Syncronisation is defined as a mechanism which ensure that 2 or more concurrent Ps or threads don't simultaneo execute some particular program segment known as a critical section.

◉ Mutual Exclusion :-

If a process is executing in its critical section, then no other process is allowed to execute in the critical sectio

◉ Semaphore :- → signaling mechanism
→ Integer value

Binary
either
0
or
1

counting
have
any
value

Concurrency :-

~ - running

- Execution of multiple instruction sequences at the same time.
- It happens in OS, when there are several process threads running in ||
- ~ P T always communicate with each other through SM or MP.
- Concurrency results in sharing of resources results in problems like deadlock & resource starvation.

Principle of Concurrency :-

- Multicore P & || Processing, allows multiple processes and threads to be executed simultaneously.
- M.Ps & threads can access the same MS, D.V in code, or ever R or W to the same memory storage ~~to~~ Device var
- The amount of time it takes a process to execute cannot be simply estimated, and you cannot predict which process will complete 1st
- concurrent procen → Inter leaved / Overlapping

It is impossible to predict the relative speed of execution and following factors determine it:

1) The way OS handles interrupts.
2) The other processes activities.
3) The OS's scheduling policies.

Interleaved :- Divides memory into small chunks. It is used as ↑-level technique to solve memory issue for MB & chips.

Overlaping :- A procedure in which a CS works on several programs, suspending work on a program & moving to another when it encounters an instruction for i/p or o/p

* Problems in Concurrency :-

1) Sharing global resources. → Safely is difficult
   * Then the order in which various R & W op executed is critical. → If 2 P both make use of a global var & both perform R & W on that var *

2) Optimal allocation of resource :-
- It is difficult for the OS to manage the allocation of resource optimally.

3) Locating Programming errors :-
- It is very difficult to locate a programming error because reports are usually not reproducible.

4) Locking the channel :-
- It may be inefficient for the OS to simply lock the channel & prevent its use by other processes.

* Advantages of concurrency :-

1) Running of multiple applications
   → It enable to run multiple applications at the same
2) Better Resource Utilization :-
   → It enables that the resources that are unused by one application can be used for the other applicati
3) Better Average Response time :-
   → Without CC, each appl^n has to be run to completio before the next one can be run.
4) Better performance :
   → It enables the better performance by the OS. When 1 appl^n uses only the processors & another ap uses only the disk drive then the time to run both concurrently to completion will be shorter than the time to run each application consecutively

Drawback of Concurrency :-

- It is required to protect multiple appln from one another.
- -u- coordinate multiple appln through additional mechanism
- Addn performance overheads & complexities in OS are required for switching among appln
- Sometimes running too many applications concurrently leads to severly degraded performance.

Issues of Concurrency

1) Non-atomic : Opn that are n-a but interruptible by multiple cause Problem
2) Race-condition - A RC occurs of the outcome depends on which of several process gets to a 1st
3) Blocking - P can B waiting for resource.
4) Starvation ; It occurs when a P d'n obtain service to Progress
5) Deadlock ; -u- 2 Process are blocked & hence neither can proceed to execute.

Four conditions for correct mutual Exclusion.

1) No 2 processes simultaneously in critical section.
2) No assumptions made out about speeds or no.of CPU's.
3) No Process running outside its cs region may block another process running in the critical region.
4) No Process must wait forever to enter its critical region
   ↳ Waiting forever indicates deadlock condition

Requirements for Mutual Exclusion :-

relationship

1) Freedom from deadlocks : endless waiting due to circular wait
2) Freedom from starvation : unbounded w.d.t. order of service Policy
3) Fairness : R. are served in order they are made.
4) Fault tolerance ; Algo breaks if P die or messages are lost or garbled
5) Safety : only 1 Process / thread at a time inside cs
6) Progress ; if nobody has access and somebody wants in, somebody gets in.

Mutual Exclusion : OS support (S semaphore & M mutex)

- M & S both provide synchronization service but they are not same.

i) Mutex :-
- is a mutual exclusion object.
- that synchronizes accen to a resource
- It is created with a unique name at the soft
- Is a locking mechanism
- makes sure only 1 thread can acquire the M at a time & enten cs
- This thread only release the M when it exits the

```
wait (mutex);
cs
signal (mutex);
```

A binary semaphore as can be used as a Mutex but a Mutex can never be used a semaphore

ii) Semaphore :-
- Counting
- Binary

- Is a signaling mechanism
- A thread that is waiting on a semaphore be signaled by another thread
- uses 2 atomic operation → wait, signal
- wait -- of S
- If S is 0 or -ve no operation is performed

```
wait (s)
{
  while(s<=0);
    s--;
}
```

```
signal (s)
{
  s++;
}
```

Classical synchronization problems :-

i) Reader / Writers Problem.
ii) Producer and Consumer Problem.
iii) Inter - Process Communication (Pipes, shared memory).

i) Reader / Writers Problem.

→ The R/W Problem related to an object such as a file that is shared between multiple process.

→ The R/W problem is used to manage syncronization so that there are no problem with the object data.

- There are four types of cases could happen :-

case	Process 1	Process 2	Allowed / Not Allowed
1	Writing	Writing	NA
2	Writing	Reading	NA
3	Reading	Writing	NA
4	Reading	Reading	A

Reader Process.

```
wait (mutex);
rc++;
if (rc == 1)
wait (wrt);
signal (mutex);
Read db
wait (mutex);
rc--;
IF (rc == 0)
signal (wrt);
signal (mutex);
```

Writer Process.

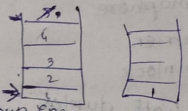
```
wait (wrt);
write db;
signal (wrt);
```

ii) Producer - Consumer Problem

- A classic syncronization problem in the OS.

```
mutex = 1
empty = 5
full = 0
buffer [5], max-items = 5, in=0 out=0
producer ()
{
  while (true)
  {
    down (empty)
    down (mutex)
    buffer [in] = item
    in = (in+1) % N
    up (mutex)
    up (full)
  }
}
```

down(full)
item = buffer[out]
out
up(empty)



Inter Process Communication (Pipes, Shared Memory)

- IPC is used for exchanging useful information between numerous threads in one or more processes (or programs)

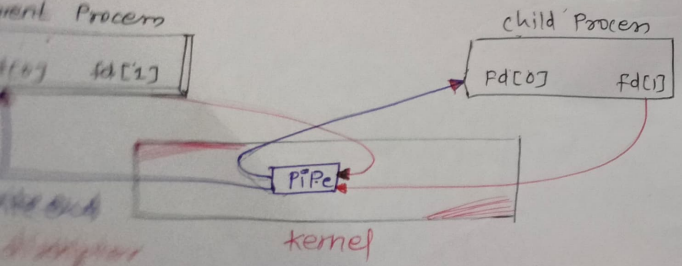
Role of synchronization in IPC

→ Typically, this is provided by IPC mechanism, but sometimes it can also be controlled by communication procen.

→ Methods to provide synchronization
  i) Mutual Exclusion
  ii) Semaphore
  iii) Barrier
  iv) Spinlock

Approches to IPC
1) FIFO
2) Message ques
3) Pipes
4) Indirect communication
5) direct communication
6) Message Passing
7) Shared Memory

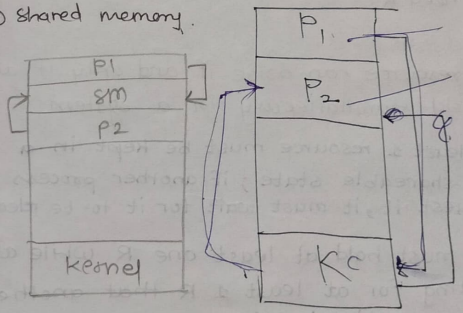
half duplex
Pipe: Behave like queue ds.
→ Type of data channel that is unidirectional in nature
Data in this type of data channel can be moved in only a single direction at a time.
Still, one can use 2-channel of this type, so that can able to send & receive data in 2 process.
Typically it uses the standard i/p & o/p
These pipes are used in all types of POSIX sys & diff ver. of os as well
Is a mechanism by which the o/p of 1 P is directed i/p of another P
de 1 flow of data between 2 related P



Why IPC.
→ Computational
→ It help to speedup modularity
→ Privilege Separation
→ convenience.
→ Help os to communicate with each other & synchron their action.

◉ FIFO & Full
→ Communication between two unrelated processes.
→ It is full-duplex method
→ Which means that the 1st procen can communicate wit the 2nd procen
→ And the opposite can also happen.

◉ Shared memory.



adlock

inciples of Deadlock

deadlock occurs when a procen or thread enters waiting state because a requested system resource held by another waiting procen, which in turn waiting for another resource held by another waiting ocen

n communications sys, deadlocks occurs mainly due to lost r currapt signals rather than resource contention.

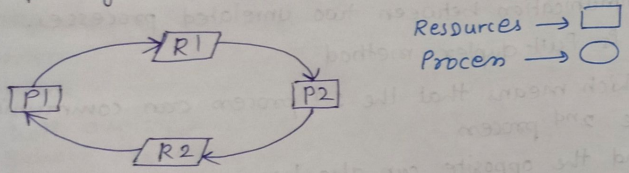


Fig: Both P need R

cessary conditions :-

deadlock situation on a resource can arise if and only if all following conditions hold simultaneously in a system

Mutual Exclusion :- At least 1 resource must be kept in a non-shareable state; if another process request it, it must wait for it to be release.

Hold and Wait :- A P must hold at least one R while also waiting for at least 1 R that another P is currently holding.

No preemption :- Once a process holds a R (i.e after its request is granted), that resource can't be taken away from that P until th P voluntarily release it.

Circular wait :- There must P P0, P1, P2..., PN such that every P[1] is waiting for P[(1+1) Percent (N+1)].

◉ Deadlock modeling

- As a collection of limited resources that can be divid into different categories and allocated to a variety of processes, each with different requirements.
- Memory, Printers, CPUs, Openfiles.
- The kernel keeps track of which resources are free and which are allocated, to which process they are allocated, and a queue of processes waiting for this resources to become available for all kernel managed resource.

Methods to Handling deadlock.

1) Deadlock Prevention
2) Deadlock Avoidance
3) Deadlock Detection
4) Deadlock Recovery

* Deadlock Ignorance

1) Deadlock Prevention :- we can prevent by avoiding o one of 4 necessary conditions.

2) Deadlock Avoidance :- To avoid deadlocks by avoid at least one of the aforement condition

Schedular → max no. of each R that P could potentially use in algo

3) Deadlock Detection :- IF it can't avoided, another is to detect them & recover in

Aside from the performance hit of constantly checking fo
→ A policy/algorithm for recovering from DL must be in p & when P must be aborted or have their resources pre there is the possibility of lost work.

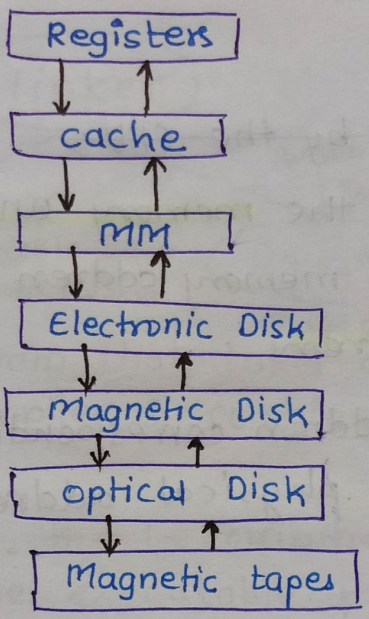
4) Recovery from Deadlock :-
1) Inform the sys operator & give him/her permission to inte
2) Stop 1 or more of the P involved in the dL
3) Prevent the use of resources.

Unit IV : MEMORY MANAGMENT

*) Memory Managment Requirement

◉ What is **main memory** ?

- A central to the operation of a mordern computer.
- MM is a large array of words or bytes, ranging in size from 100's to billions
- MM is a repository of rapidly available info shared by the CPU & I/O deviecs
- MM is the place where programs and info are kept when the processor is effectively utilizing them.
- RAM, Volatile M
  - ↓
  - lost data when Power interupts occurs.



◉ What is **Memory mgmt.**

- The task of subdividing the memory among different Process is called memory managment.
- Method in os to manage operations between mm & disk durin Process execution.
- Aim of M.mgmt is to achieve efficient utilization of memor

Why M.mgmt is required?

- Allocate and de-allocate memory before and after Process execution.
- To keep track of used memory space by procen.
- To minimize fragmentation issues.
- To proper utilization of mm
- To maintain data integrity while executing of procen.

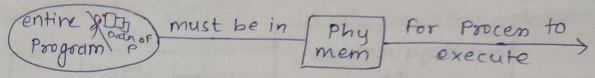
- A address generated by the CPU is known as a "Logical Add"
  - Also virtual add
  - Logical add. space can be defined as the size of the procen.
  - can be changed.

Physical Addren Space

- ~~An addren generated by the CPU~~
- An address seen by the memory unit
  [one loaded into the memory addren register of the memm]
- known as Real addren.
- Set of all physical addren corresponding to these logical address is known as physical addren space.
- Remains constant
- PA computed by mmu
- RT mapping from V to Phy add is done by a h/w device mmu.

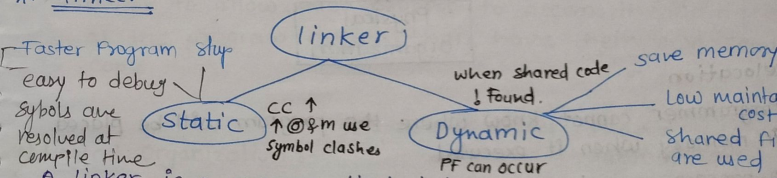
1) Static loading -
   - loading the entire program into a fixed address.
   - Requires more memory space.

- Dynamic loading
  - To gain proper memory utilization, DL is used



- Size of a process is limited to the size of physical memory
- Routine is not loaded until it is called.
- All Routine are residing on disk in a relocatable load form
- Unused Routine never loaded

* linker



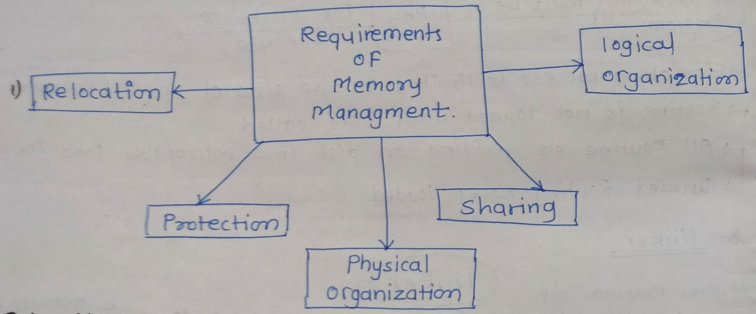
- A linker is a program that takes one or more object fi generated by a compiler & combines them into a single executable files.

- In static linking, the L combines all necessary prog modules into single executable program.
- so there is no runtime dependency.

- In Dynamic linking is ~~a~~ similar to dynamic loac
- stub small piece of code. → in → for each appropriate lib routin
- stub execute → checks → needed routine already in memory
  - if not → program loads the routine into memory.

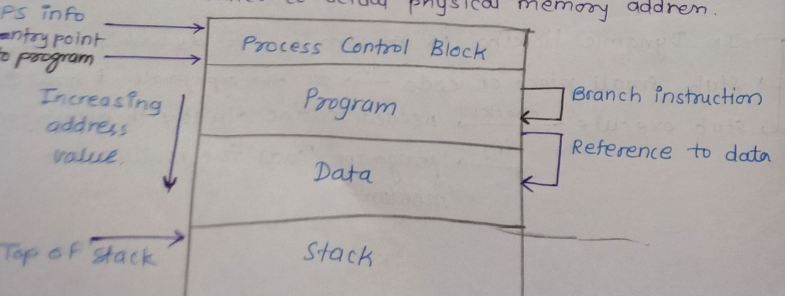
* Memory Mgmt Requirement :-

- MMgmt keeps track of the status of each memory location.
- It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed
- Requirements



1) Relocation

- Programmer cannot know where the program will be placed in memory when it executed.
- A process may be often relocated in main memory due to swapping.
- Swapping enables the OS to have a larger pool of ready-to-execute processes.
- Memory reference in code (for both instructions and data) must be translated to actual physical memory address.



2) Protection :- Prevent P from interfeering with O.S. or other P.

- P should be not able to reference memory locations in another P without permission.
- Impossible to check addresses at compile time in programs since the program could be relocated.
- Address reference must be checked at run time by h/w
- Often Integrated with relocation.

3) Sharing :- Allow P to share data/Program

- Must allow several processes to access a common portion of main memory without comporomising protection.
  - cooperating P's may need to share access to the same
  - Better to allow each process to accen the same cop of the program rather than have their own separat copy.

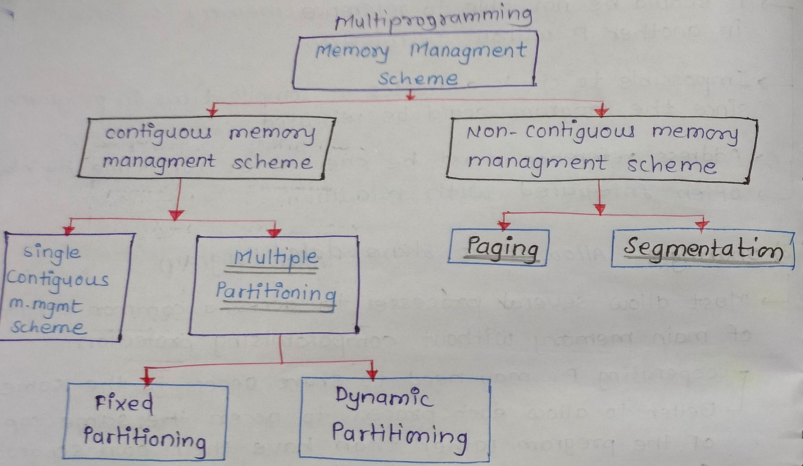
4) Logical Organization :- Support modules, shared subroutines.

- Users w programs in modules with diff. characteristics.
  - Instruction modules are execute-only.
  - Data modules are either R-only or R/W
  - Some modules are Private others are public.
- To effectively deal with user programs, the OS & h/w shou should support a basic form of module to provide the required protection & sharing.

5) Physical Organisation :- Manage mem ⟷ Disk transfer

- SM is the long term store for programs and data while MM holds program & data currently in use.
- Moving info between these 2 levels of memory is a major concern of memory managment.
  - It is ↑ inefficient to leave this responsibility to the AP.

* Memory Partitioning :-



⇒ Ways to organizing prog in mem.

* Contiguous memory managment scheme :-
  - Contiguous memory allocation means assigning continuous blocks of memory to the procesn.
  - low overhead, ip to find large enough block
* Non-Contiguous memory managment scheme :-
  - easy to find hole in which seg it fit chunks/called segment
  - The program is divided into blocks (Fixed size or variable) and loaded at different portions of memory.
    each seg can be placed in diff part of mem

Multiple Partitioning :-

mm is divided in a no.of fixed-sized partitions where each partition should contain only 1 process.

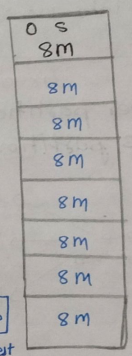
Available memory is 'hole'

i) Fixed Partitioning :- OS maintain a table that indicates which part of memory are available & wa occupied by Process.

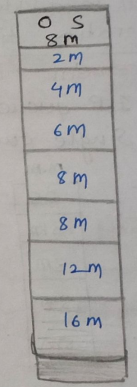
- Partition main memory into a set of non overlapping regions called partitions. pros:- simple to implement little os overhead
- Partitions can be of equal or unequal sizes.
- Any process whose size is less < Partition size or = Ps can be loaded into the partition
- If all partitions are occupied, the os can swap a p out of a Par
- A program may be too large to fit in a partition. Thes programmer must then design the program with overlays.

(P) arrives need mem → Search hole → large enough to store P → allocate a memory to P

otherwise keep rest available to future request
→ WASM Dy.storage allo. problem occur.
which concerns how to satisfy a request of size n from a list of free holes.



Equal size Partition

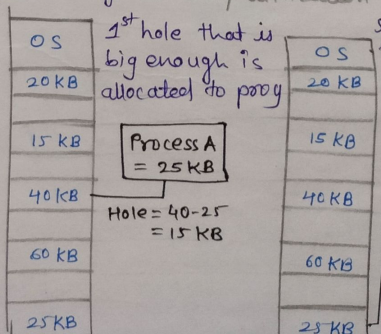


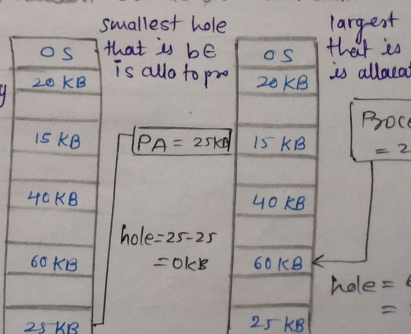
unequal size

cons:- Inefficient Partition.
Max no.of active proce

ii) Dynamic Partitioning
- Partitions are of variable length & number
- Each P is allocated exactly as much memory as it require

Placement Algo:- memory utilization should be maximum





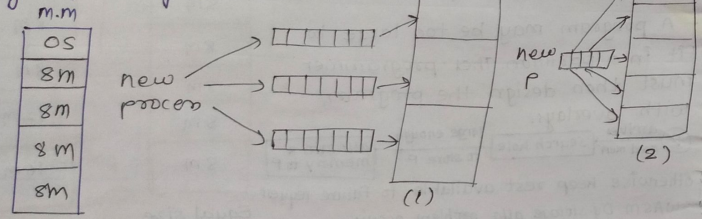
Dynamic Partitioning :-

- Partitions are created dynamically
- Each process is loaded into a partition of exactly the same size of that process.
- m.m is divided into no of static partitions of sys Generation time

Placement Algo for fixed partition.

1) 1 P queue per partition
2) Single queue partition.

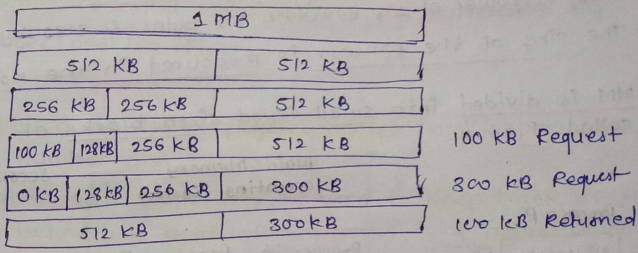
ex:-



Buddy system :-

256
256
512

- Is a simple dynamic storage allocation method
- BS is a mem allocatn & mgmt algo that manages mem in power of 2 inc
- Linux OS manages mem using buddy Algo



- Single allocation block to be split to form two blocks half the size of the parent block.
- These two blocks are known as 'buddies'
- An algo in which larger memory block is divided into small parts to satisfy the request
- An algo used to give best fit.

* Paging :-

→ Paging is a memory managment schema that eliminates the need of contiguous allocation of physical memory.

→ Paging plays important role in implementing virtual memory.

→ The PAS is conceptually divided into several fixed-size blocks, frames

→ Paging is a memory managment technique in which process address space is broken into blocks of same size called page (size is power of 2, between 512 bytes & 8192 bytes).

→ The size of the process is measured in the no. of pages.

→ MM is divided into small fixed-sized blocks of memory called Frames.

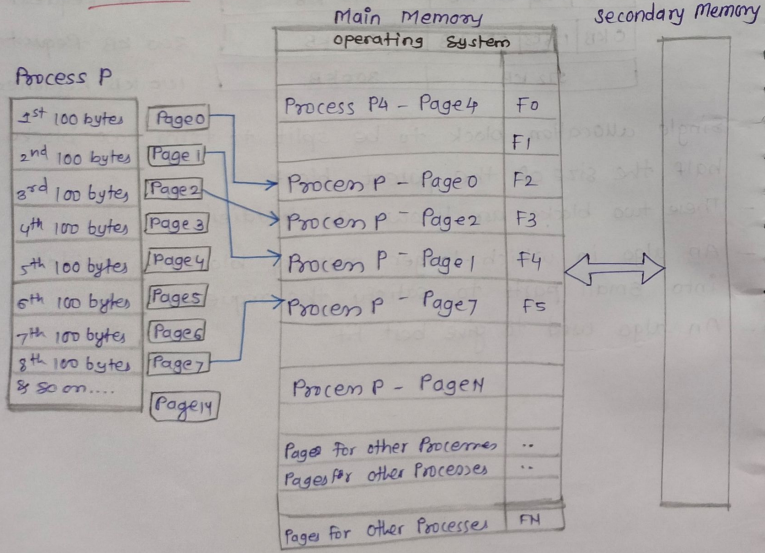
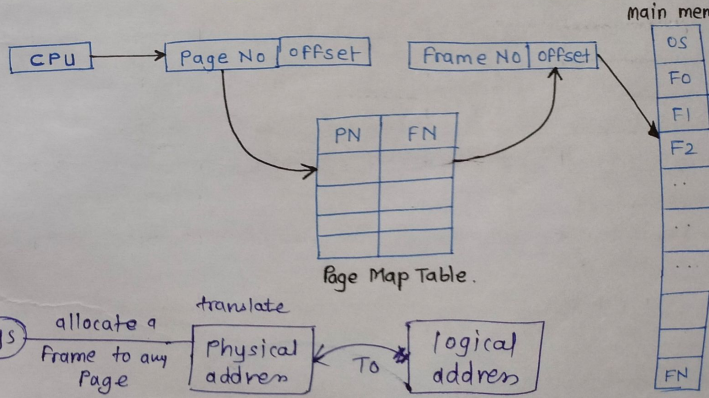


Fig : Paging.

- Logical Address or Virtual Address : An generated by the CPU
- LA space or VA space : The set of all LA generated by a program
- Physical Address or : An address actually available on a memory un (represented in bits)
- Physical Address space : The set of all physical addresses (words / bytes) corresponding to the logical addresses
- Page address is called logical address and represented by Page number and the offset.
- Frame address is called physical address and represented a Frame number and the offset.
- A DS called page map table is used to keep track of th relation between a page of a process to a frame in phy



& create entry ~~to~~ into the page table to be used throu execution of the program.

Advantages of paging

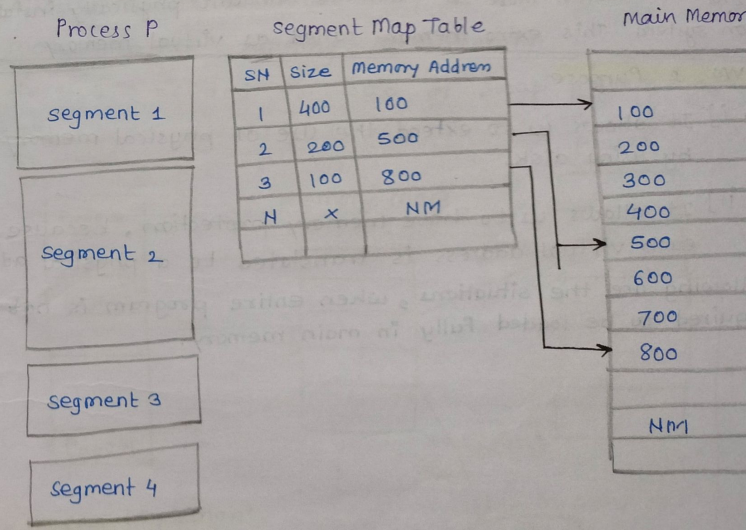
- Reduce external Fragmentn
- Simple to implement
- efficient M.mgmt tech
- Swapping becomes very easy

Disadvantages

- Suffer from internal fragmentation
- Requires extra memory space
- need high specified sys.

* segmentation :-

- It is memory mgmt technique.
- Divided into several segments of different sizes
- One for each module that contains pieces that perform related functions.
- Each segment is actually a different LAS of the prog
- When a P is to be executed
- Its corresponding segmentation are loaded into NCM through every segment is loaded into cm block of available memory.



* Virtual Memory :-

- A common techinque in computers operating system.
- Storage allocation scheme.
- 2ndary memory can be addressed as though it were part of the main memory.
- The size of virtual storage is limited.
- It is a technique that is implemented using both h/w & s/w
- It maps memory addresses used by a program, called V.add. into physical add's in computer.

- can addrem more than the amount physically installed on system. This extra memory called as virtual memory.

VM : Purpose

i] It allows us to extend the use of physical memory by using disk.

ii] It allows us to have memory protection, because each virtual address is translated to a physical add.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- somtg which appear to be present but actually it is not.

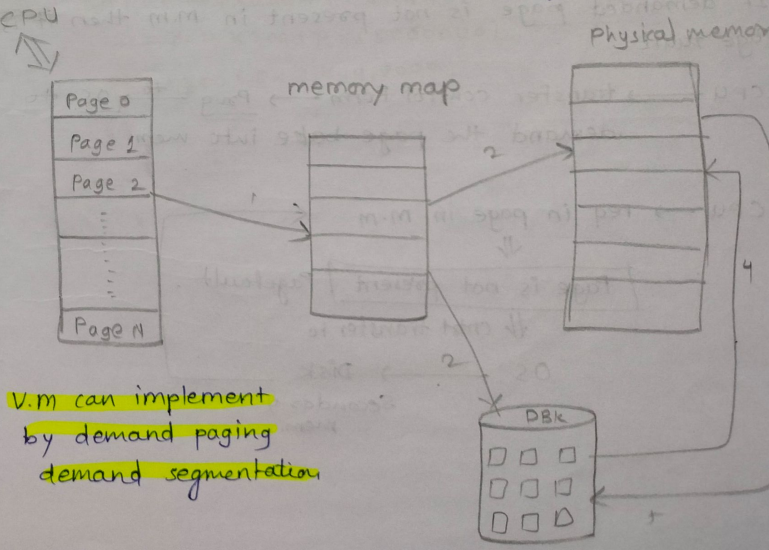
16 GB ⟺ 4 GB

Users use V.m — Real mem of comp.

⇒ It just gives illusion to us

* A programmer can w a prog which requires more memory than the capacity of M.m such a prog is executed by virtual mem tech

- Only part of program needs to be in memory of exec
- logical add space can there for be much larger than phy A.S.
- Allows A.S to be shared by several P
- more Program running concurrently
- less I/O needed to load or swap procen.



V.m can implement
by demand paging
demand segmentation

Performance of Demand Paging.

Pagefault rate $0 \leq P \leq 1.0$

if $P = 0$ no page fault.

if $P = 1$ every reference is a fault.

Effective Access time

EAT = (1-P) × Memory Access + P (Page Fault overhead + swap page out + swap page in + restart overhead)

Eg:- MAT = 200 ns

APF = 8 ms

Average Page fault Service time

$$EAT = (1-P) \times 200 + P(8ms)$$

$$= P - P \times 200 + P \times (8000000)$$

$$EAT = 200 + P \times 7,999,800$$

* Page fault

The demanded page is not present in M.M then its PF.

* Page fault can be handled by OS

1) OS looks at PCB to decide :
   - invalid reference ⇒ abort
   - just not in memory ⇒ (load the page)
2) Get empty Frame.
3) Swap page into frame via disk operation
4) Reset page table to indicate to page.
   now in memory
   set validation bit = V
5) Restart the instruct^n that cause the PF

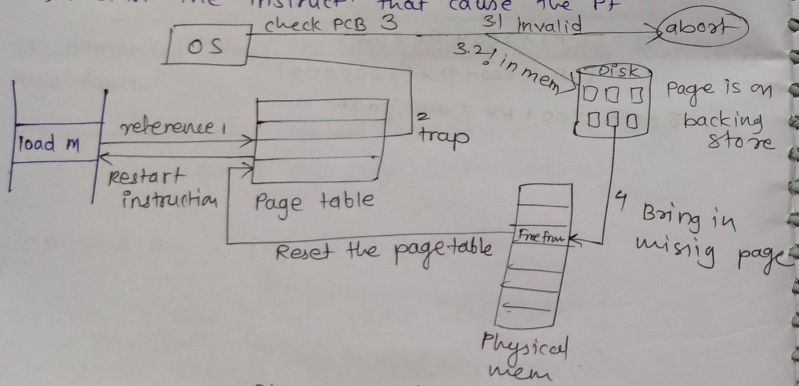


Fig: PF handling

what happens if there is no free frame ?

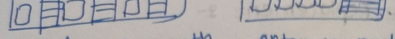
1) Page Replacement ; find some page in memory, but not really in use, swap it out

Same page may be brought into memory several times.

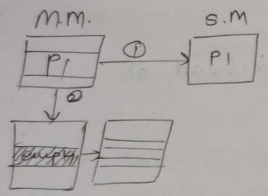
⊙ Page Replacement :-

- To prevent over-allocation of memory by modifying page-fault service routine to include page replacement.
- Use modify (dirty) bit to reduce overhead of Page transfer.

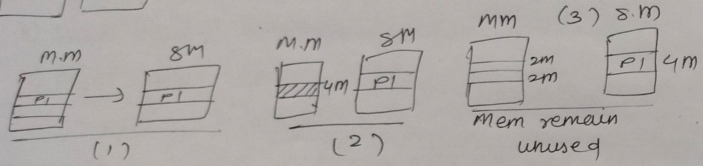
* Fragmentation



A process are loaded & removed from memory the free memory space is broken into little pieces.



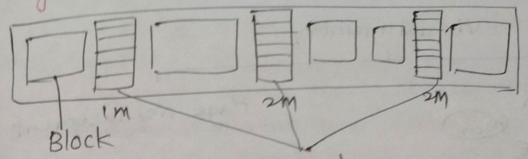
After some time that procen can't be allocated to mem cause small size mem
0 of mem block remains unused this Prob is called fragmentation



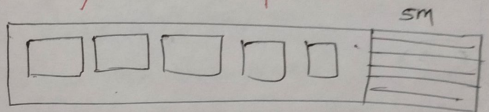
Fragmentation

- External fragmentation
- Internal fragmentation

Fragmented memory before compaction



Memory After compaction.

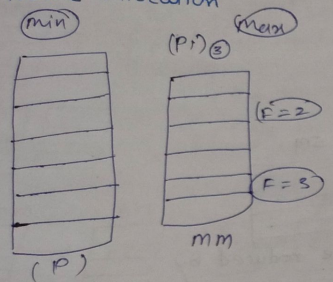


- external fragmentation can be reduced by compaction of shuffle free memory together is one large block.
- Internal fragmentation can be reduced by effectively assiging the smallest partition but large enough for the procen.

Allocation of Frames.

Virtual memory

Frame Allocation — Page Replacement



Frame Allocation

Equal — Weighted — Dynamic

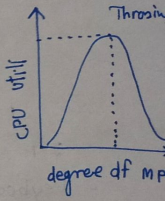
Equal: 60 Frames → 60/3 = 20

Weighted:
$P_1 = 30$
$P_2 = 30$
$P_3 = \frac{40}{100}$
F → 10 Frames
$P_1 = \left(\frac{30}{100}\right) \times 10 = 3$ f
$P_2 = \left(\frac{30}{100}\right) \times 10 = 3$ f
$P_3 = \left(\frac{40}{100}\right) \times 10 = 4$ f

Dynamic: Based on Priority
↑ P 1st ex.

Thrashing :-

- Swapping out a piece of a process just before that piece is needed
- The processor spends most of its time swapping pieces rather than executing user instruction
- If a process does not have enough pages pagefault rate ↑
  - This leads to
    - ↓ CPU utilization
    - OS think that is need to ↑ the ° of mp
    - Anothe p added to the system
- A process is busy swapping pages in & out
- This ↑ paging activity is called Thrashing
- mp ↑ paging ↓ CPU utilizatn

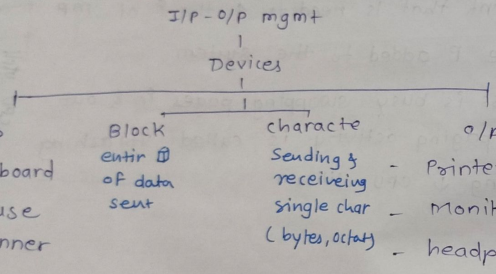


UNIT V

* I/P | O/P & File mgmt

I/O Mgmt & Disk scheduling :-

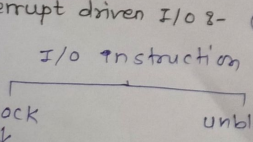
- IO Mgmt controls the data exchange between external devieces and mm.
- IO sys transfer info between comp mm & the outside world.



- Organisatn of I/O Functions :-

1) Programmed I/O :- under direct control of CPU
   ② CPU issue a command on behalf of a P to an I/O module.
   CPU then waits for the operatn to be completed before proceeding.

2) Interrupt driven I/O :- ②, if I/O instructn is nonblocking, CPU contines to execute → instru From the same procesn.



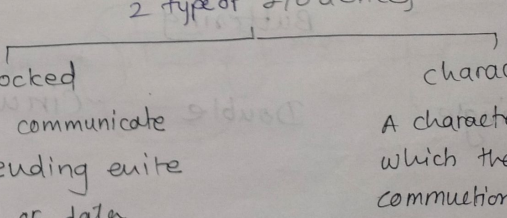
OS takes over, suspend current P, Assign CPU to another P

CPU continues to execute next instructn from same P.

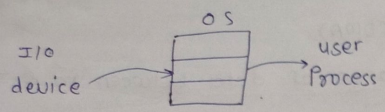
I/O Buffering :-

3) Direct memory accen (DMA)
   Module controls exchange of data between m & I/O module
   - CPU → send request → transfer block → Block of data → to → DMA module → is interrupt only after entire block has been transferred
   - DMA module take over control of sys bus to perform the transfer
   - CPU initiates the I/O by sending the following informatio to DMA module.
     + Request type - read or write ; using the read or write control line between the CPU and DMA modu
     - Address of the I/O devices ; on the data line
     - starting location in memory for read / write; communicat on data lines and stored by DMA module in its address register.
     - No. of words to read / write ; communicated on data lin and stored in the data count register

Enlist the character of block & character device



I/O Buffering [match speed gap of I/O devices & CPU]



- A user p reads block of a data from a tape, one at a time, with each block having a length of 100 bytes.
- data should be read into user space area
- One simplest way would be to execute an I/O command to the tape unit and then wait for the data to become available

- To overcome inefficiency & inconvinence
- improves ~~I/O~~ throughput of I/O operat^n
- In Buffering, OS stores its owen copy data in memory while transs → to or from dev

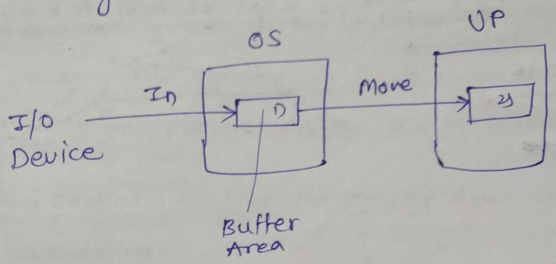
to cope with device speed mismatch
transfer size mismatch
to maintain copy sematics

Buffering

Single Double Circuler

Single Buffering.

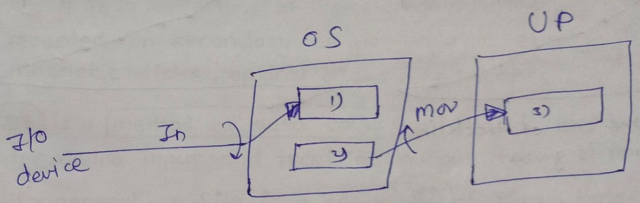
Block oriented (T+C)
string oriented
└ line at a time fashion
Byte at a



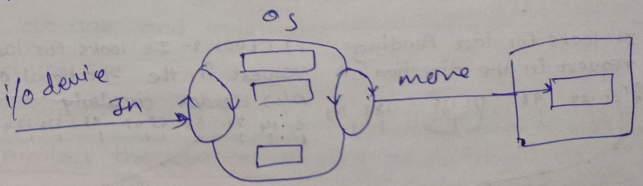
Double Buffering.

BO
SO

CC=T , C
for line
for byte

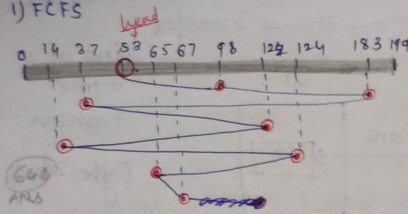


Circular Buffer



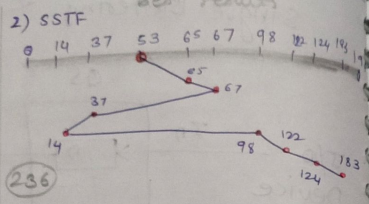
Disk Scheduling. 98, 183, 37, 14, 124, 65, 67, 122

1) FCFS



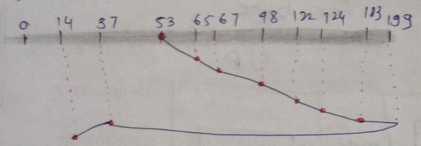
640
Ans

2) SSTF



236

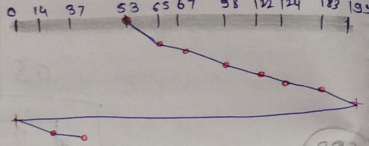
3) SCAN (ELEVATOR) (High Order)



Scan goes to the end of track in which direction is given.

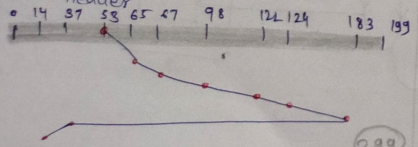
33

4) CSCAN :- Goes end of both track



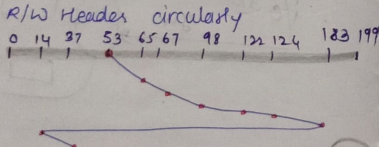
382

5) LOOK - It looks for last Pending request in the direction of R-W- header



299

6) CLOOK :- It looks for last pending request in the Directional of R/W Header circularly



322

File Managment

Managing the file system to enable users to keep their data safely and correctly is an important function of OS.

- Managing file systems by OS is called File mgmt
- File mgmt provides tools for these file related activities
  - → Creating new files for storing data
  - → Updating
  - → securing data through passwords & encryption.
  - → Recovery in case of sys failuer.
- A File is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes.
- It is a method of data collection that is used as a mediu for giving input and receiving output from that program.
- In general, a file is a sequence of bits, bytes, or record whose meaning is defined by the file creator & user.
- Every file has a logical location where they are located for storage and retrival.

• Objectives

- I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data.
- Helps OS to standardized I/O interface routine for user processes.
- It provides I/O support for multiple user in a multiuser systems envirnoment

F - Files
- storage
- Disk

Properties

F are on ⊙ or other ▭ & do not disappear when user log off

f have names & are associated with access Permission that Permits controlled sharing

F could be always ▯▯▯ or more complex structures to reflect the relationship between them

File structure

A FS needs to be predefined format in such a way that an OS understands.

It has an exclusively defined structure, which is based on its types.

F → C → by owner → Per
P ← access

Types of FS
- executable file
- Text file — It is a series of characters that is organised in line
- Object file — series of bytes that is organized into blocks
- Source file — Series of functions & Processes.

File Attributes
- Name — string used to refer file
- type — Type - sys suppor diff T need to know TOF
- Identifier — refer other attribute of f — Identifier - uniquely identify file by os (Not Human R in no. form)
- Size — size of file in bytes
- Location — exact loc of f
- Protection — Diff T of Permisn for R/W & exe (associate with file)
- Time, date, Security

Time, - date, Security

File Type :-
- Character Special File :- h/w File tha R/W data character by character ex, Mouse, KB, Printer
- Ordinary Files :- store user info; may be T, exe programs, DB; Allow user to perform op t, del & modi
- Directory Files :- contains Files & other info about; A folder to hold & organize multipl files.
- Special files :- Device files; Represents physical devices ex; printer, disks, n/ws, flash, drive et

Functions of File operations
- Reposition move R/W Position
- Create File
- Find space on disk
- Make an entry in the directory
- Delete directory entry, regain disk space
- Write to File, requires positioning within the file — write()
- Read from File involves positioning within the file. — read()

Common terms associated with the Files.

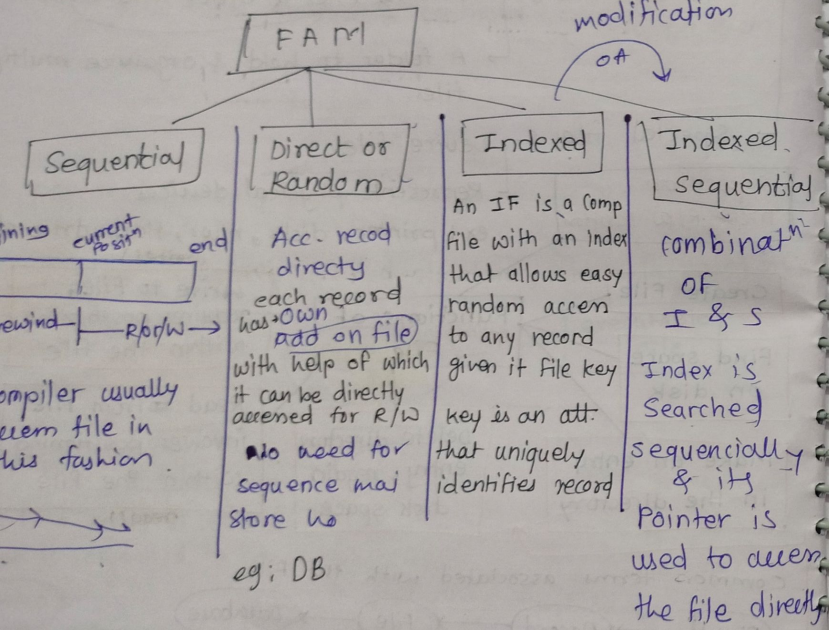
Field → Record → File → Database

This operations are available in terms of system call & can be used in user program for handaling the file relate

operation/function of file
1) Create File
2) Write in File
3) Read from File
4) file delition
5) Truncation in File - erase content
6) File seek operation - to set the Pointer at specified location for R or W

File Access Methods

- Process that determines the way that files are accessed and read into memory.
- Generally, a single access method is always supported by operating systems.
- Some OS which also supports multiple accen method.



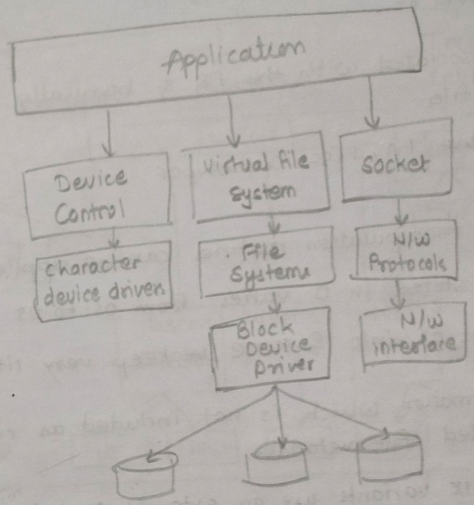
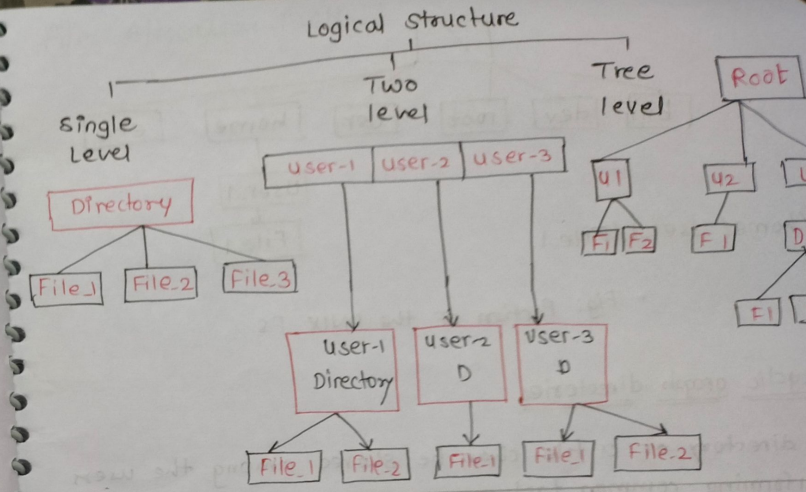


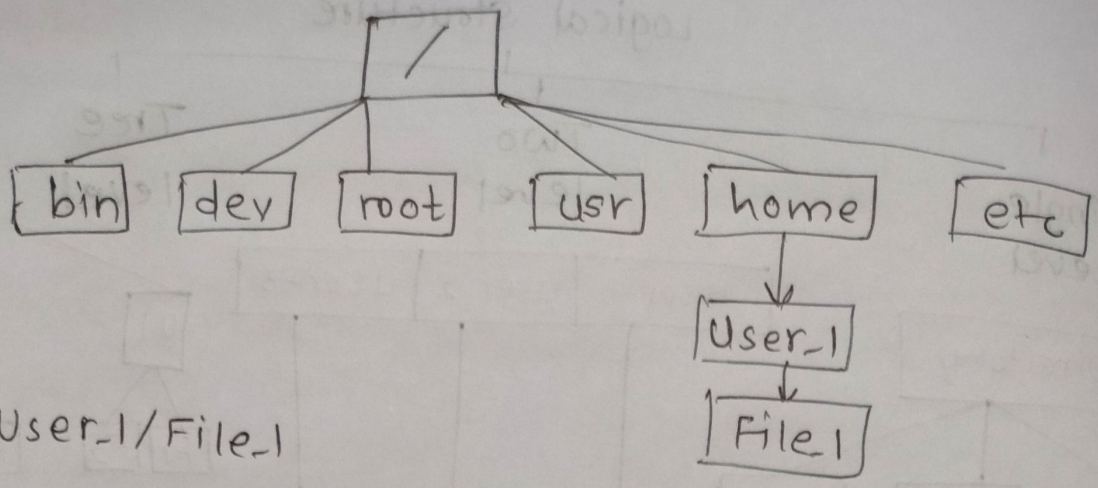
Fig: Linux File & I/O system.

File Directories :-

- Directory is associated with the FS & basically collection of files.
  - ↳ info including FA, FLOC, OwnershipOF.
  - ↳ special kind
  - ↳ various file manipulation routines can be applied.
  - ↳ other info stored in D varies from OS to OS
  - ↳ some keep lot of info & some we keep very little info.
- These other information which is not included as content of the file is called file metadata.
- System like UNIX variants use an external table called as inode in which such info are stored.

Basic Information	- File name - Type of the file - File Organization
File access related information	Various categaries of users are able to accen a file with specific permission.
Location of a file in secondary memory related info	Volume info indicates the device in which file actually resides. Starting LOC of file in the disk
Other info of the file	Owener of file Date of file created Last accen date of file.



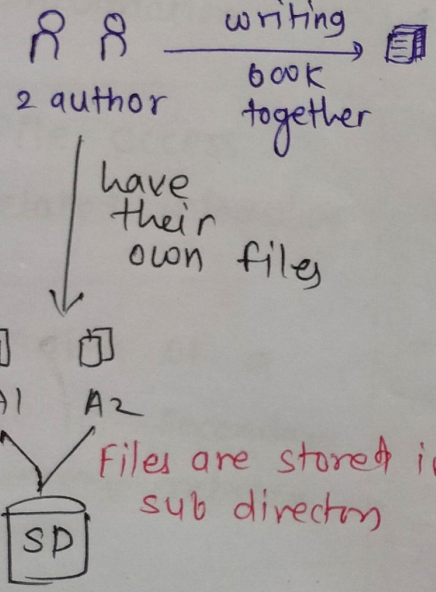


/home/ User_1/File_1

Fig: Portion of the UNIX FS.

Acyclic graph directories

- A directory or a file chan be shared among the users performing common task.
- A SD or files therefore can exist in two or more places at the same time
- The need of SD or File can be described by an ex.



equal Responsible to
W book so need
file in their directory
so it should be shared
but hierachical tree structure

★ An Acyclic graph directories & allow sharing of directories & files.

- Same file or subdirectory may be in 2 diff directory

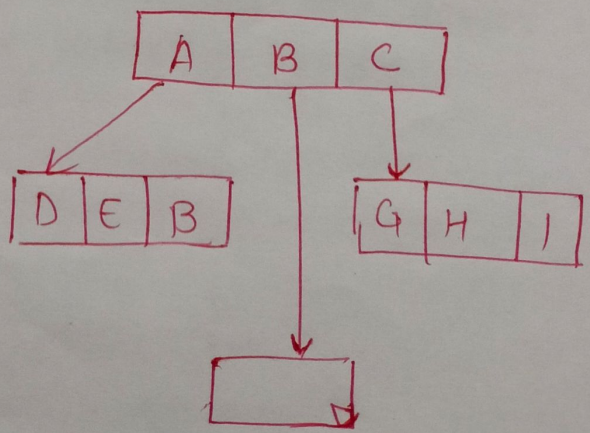


Fig: Acyclic graph directory

Assembler

collecth of mnemonic is Assembly lang

Assembler is a translator that converts assembly lang into machine lang along with some infor for the loader

syntax

[lable:] mnemonic [opreand] [;]

Add R1, R2;

Assembly level Prog terms

LC - location counter

Literals - constant

Symbols - var - holder

Procedure - function like c

Adv

- easy
- Binary no use mnemonic
- AP ↑ procesin

dis 3-

- Need translator