Unit – 5

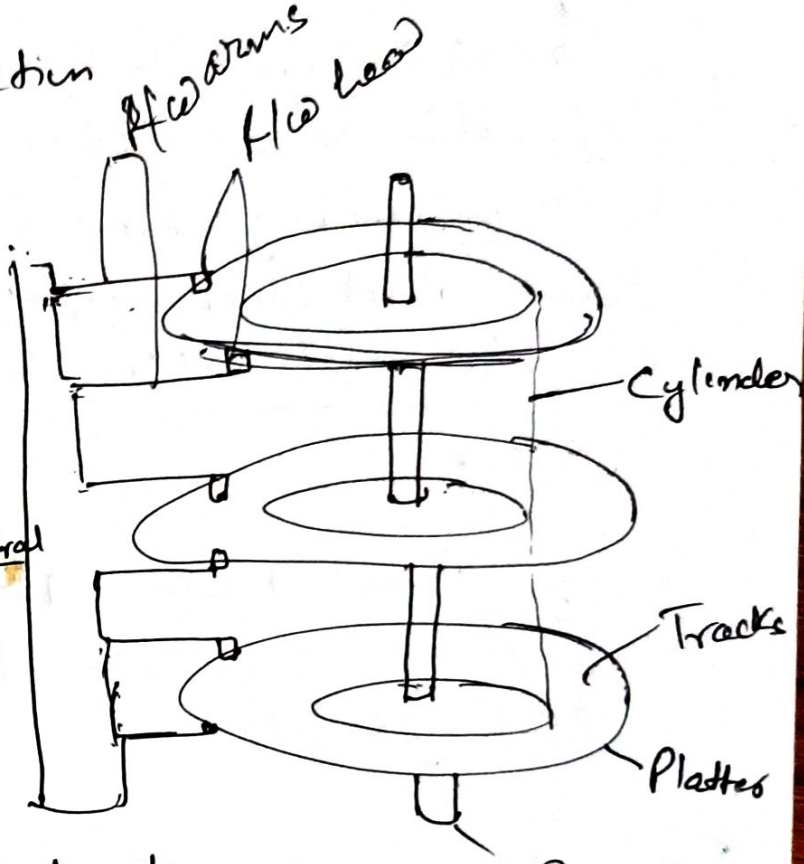
Disk Management

A disk o read/write can result following three outcomes –

* Successful Completion
* total failure
* Partial failure. (Some of the disk plate failing on Bad Sector)

Terminologies

* Sector
* Track — Concentric or spiral circles
* Cylinder — Tracks with same radius
* Allocation Unit — Portion of a track enclosed within a unit or Amount of
* Capacity –
* Density — No. of bytes per track.
* Read – write head assembly
* Platter
* Spindle rod
* Transfer time (Seek time and Latency time)
* Disk Partition (Logical and Physical) same disk in multiple drives
* Bad block management

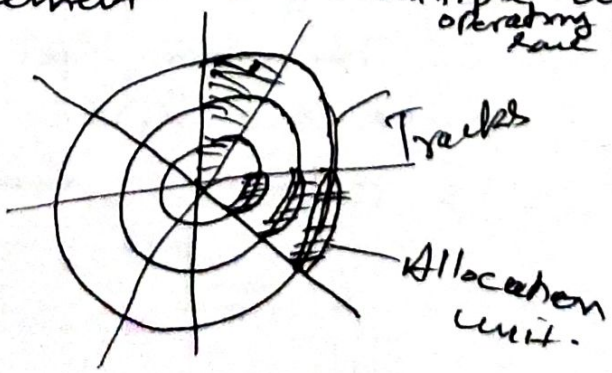


Disk Structure

Where you can't perform R/W operation

* Seek time – To Put read/write head on the respective track
* Latency time – To Put R/W head on respective Sector.

Multiple disk operating in the same system



Hard disk → IDE (Integrated device electronics)
→ SCSI (Small Computer System Interface)

In this bad block management is done by address translation i.e disk controller will be having some reserve sectors and when some bad block sector encounters the controller will translate the bad sector to new good sector.

→ Bad block management is done by some software that marks all bad sectors.

* Capacity = no of cylinders × No of sectors per track × No of surface × Allocation unit

* Allocation unit size is the block size for each block. The decision of which size to set it at is a matter of balance between speed and not wasting harddrive space.

Ex- If allocation unit size is 4 MB means we can put 4 MB size of file in one block.

Disk Access Time :-

1) Seek Time — Time taken by R/W head to reach to desired track.

2) - Rotation time :- Time taken for one full rotation (360°)

3) - Rotational latency :- Time taken to reach to desired sector (half of Rotation time)

4) → Transfer time = Data to be transfer / Transfer rate

Transfer rate = No of heads × Capacity of one track × No of rotations in one second
or
Data rate

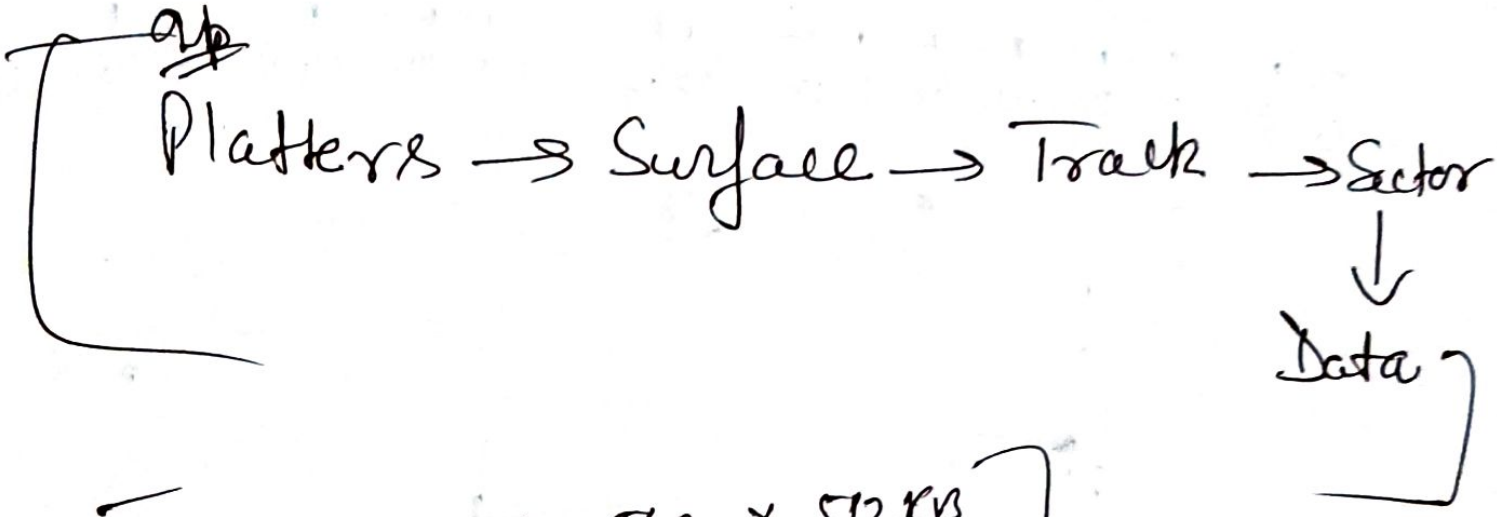
Total access time = ST + RT + Transfer time + CT (Controller time) ↓ optional + Queue time

Q-2 - FCFS - A disk contains 200 track/block (0-199). Request queue contains track no 82, 170, 43, 140, 24, 16, 190 respectly. Current position of R/W head = 50

Calculate total no. of track movements by R/W head. (Seek time)

(⇐ Performance Issue)

block → It is a group of sectors that the O.S can address (Point to). A block might be one sector, or it might be several sectors (2, 4, 8 - - -)

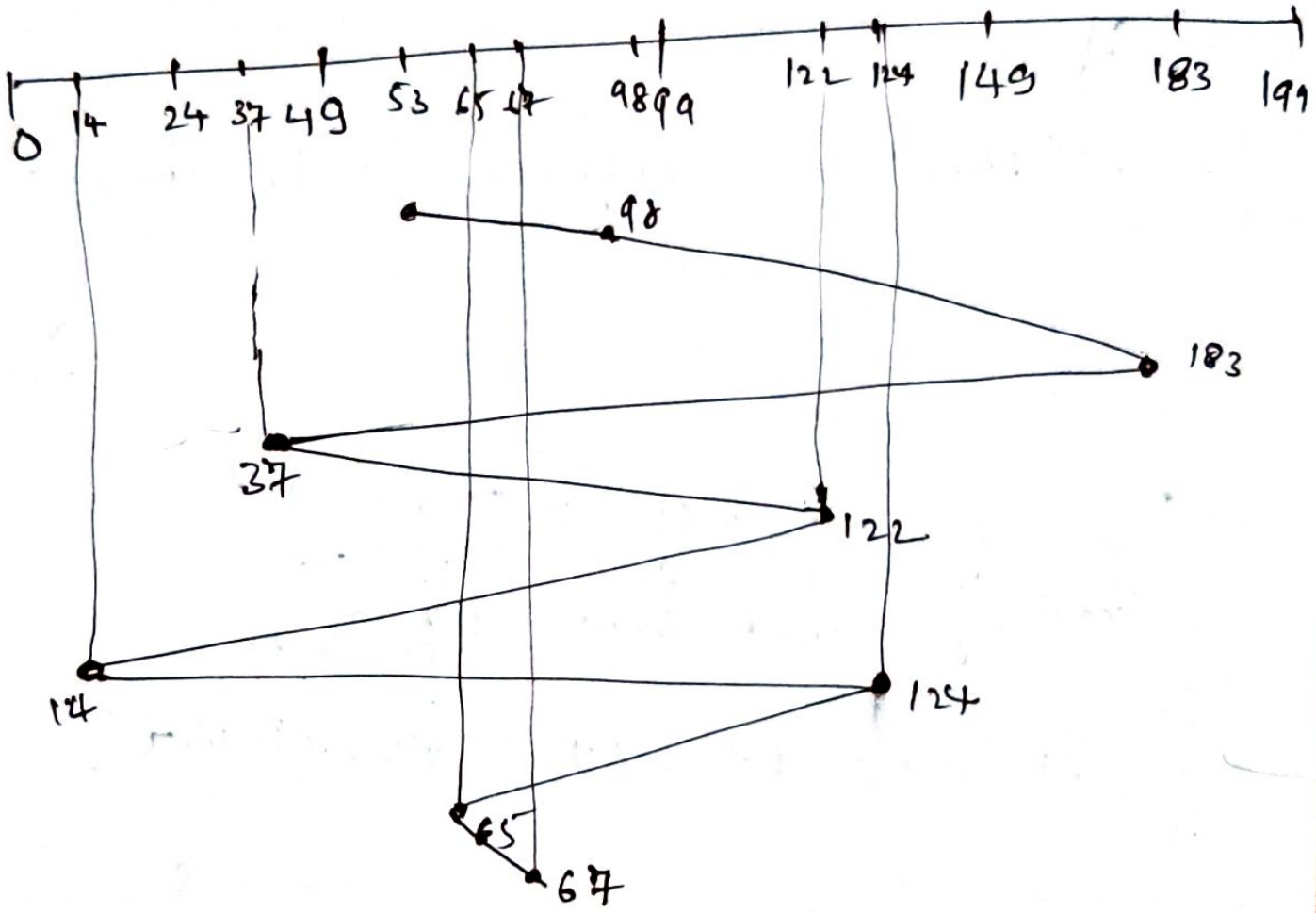


FCFS Scheduling →

Question Suppose a disk is having 200 cylinders numbered from 0 to 199. The disk is currently servicing at cylinder 53 and previous request was at cylinder 60. The queue of pending requests in FIFO order is — 98, 183, 37, 122, 14, 124, 65, 67

- Calculate the total distance ~~the~~ the read/write head will traverse.

Soluⁿ

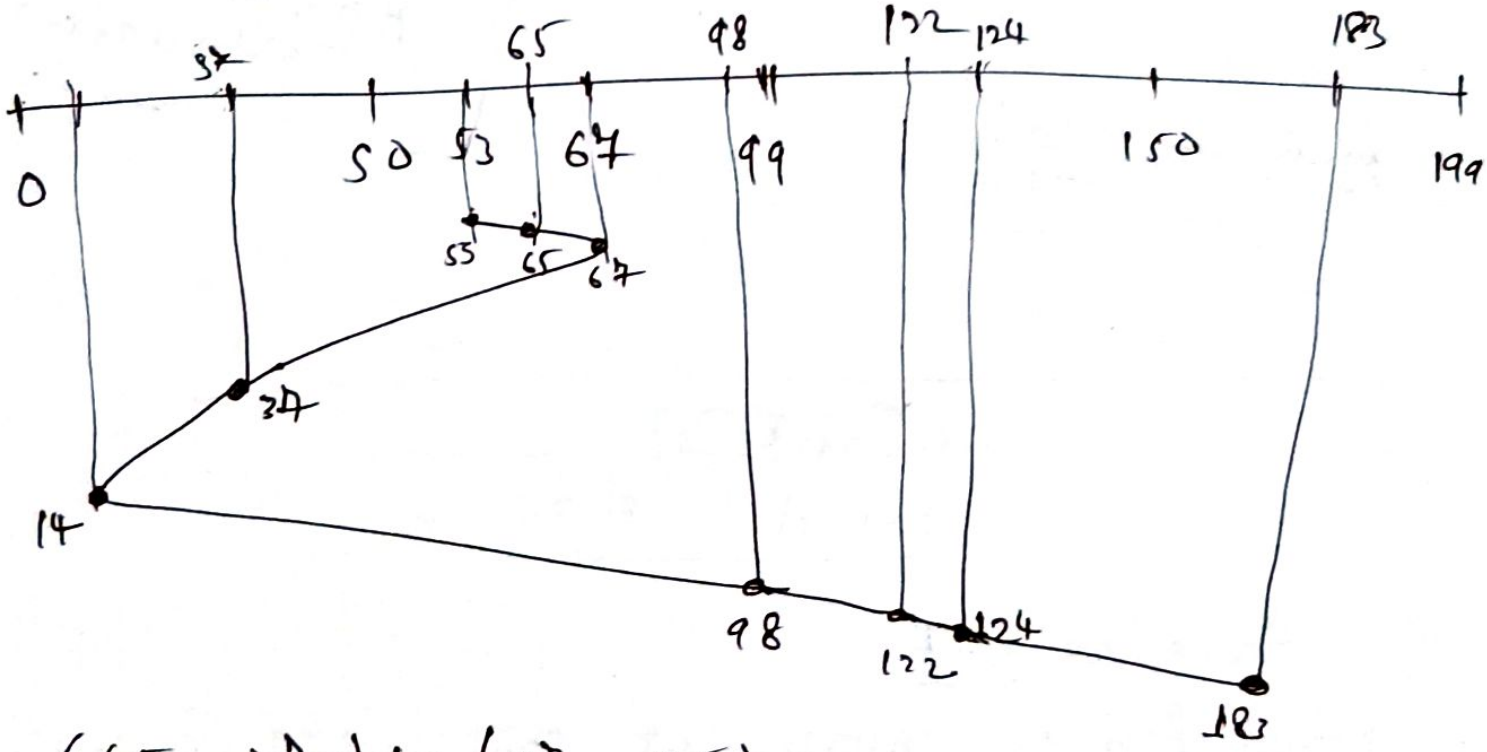


$$(98-53)+(183-98)+(183-37)$$
$$+(122-37)+(122-14)+(124-14)$$
$$+(124-65)+(67-65)$$
$$=48+85+146+85+108+110$$

SSTF → Shortest Seek Time First

Q— Same

Logic — Go to that cylinder which is having shortest distance from current



$(65-37)+(67-65)+(67-37)+(37-14)$
$+(98-14)+(122-98)+(124-122)$
$+(183-124)$

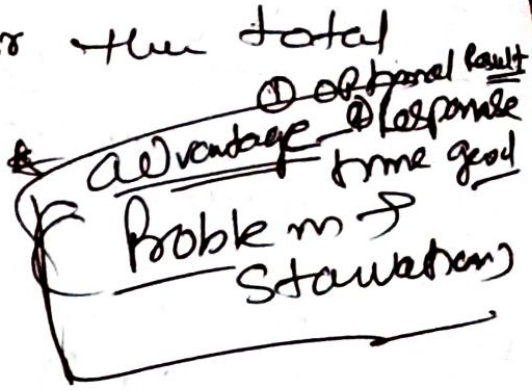
$= 12+2+30+23+84+24+2+59$

$= 236$

SSTF — Same as Q — 2

(ii) If H/w head takes 1 NS to move from one track to another then total time taken. — ?

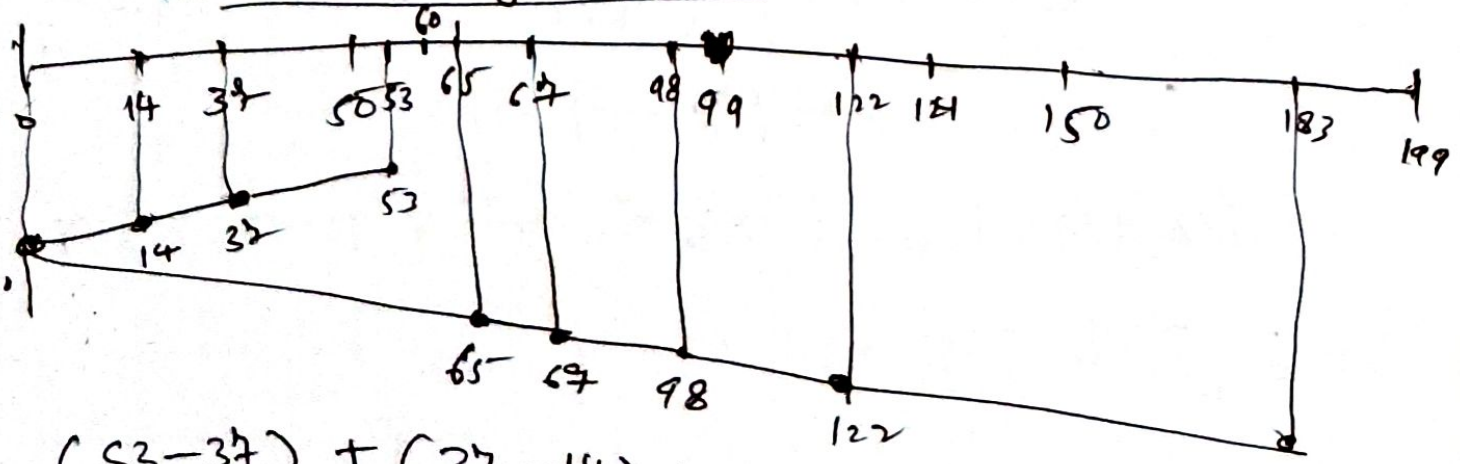
Ans — 208



SCAN Algo

Q — Same as Q-1

Note — from the question we can see that initially the H/w head was at 50 and Now at 53. So, we can say that the H/w head is inclined to decreasing cylinder.



(53−37) + (37−14) + (14−0) + (65−0) + (67−65)
+ (98−67) + (122−98) + (124−122) + (183−124)
= 16 + 23 + 14 + 65 + 2 + 31 + 24 + 2 + 59 = 236

SCAN Contd —

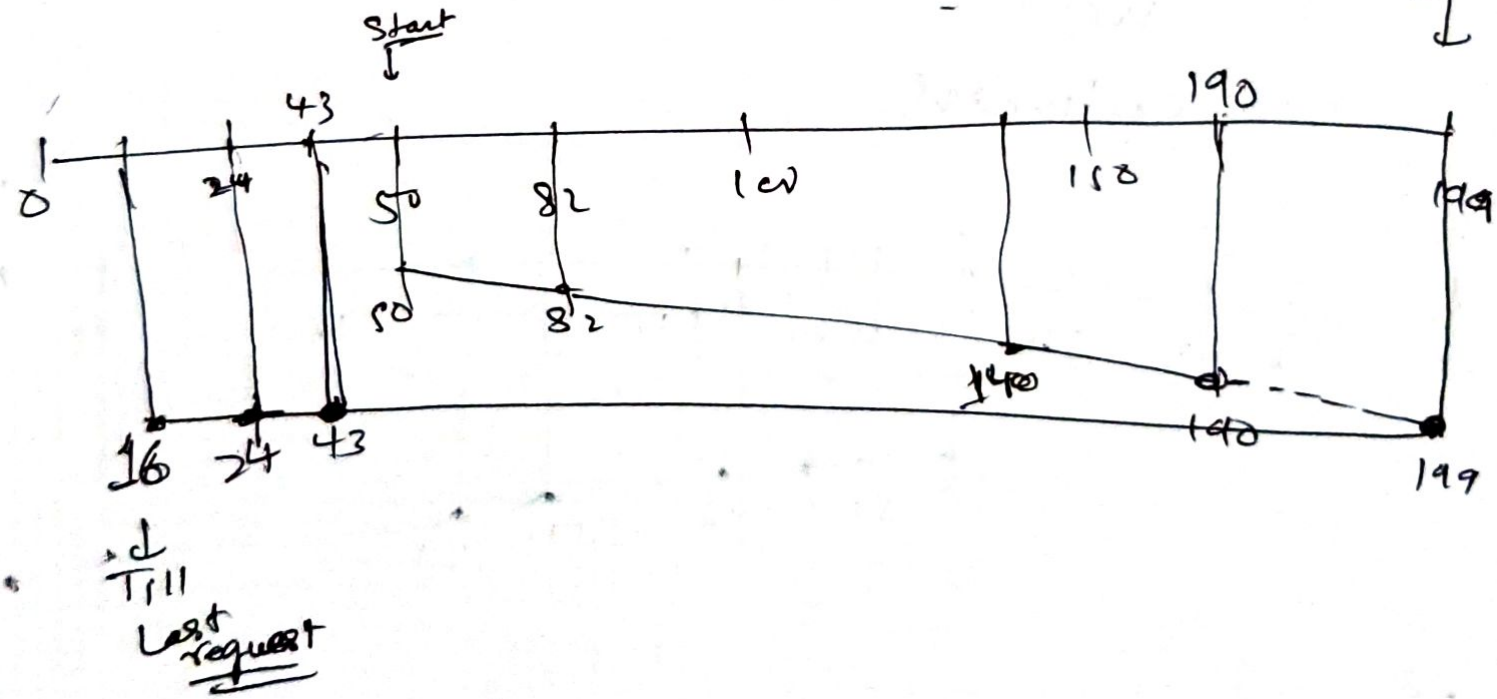
(or Elevator Algo)

NB The R/w head has tendancy to move toward one side i.e towards larger or smaller value
- It will be given in question

82, 170, 43, 140, 24, 16, 190

Current R/w head = 50

⇒ Here direction towards the larger value



Ans — 332

So, 332 × 1 ms

= 332 ms Ans

C-SCAN

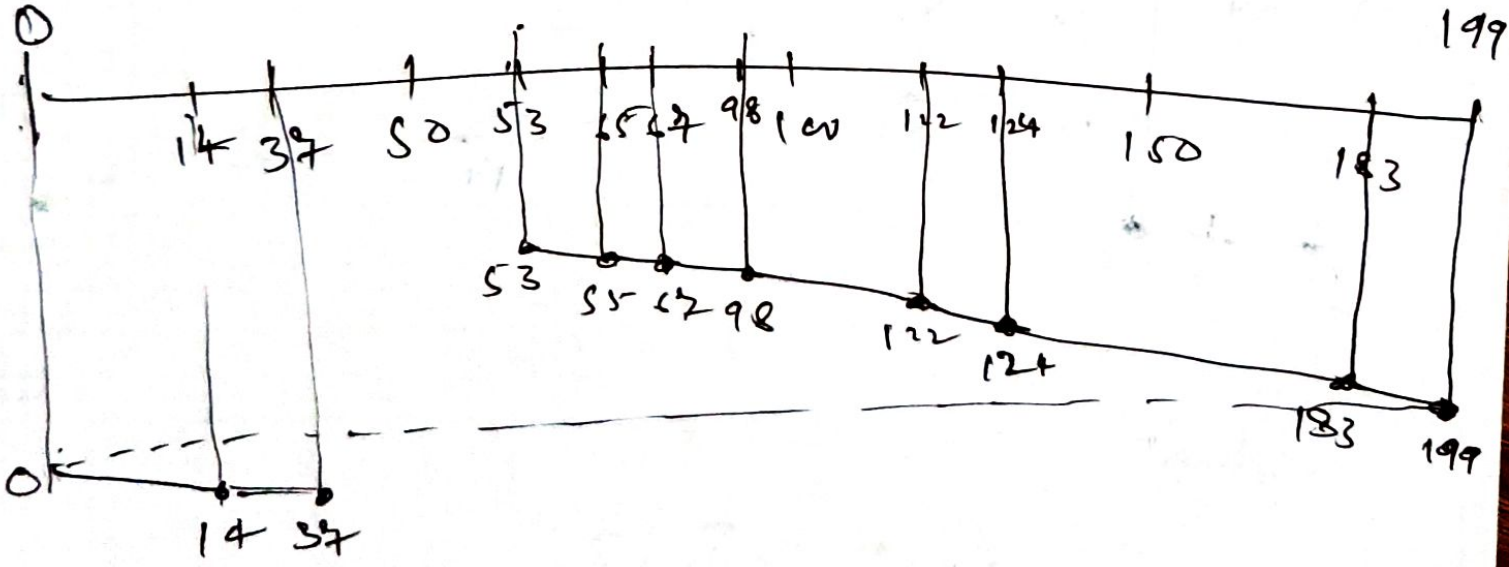
C → Circular

Q— Same Q—1

1st Assumption — If we have 0—(n-1) Cylinder no. then next Cylinder no will be 0 (because its a Circular).

2nd — Assumption — It will move in the increasing order of cylinder no

200 cylinders



(65-53) + (67-65) + (98-67)
+ (122-98) + (124-122) + (183-124)
+ (199-183) + (14-0) + (37-14)
= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 14 + 23
= 183

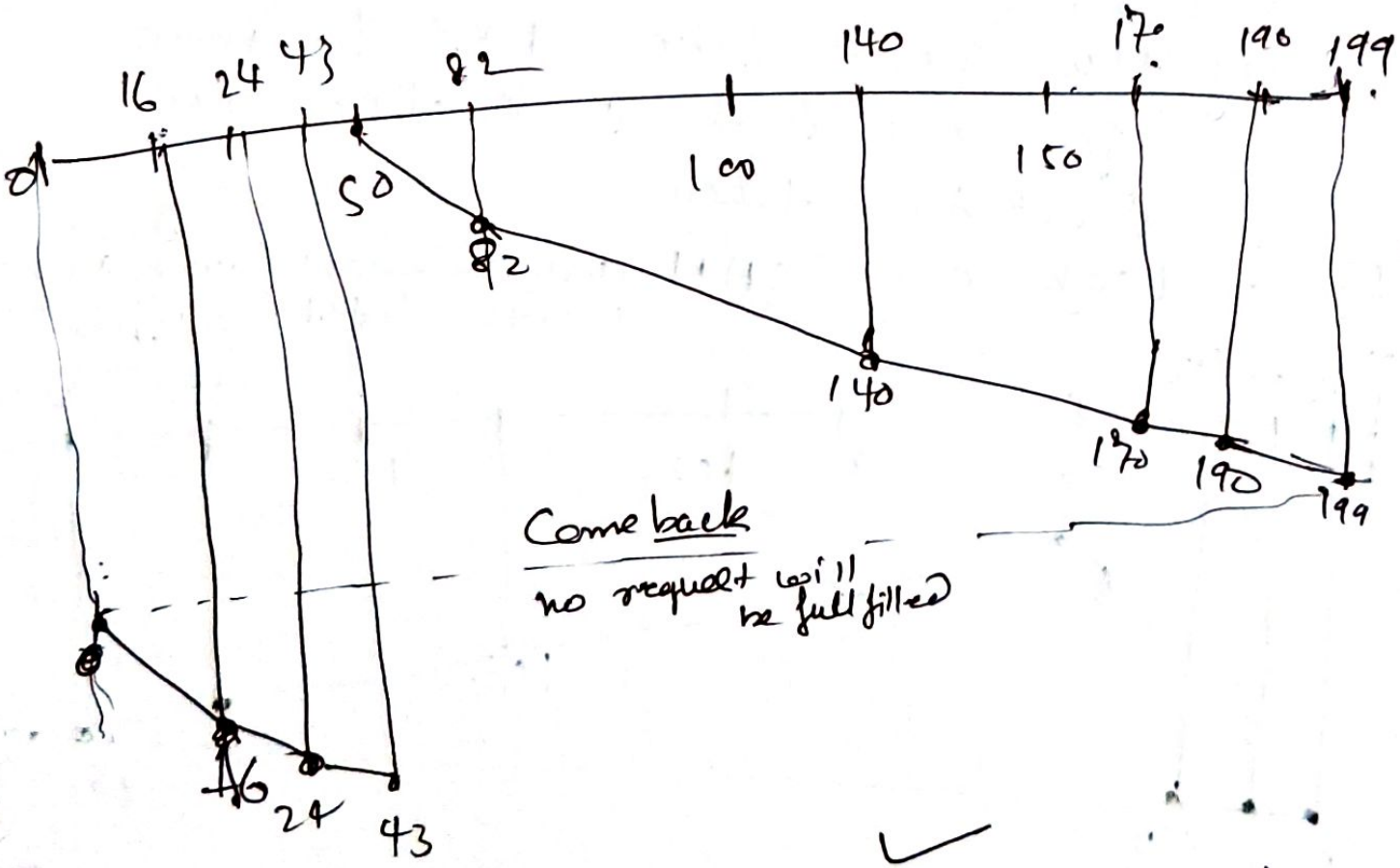
C-SCAN Contd —

Q- Same as Q-2

82, 190, 43, 140, 24, 16, 190

Current H/W = 50

Direction towards large value



(199-50) + (199-0) + (43-0)

≈ 391

VIMP

SCAN & LOOK both we must know the direction

LOOK Algo

Q— Req tracks (Direction towards larger value)

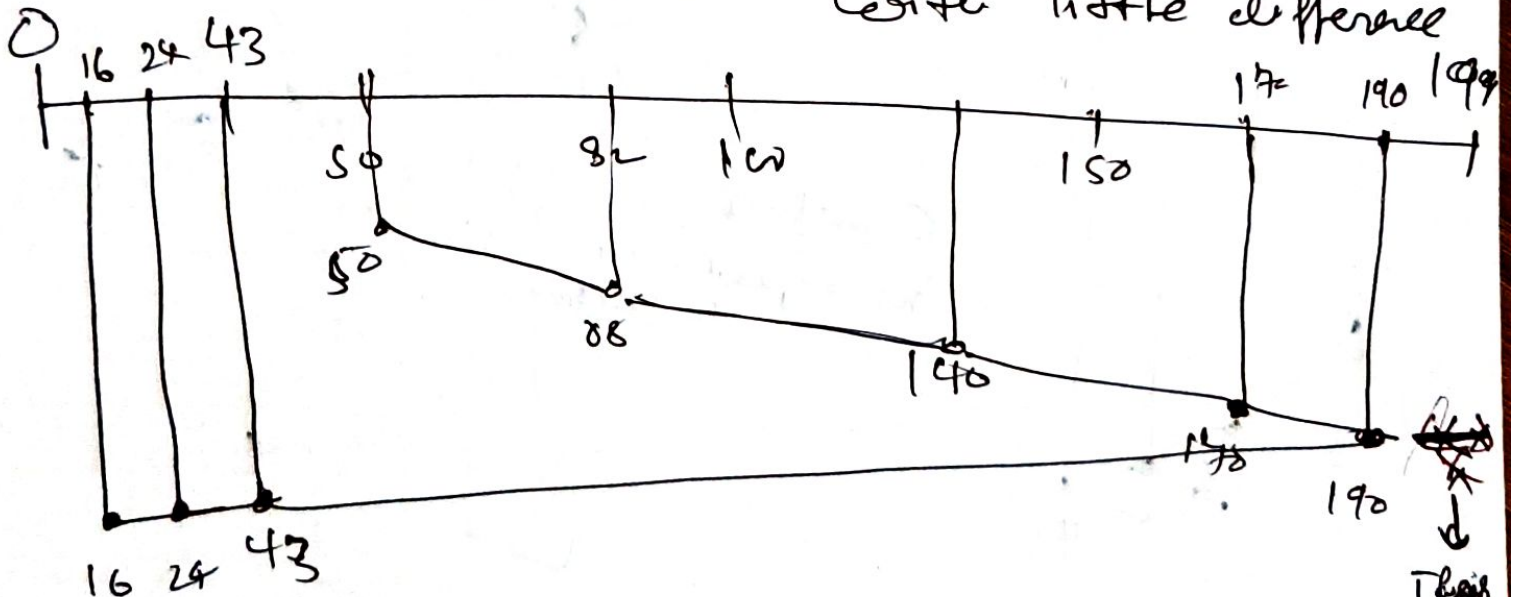
82, 170, 43, 140, 24, 16, 190 respectively

Current R/W head = 50

① Calculate total No of track movement using LOOK?

② if R/W heads takes 1 ms to move from one track to another the total time taken.

NB — LOOK & SCAN are somehow similar with little difference



This extra in SCAN

$(190-50)+(190-16)$

$= 314$ (which is less than SCAN)

314×1 ms

C-Look

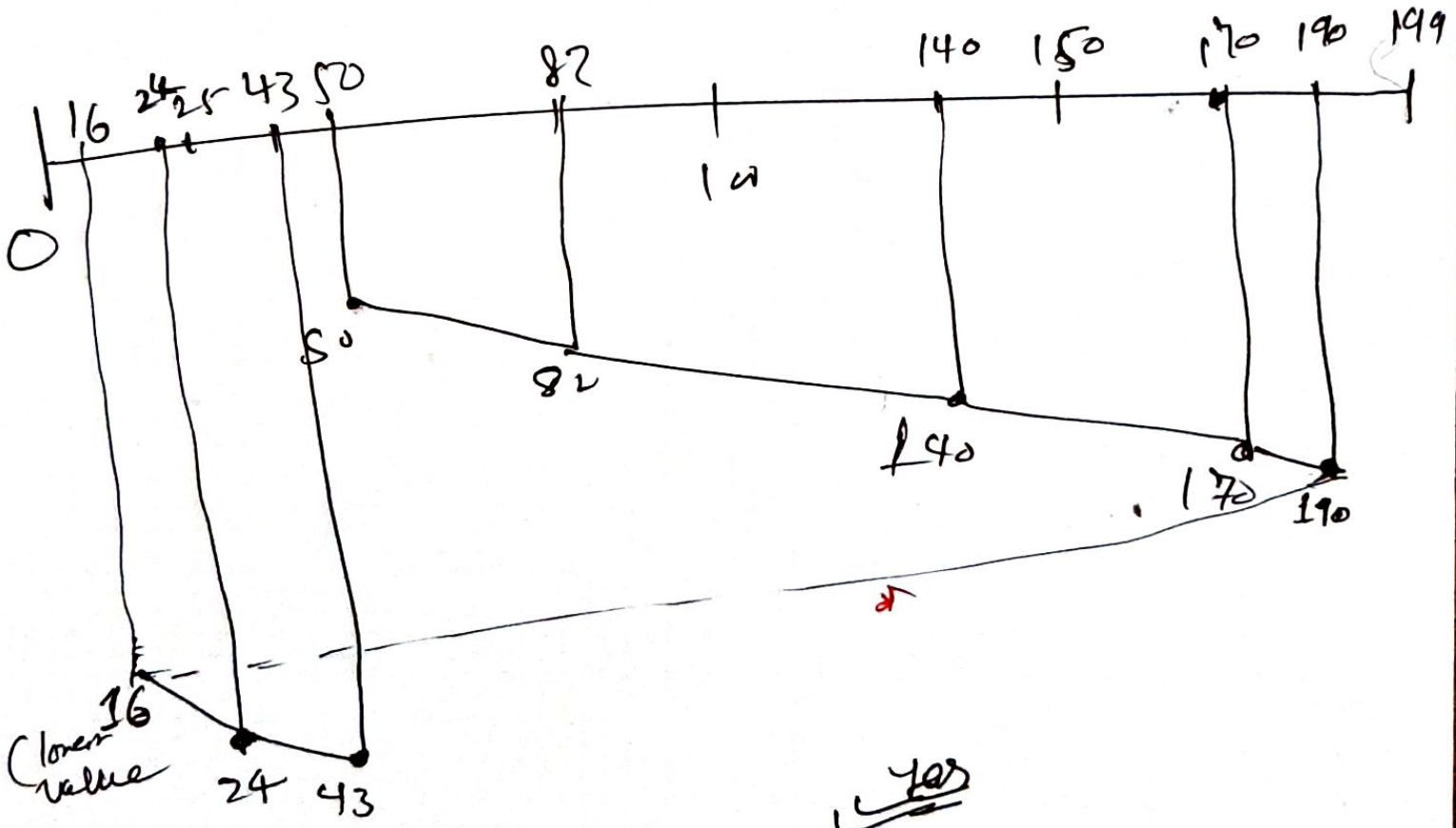
Q- 200 tracks (0-199)

82, 170, 43, 140, 24, 16, 190 respectively

current R/W head = 50

using C-Look

* Direction is towards large value.



Ans

(190 - 50) + (190 - 16) + (43 - 16)

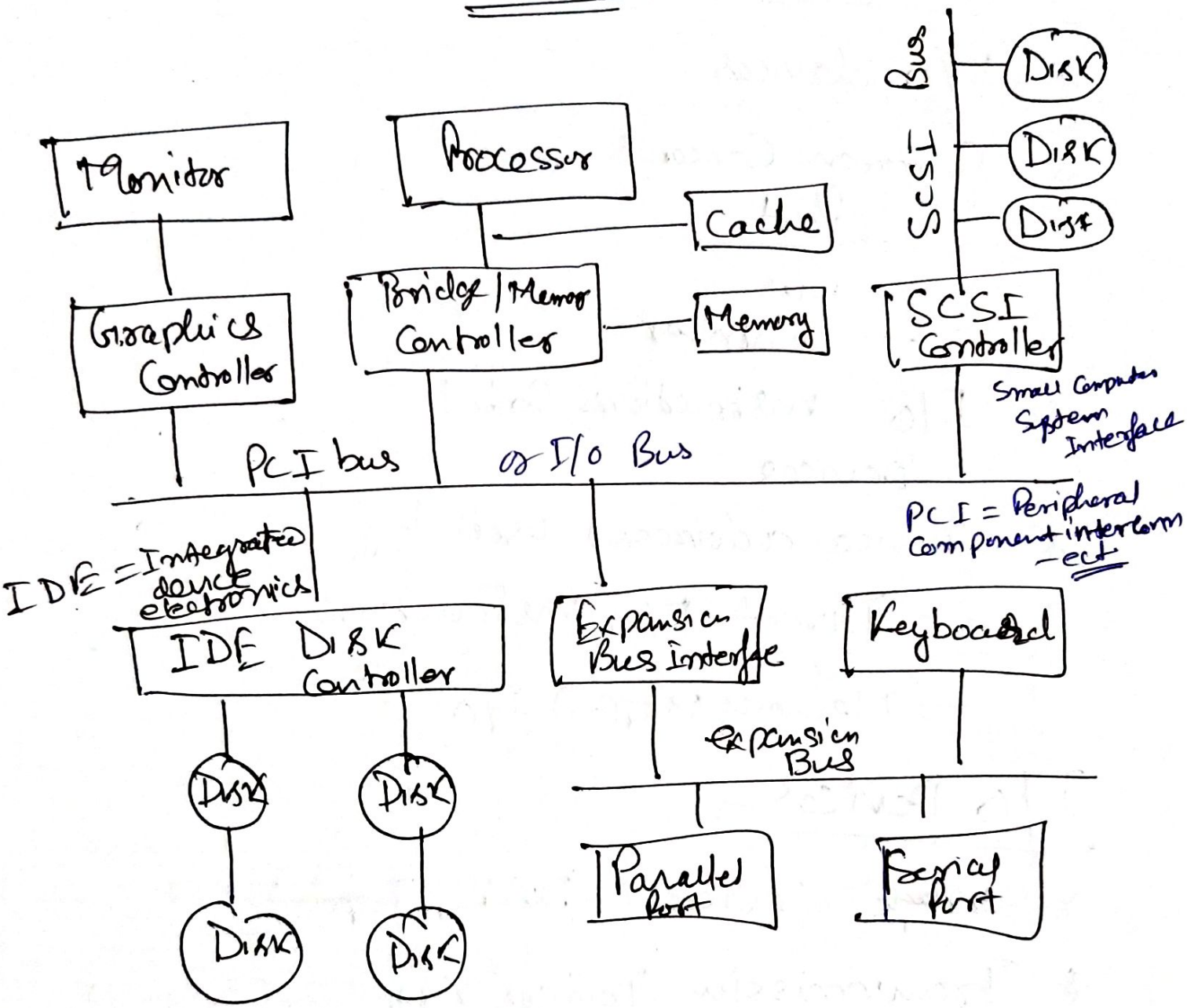
= 341 Ans

Q- Q-1 with C-Look [Here * part is not added]

153 Ans

SCSI — Small Computer System Interface

I/O



A typical PC Structure

Two categories

① Character oriented Device — Byte Communicable

② Block oriented Device — a chunk of data is transferred

* Means a single byte can be transferred in the device / system

(Mouse, Keyboard, Printers — character oriented device

(disk — block oriented device

I/O Hardware →

* I/O devices
* Common Concepts
  - Port
  - Bus
  - Controller
* I/O Instructions Control Devices
* Device addresses used by
  - Direct I/O instructions
  - Memory mapped I/O

I/O Devices →

* Storage devices (Disk, Tapes)
* Transmission Devices (Network Card, MODEM)
* Human Interface devices (Screen, Keyboard, Mouse)
* Specialized device (Joystick)

Device Controllers & I/O units typically

Consists of

- A mechanical component, device itself
- A electronic component, adapter or controller

* low level interface between controller and device

I/O Controller

* DISK Controller
  - DISK Side Protocol (: bad error mapping, buffering, Prefetching, Caching)
* Controller has 3 registers for data and Control
* CPU and Controller Communicate via
  - I/O instructions and registers
  - Memory mapped I/O

I/O Ports :

* 4 register — Status, Control, Data-In, Data-out

Status- States whether the current command is completed, byte is available, any error etc

Control — Host determines to start a I/O Command or to change the mode of the device

Data In — Host reads to get input

Data-out → Host writes to send output

* Typical Size of register is 1 to 4 bytes

Three Techniques for Performing I/O are:-

(i) Programmed I/O → The Processor issues an I/O Command on behalf of a Process to an I/O module; the Process then busy waits for the operation to be completed before Proceeding.

(ii) Interrupt-Driven I/O → The Processor issues an I/O Command on behalf of a Process
- If non-blocking — Processor Continues to execute instructions from the Process that issued the I/O Command
- If blocking — OS will put the current Process in blocked state and Schedule another Process

(iii) Direct Memory Access (DMA) — A DMA module controls the exchange of data between main memory and I/O module

I/O Kernel Subsystem →

* **Scheduling**
  - I/O request queue for a device
* **Buffering** - Storing data in memory while transferring between devices
  - Device speed mismatch
  - Device transfer size mismatch
* **Caching** → fast memory holding copy of data
  - Key to performance
  - Faster storage of data item that resides elsewhere
* **Spooling** → Hold out for a device
  - If device can serve one request at a time
  e.g. - Printer
* **Device Reservation** → Provides exclusive access to a device
  - System calls for allocation deallocation
  - Watch out for a deadlock

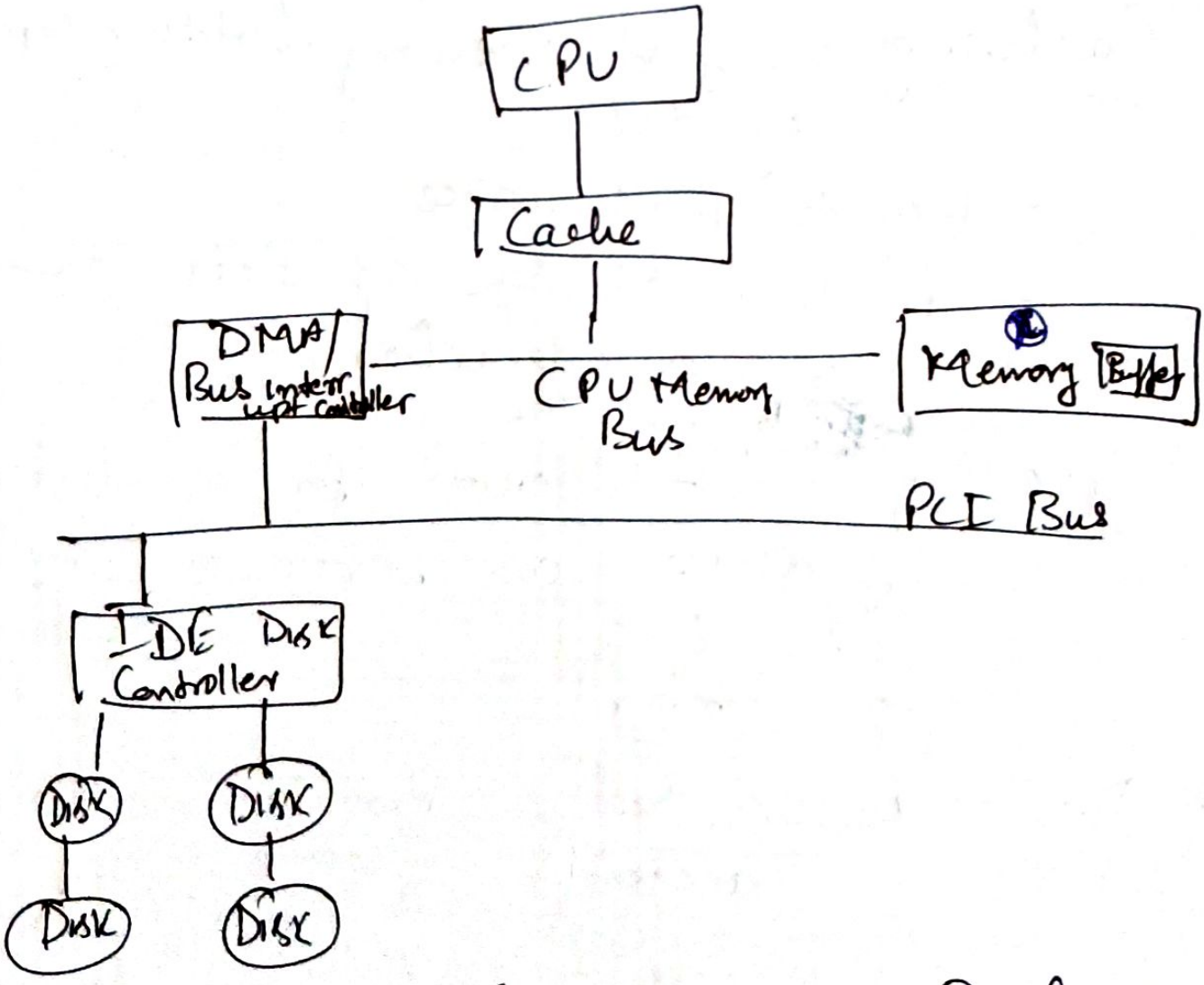
* Error handling: OS can recover from disk read/write transient errors.

— Disk unavailabilities

— System error logs hold problem reports

— Dirty block management.

DMA



{ 6 - Steps to Perform DMA transfer }

① Device driver is told to transfer disk data to buffer at address (x)

② Device driver tells disk Controller to transfer 'c' bytes from disk to buffer at address (x).

③ Disk Controller initiates the DMA transfer.

④ Disk Controller sends each byte to DMA Controller.

⑤ DMA Controller transfers bytes to buffer (x), increasing memory address and decreasing 'c' until c=0.

⑥ when c=0, DMA interrupts CPU to signal transfer completion

Pros and Cons of DMA:

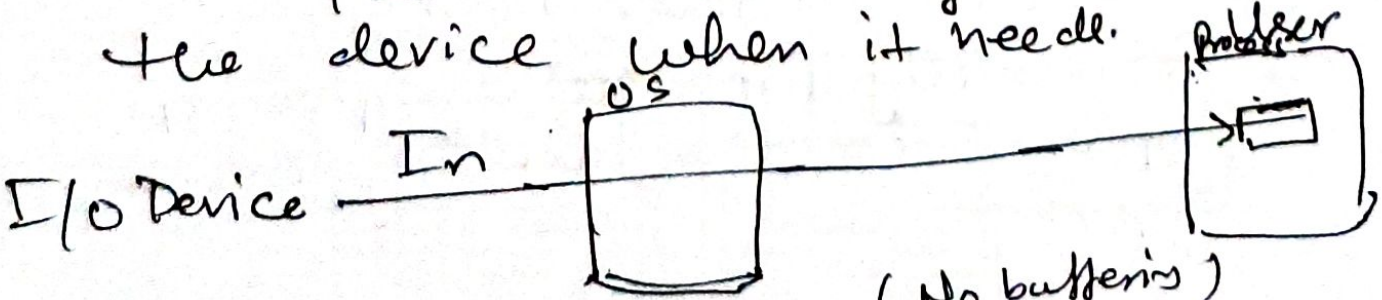
* Pros
  * free up CPU
* use CPU instead of DMA if
  * Device Speed is fast
  * CPU has nothing to do
  * Want to save money (by getting rid of DMA)

Buffering

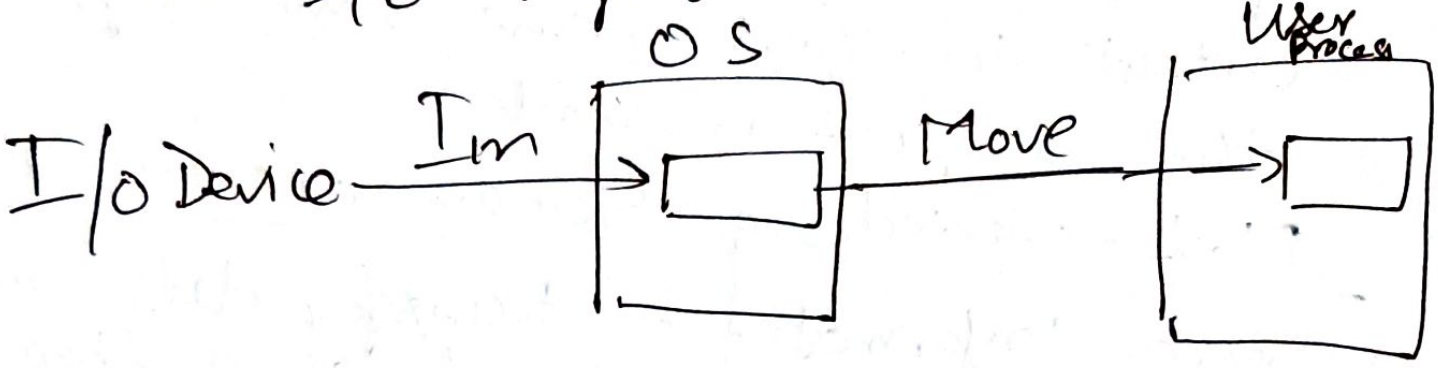
* Perform input transfers in advance of requests being made and perform output transfers some time after the request is made.

Block-oriented device	Stream-oriented device
→ Stores information in blocks that are usually of fixed size	→ transfer data in and out as a stream of bytes
→ Transfers are made one block at a time	→ no block structure
→ Possible to reference data by its block number	→ terminals, printers, communications ports, and most other devices that are not secondary storage are examples

No Buffer → Without a buffer, the OS directly accesses the device when it needs.



Single Buffer:- OS assigns a buffer in main memory for an I/O request



Single buffering

Block-Oriented Single Buffer:

* Input transfers are made to the System buffer
* Reading ahead/anticipated input
  - ⇒ is done in the expectation that the block will eventually be needed
  - ⇒ when the transfer is complete, the process moves the block into user space and immediately requests another block.
* Generally provides a speedup compared to the lack of System buffering

Disadvantages:

→ Complicates the logic in the O.S

→ Swapping logic is also affected

Stream-Oriented Single Buffer

* Line-at-a time operation
  - → appropriate for scroll-mode terminals (dumb terminals)
  - → User input is one line at a time when carriage return signaling the end of line
  - → output to the terminal is similarly one line at a time

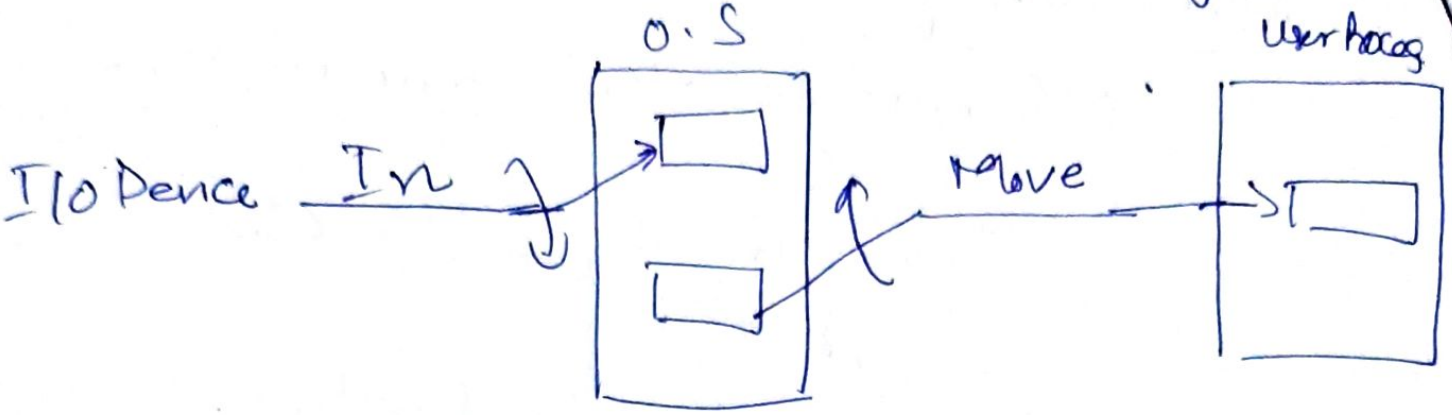
* Byte-at-a-time operation
  - → Used on form-mode terminals
  - → when each keystroke is significant
  - → other Peripherals such as sensors and Controllers

Double Buffer

→ * Use two system buffers instead of one.

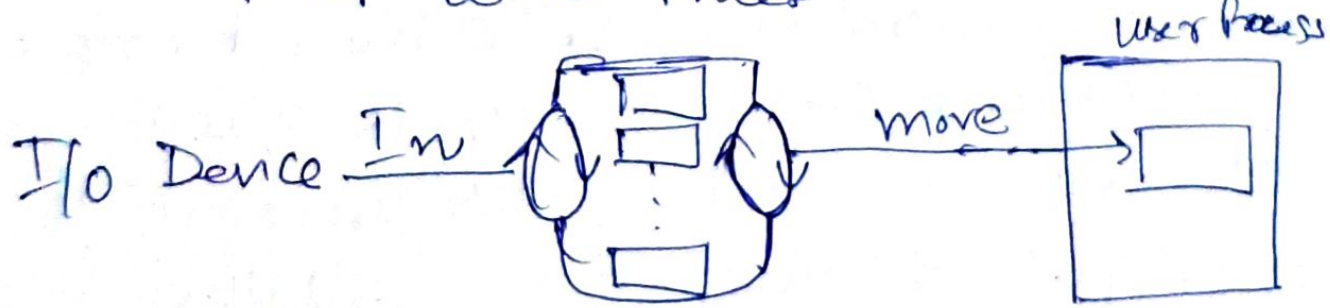
* A process can transfer data to or from one buffer while the O.S empties or fills the other buffer.

* Also known as buffer swapping



Double buffering

Circular buffers
* Two or more buffers are used
* Each individual buffer is one unit in a circular buffer.
* Used when I/O operation must keep up with Process

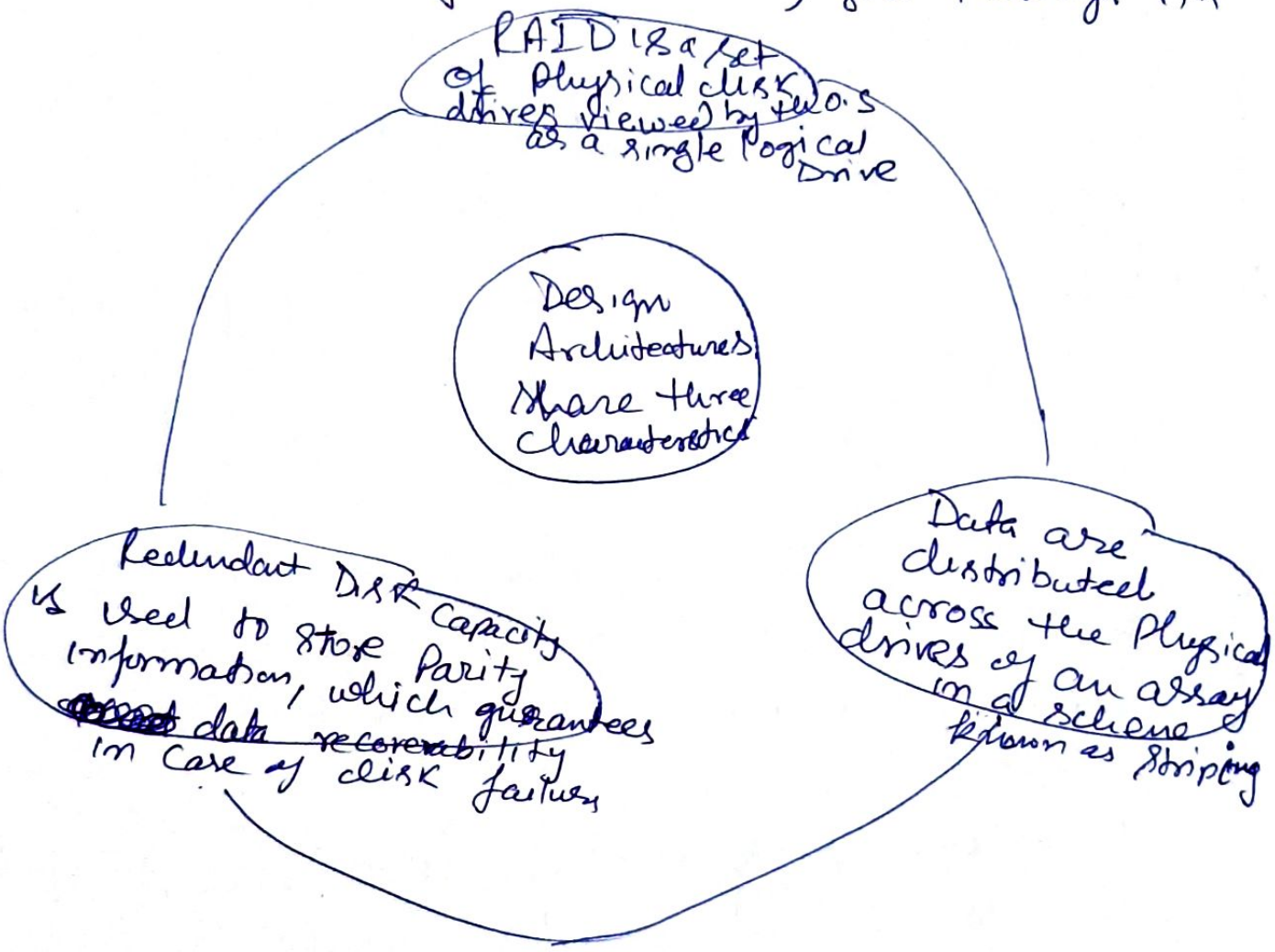


Circular Buffering

The Utility of Buffering
* Technique that smoothes out peaks in I/O demand
  * with enough demand eventually all buffers full and their advantage is lost
* when there is a variety of I/O and Process activities to service, buffering can increase the efficiency of OS and performance of individual process

RAID

* Redundant Array of Independent Disk
  = of inexpensive Disk
* Consists of seven levels, zero through six



⇒ The term was originally coined in a Paper by a group of researchers at the university of California at Berkeley

⇒ The Paper outlined various configurations and applications and introduced the definitions of the RAID levels.

⇒ Strategy employs multiple disk drives and distributes data in such a way as to

enable simultaneous access to data from multiple drives

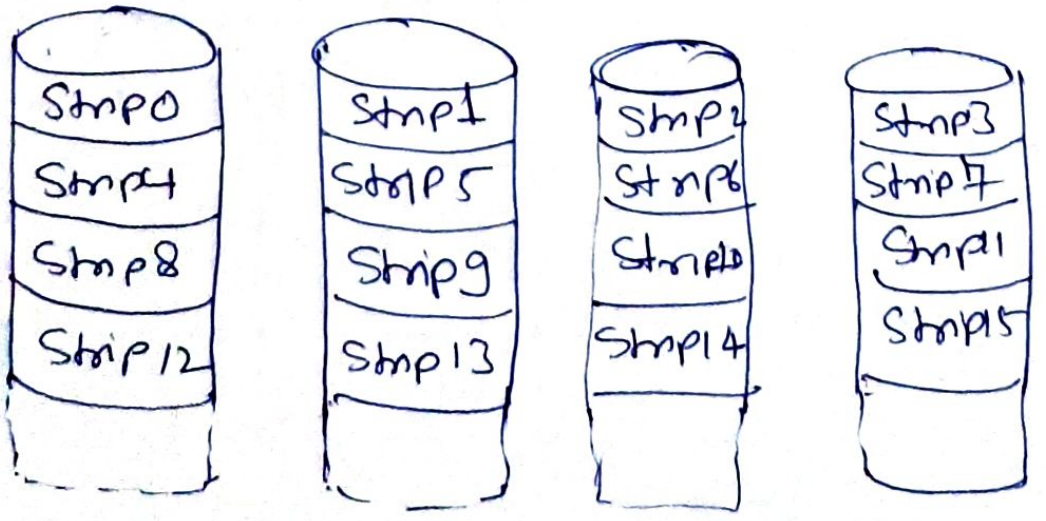
→ Improves I/O performance and easier incremental increases in capacity.

⇒ The unique contribution is to address effectively the need for redundancy.

⇒ Makes use of stored Parity information that enables recovery of data lost due to disk-failure

RAID Level 0 : * Not a true RAID because it does not include redundancy to improve Performance or Provide data protection.

* User and System data are distributed across all of disks in the array
* Logical disk is divided into strips

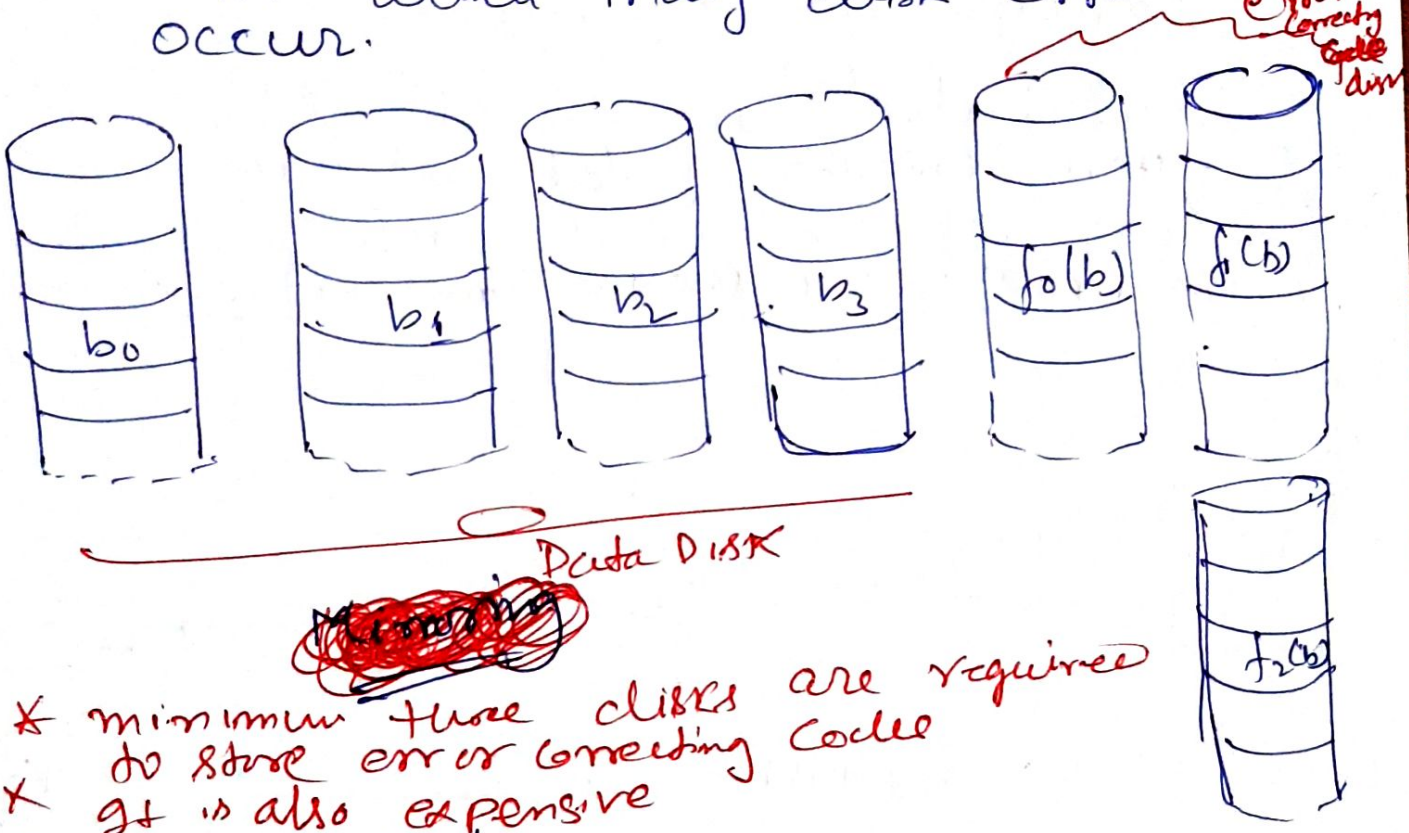


(RAID 0 (non redundant)

RAID Level 2 → * Makes use of Parallel access technique.

(It is obsolete technology we break data at bit level)

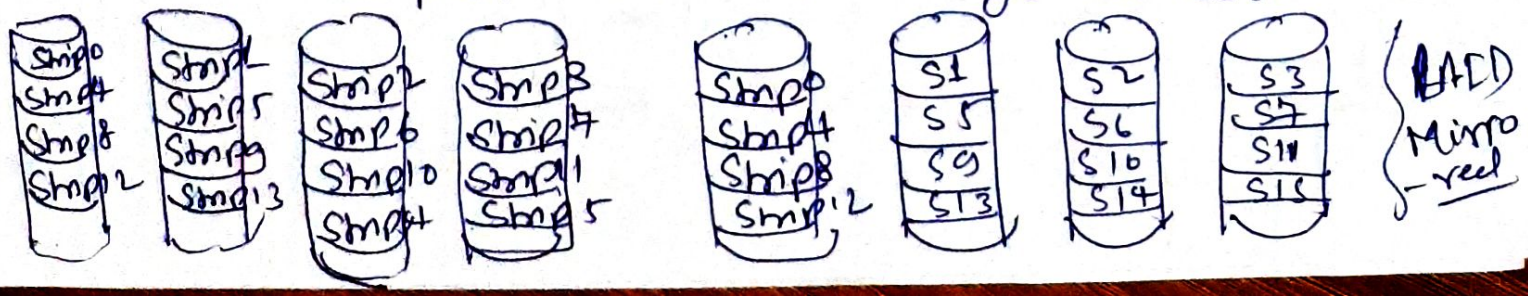
* Data striping is used
* Typically a Hamming Code (error correcting code) is used
* Effective choice in an environment in which many disk errors occur.



* minimum three disks are required to store error correcting codes
* It is also expensive

RAID Level-1 → * Redundancy is achieved by the simple expedient of duplicating all the data. (mirroring)

* There is no "write penalty" (mirroring)
* When a drive fails the data may still be accessed from the second drive
* Principal disadvantage is cost.



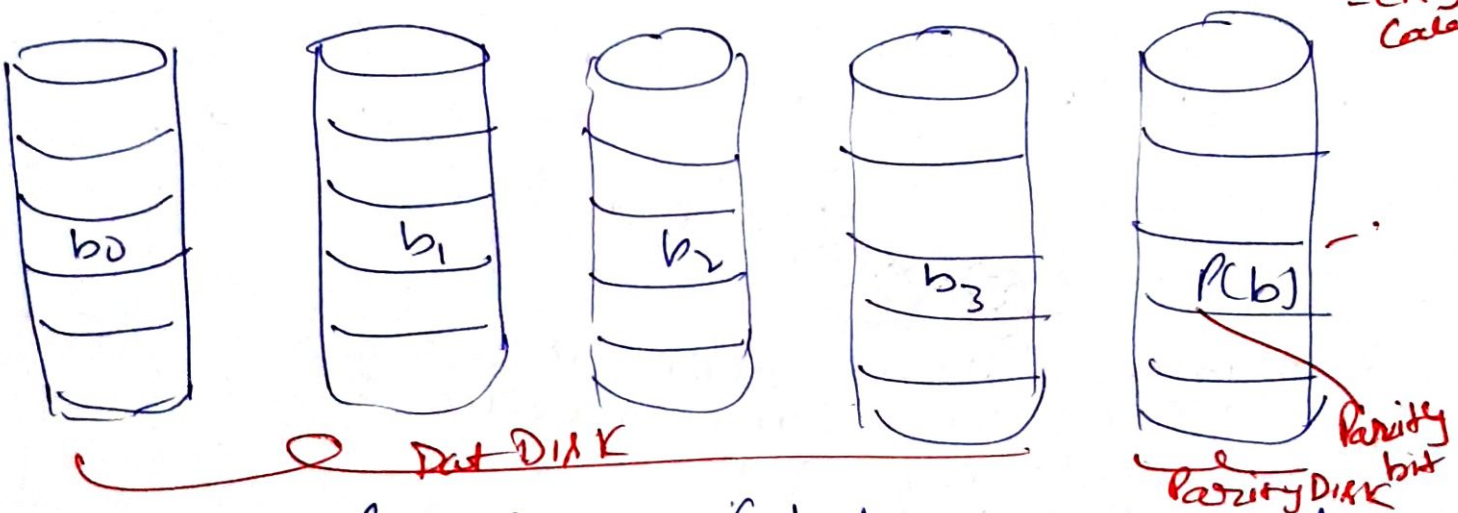
RAID Level 3 ⇒ * Requires only a single redundant disk, no matter how large the disk array.

* Employs Parallel access, with data distributed in small Strips

x Here division of data bit wise

* Can achieve very high data transfer

x In RAID 3 we use the concept of Parity, instead of comparing Codes

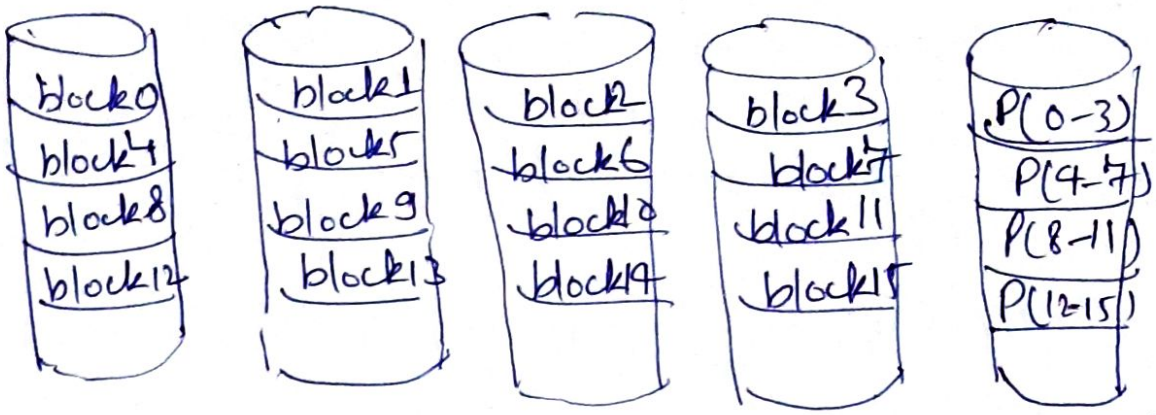


RAID 3 (bit - interleaved Parity)

RAID LEVEL 4 ⇒ * Makes use of an independent access technique.

* A bit-by-bit Parity strip is calculated across corresponding strips on each data disk, and the Parity bits are stored in the corresponding strip on the Parity disk.

* Involves a write Penalty when a I/O write request of small size is Performed

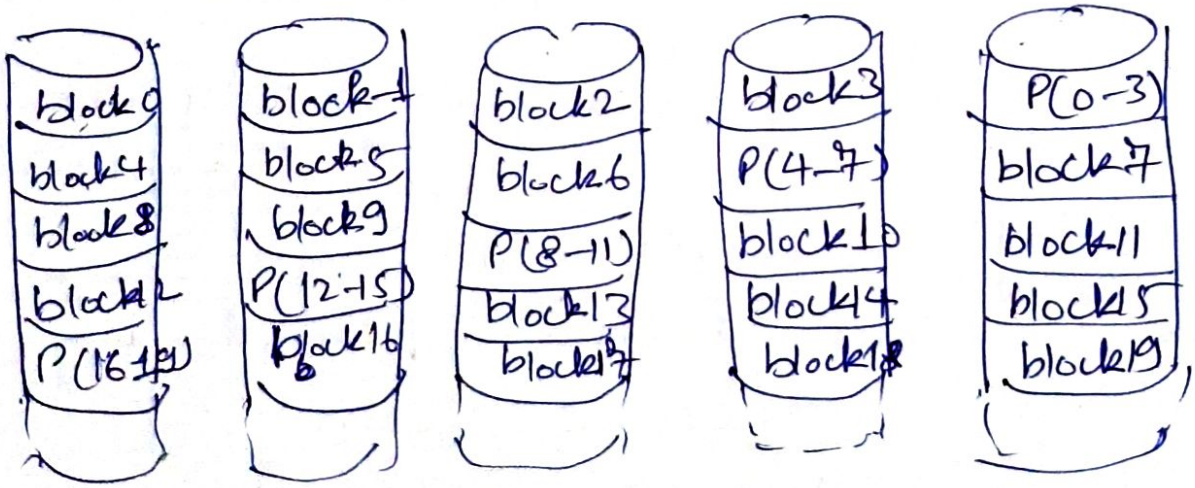


RAID 4 (block-level Parity)

* Here the division of data block wise

RAID Level 5 :- * Similar to RAID-4 but distributes the Parity bits across all disks

* Typical allocation is a round-robin
* It has the characteristic that the loss of any one disk does not result in data loss.

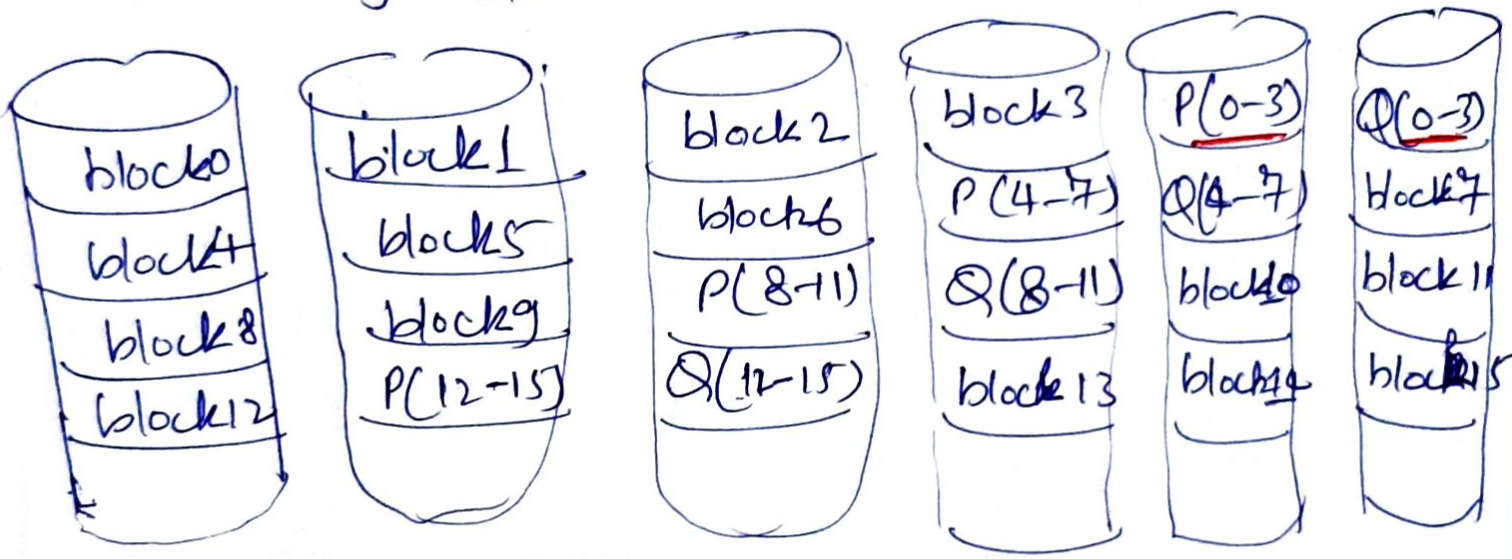


RAID 5 (block-level distributed Parity)

* Here Parity data is distributed in all disks rather than in a single disk as in RAID 4, RADI 3

RAID level 6 - * Two different Parity Calculations are carried out and stored in separate blocks on different disks

* Provides extremely high data availability.
* Incurs a substantial write penality because each write affects two Parity blocks.

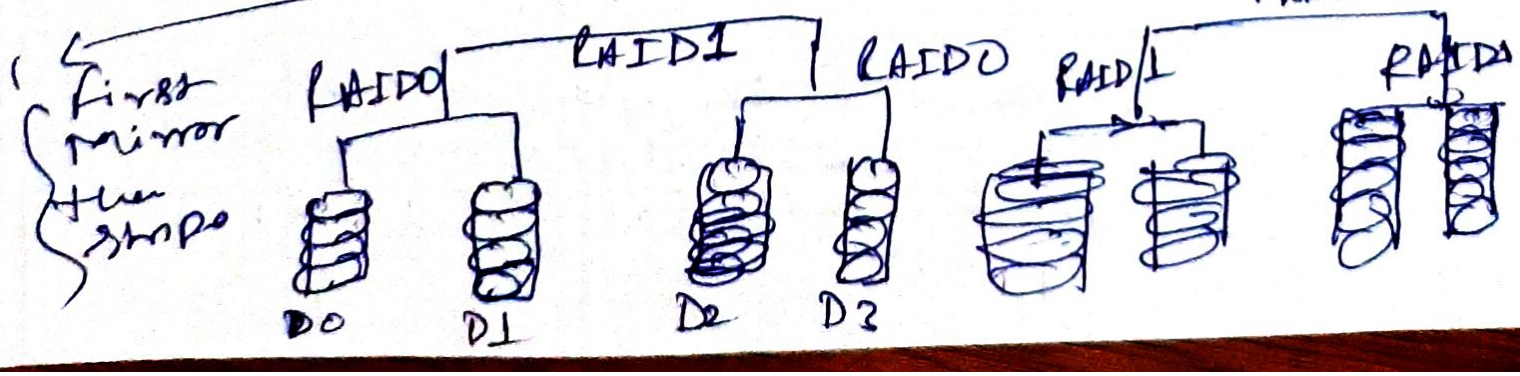


Here we use two Parity ex- P(0-3) Q(0-3)

So if two disk failure occur, we can recover the data from Parity

RAID-10 - It is a hybrid of RAID 0 and RAID 1

RAID 0+1



RAID 1+0 we first stripe then mirror it

File System Management

* a file is a collection of related information defined by its creator

ex — a.txt — Contents → Personal detail, account detail

* In General file is a sequence of bits, bytes / lines or records.

Example of Types of file :

→ Text file → Sequence of characters organized into lines.

a → f1() calling f2() → Source file → Sequence of Sub routines or functions which are executable

→ Object file → Collection of words, organized into loader record Blocks

File Attributes :

Are the Parameters used to keep track of file in O/S.

(i) Name → Human readable form (Filename "a")
(ii) Identifier → Unique id — number (123)
(iii) Type → File type (.txt)
(iv) Location → Pointer to a device and location of file
(v) Size → Current Size of file (1024 KB)
(vi) Protection → Access Control info. (R/W access)
(vii) Time & Date → Creation, modification, deletion
(viii) User Id

file operations :

(i) Creating a file — Two steps
(1) There should be space
(2) Entry for new file

(ii) Writing a file : By using system call
ex= write(a, "ABC")

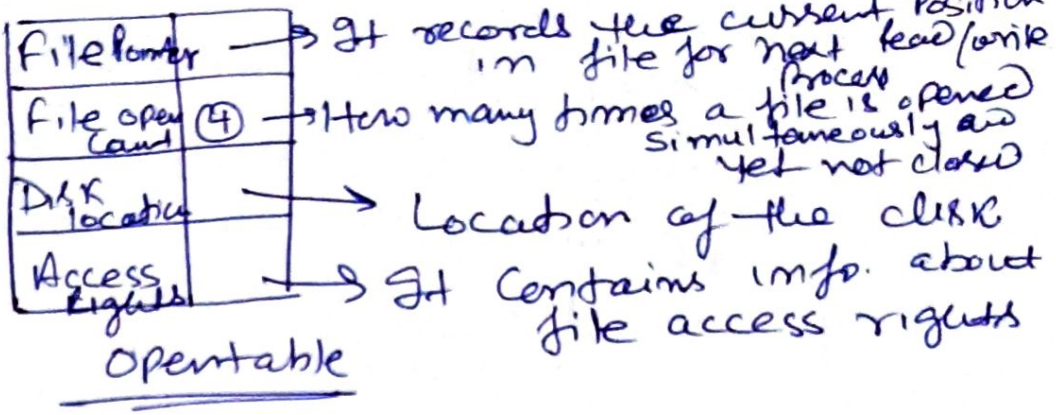
(iii) Receiving a file : By using system call.
read(a)

(iv) Repositioning a file : file seek operation.

(v) Deleting a file → delete file from Entry.

(vi) Truncating a file : If user wants to delete the content of a file but still user wants to hold file's attribute.

Open file Table :- Contains information about currently open files



File locking :- Used to prevent from situations like deadlock

(i) Shared Lock (Read)
(ii) Write (Exclusive lock)
(Read and Write lock)

File Types :-

(i) Executable (Ready to Run Machine language Program) (.exe, .bin)
(ii) Object (Compiled Machine languages) (.obj)
(iii) Source code (Source Code in various languages) (.c, .java, .perl - -)
(iv) Batch (Commands to the Command interpreter Ie .bat, .sh
(v) Markup (textual data, documents) (.xml, .html)
(vi) library — library of routines for Programmers .lib, .dll.
(vii) Multimedia — Binary files Containing audio or video files. .mp3, .mp4, .MPEG,

File Access Methods

(i) Sequential Access :- Emulates magnetic tape operation.

→ one record is processed after other.

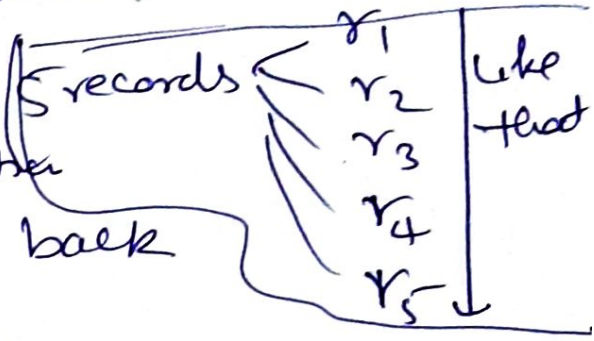
→ Supports following operations:

(1) read next - read records and move pointer to next position

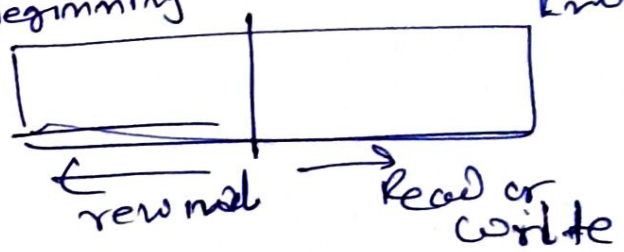
(2) write next - write and advance the pointer

(3) rewind - moving back to earlier location

(4) Skip and records → may or may not be supported by O.S



Skip(2)



(ii) Direct Access :- It is based on disk model. It allows random access. User can jump to any record and access that record. Following operations are supported

(1) read n → Reading record 'n'

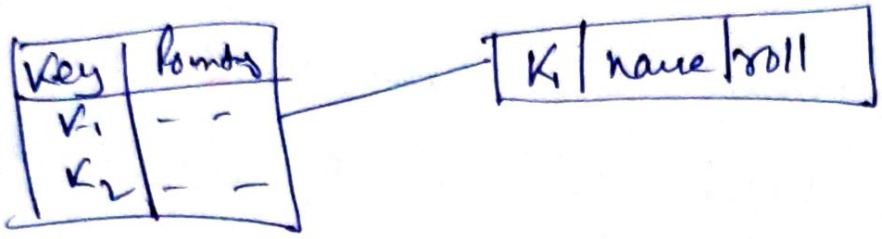
(2) write n → writing record 'n'

(3) jump to record n → It could be used to access to a particular record.

(4) Query current record → used to return back this record later

Ex - QCR

(iii) Indexed Access :- Index is created which contains a key field and pointers to the various blocks



File Allocation Methods

Allocate space to the files so that disk space is utilized in an efficient manner

Factors to Consider :->

(i) Processing Speed < Sequential / Random Access

(ii) Ability to use multisector and multitrack transfer.

(iii) Disk Space utilization

390K -> 500K / 400K

(iv) Main Memory requirement -> should be least

(i) Contiguous Allocation → Each file occupies a set of contiguous addresses on disk

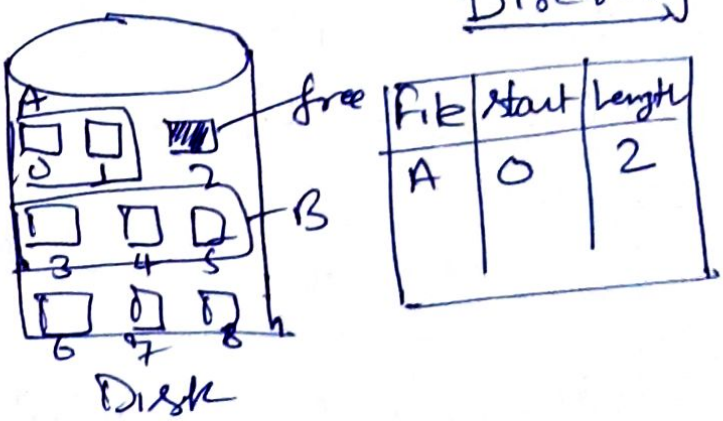
→ Linear ordering

→ Location of a file is defined by the disk address of first block and its length

→ Both sequential and direct/random access are supported.

(b+i) re(3+1)=4th record of file

Directory



Disadvantages :

(i) finding space for new file

(ii) External fragmentation.

Contiguous Allocation Application :- Dynamic Storage Allocation

(i) First Fit

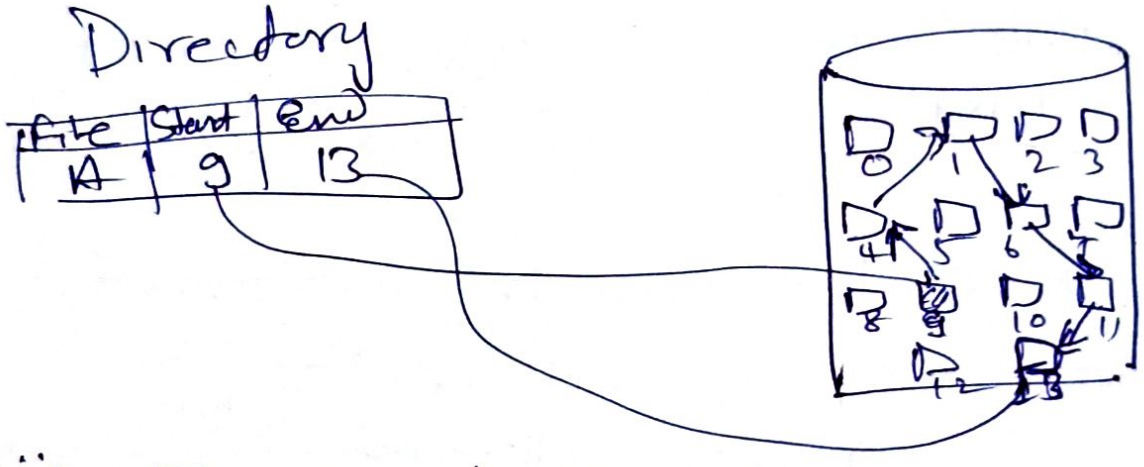
(ii) Best Fit

(iii) Worst Fit

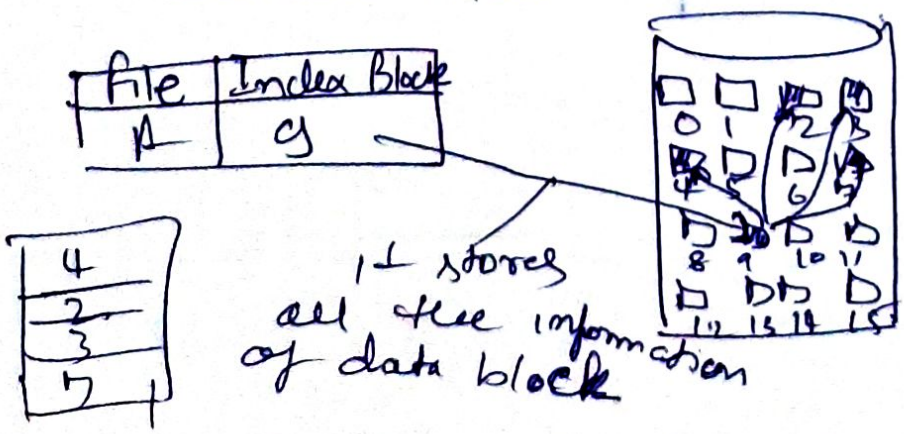
(ii) Linked Allocation → Solves all Problems of Contiguous allocation. Each file is a linked list of disk blocks

→ No External fragmentation (Advantage)

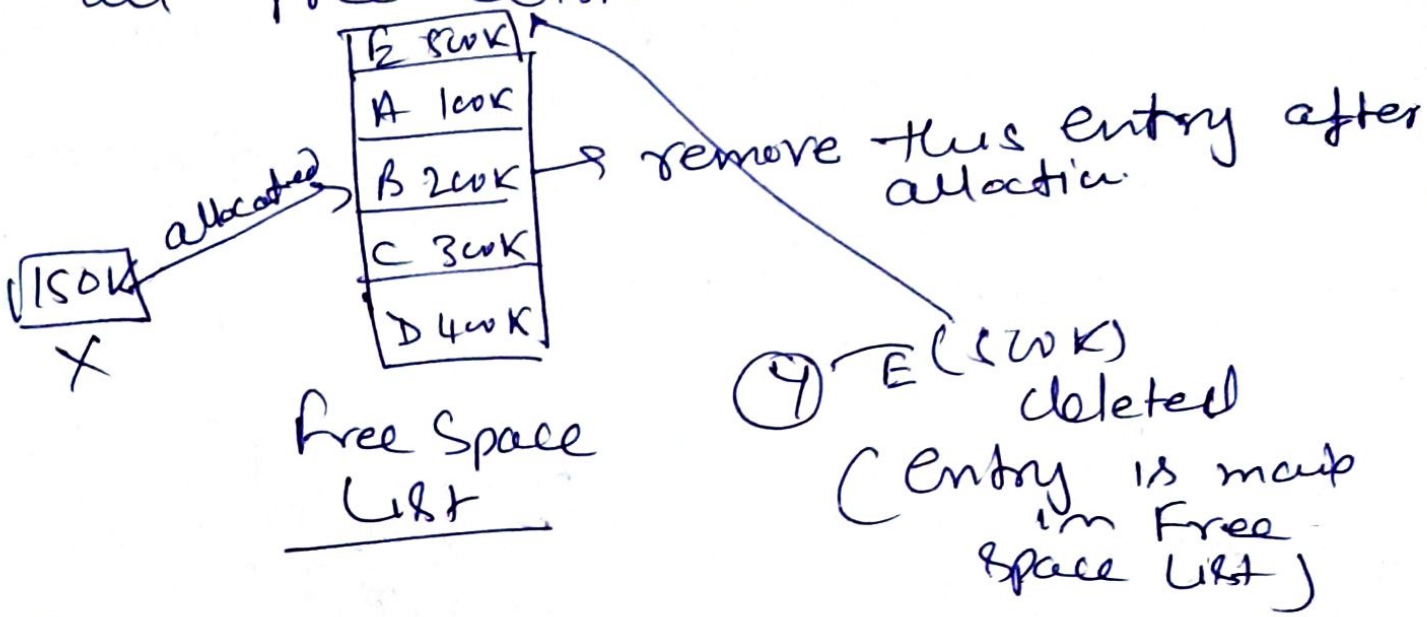
→ Can be used for sequential-access file (Disadvantage)



(iii) Indexed Allocation → Solves the Problem of linked allocation (No Random access). In this all the Pointers are brought together into one location called indexed block. Each file has its own indexed block



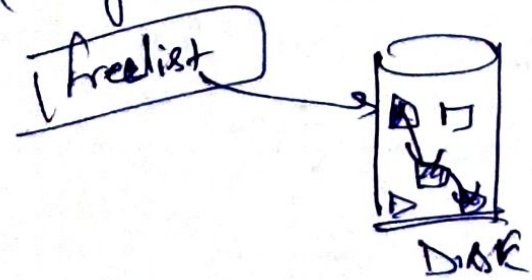
Free Space Management → free-space list is used to keep track of all free disk-blocks



⇒ Free space list can be implemented as bit-map or bit vector.

Block ← Allocated (0) bit
Free (1) bit

⇒ Another approach is to link together all free disk blocks



File Directories :- A physical Disk can be broken up into multiple partitions, or mini-disks.

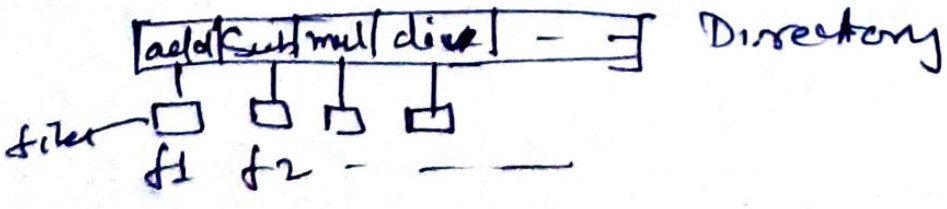
Op directory is a symbol table that translates file names into their directory entries.

Operations on directory:

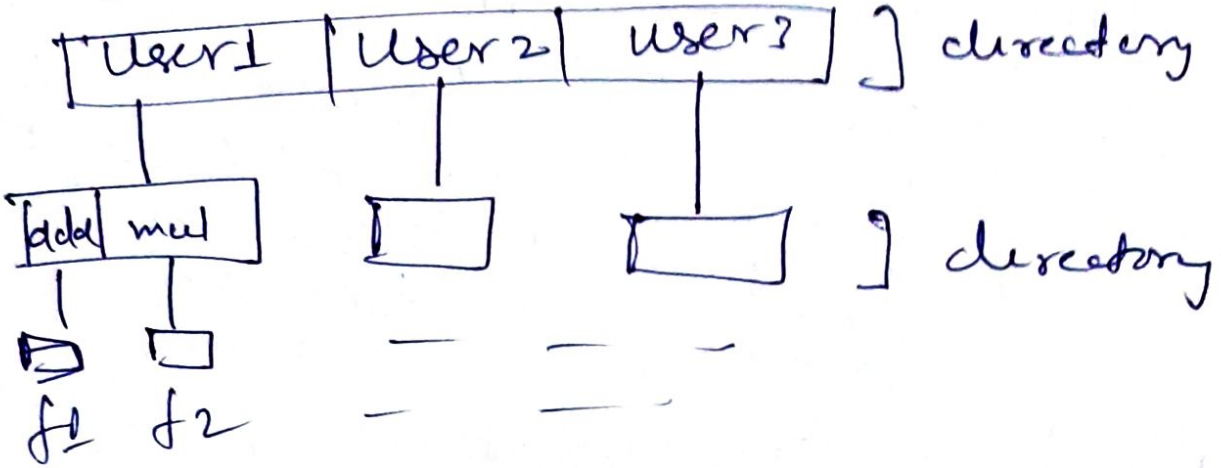
(i) Search for a file
(ii) Create New file
(iii) Delete a file
(iv) List a Directory
(v) Rename a file
(vi) Traverse file System

(i) Single-level Directory :- All the files are contained in same directory

Op :- Each file must have unique name

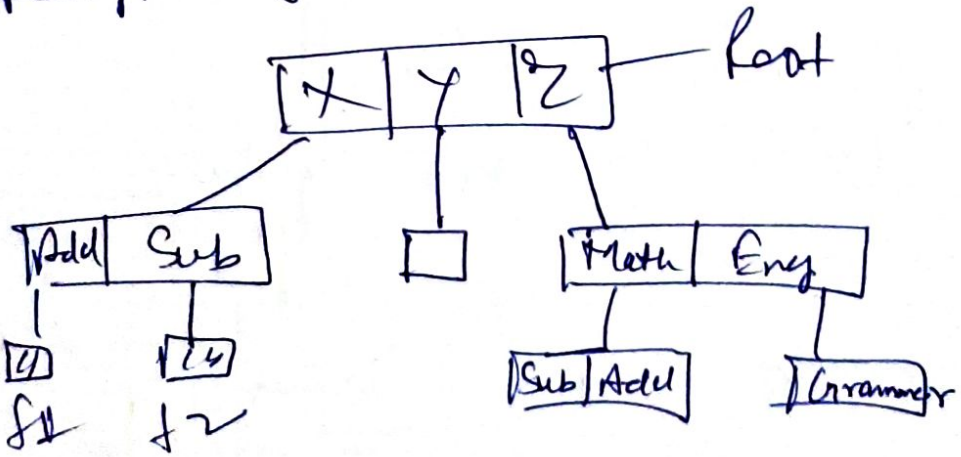


(ii) Two level Directorys → Create a directory for each user

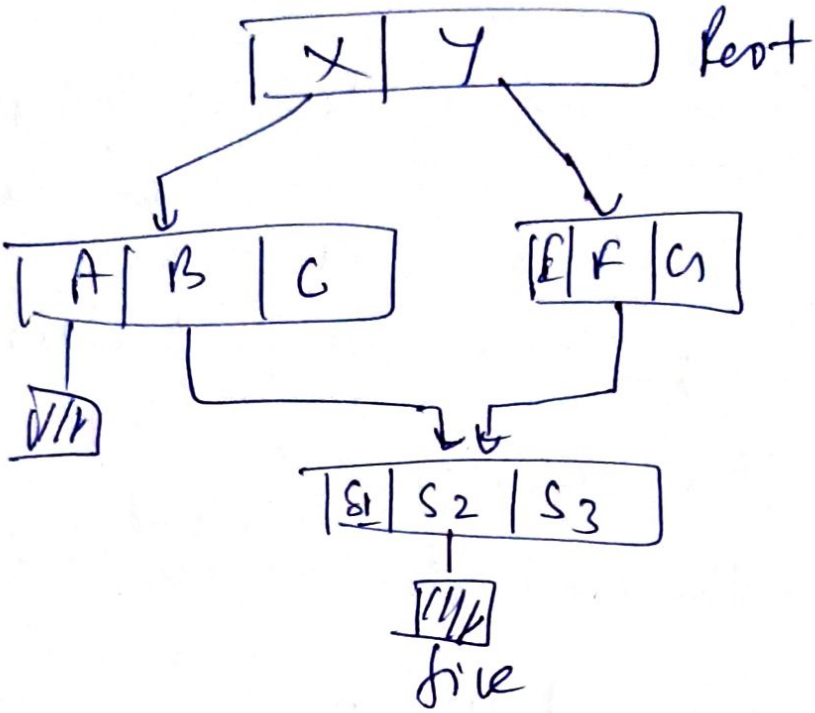


(iii) Tree-Structural Directory → Allows user to create their own sub-directories and organize their file accordingly.

Path :→ It is the Path (route) from the root through all the sub-directory to specified file.



(iv) Acyclic Graph Directories :— Useful. When the same file need to be accessed in more than one place in the directory structure [Files are shared by more than one user/Process]



Protection & Security in O.S

Goals of Protection :

→ In one Protection model, Computer Consists of a collection of objects, hardware or software.

→ Each object has a unique name and can be accessed through a well - defined set of operations.

→ It ensure that each object is accessed correctly and only by those Processes that are allowed to do so

Security Violation Categories :- (CIA)

Confi. Integrity. Availability

(1) Breach of Confidentiality
→ unauthorized reading of data

(2) Breach of Integrity
→ Unauthorized ~~destruction~~ modification of data

(3) Breach of Availability —
→ Unauthorized destruction of data

(4) Theft of Service
→ Unauthorized use of resources

(5) Denial of Service (DOS)
→ Prevention of legitimate use

Principles of Protection :-

Guiding Principle — Principle of least Privilege —

→ Programs, users and systems should be given just enough privileges to perform their tasks

→ Limits damage if entity has a bug, gets abused

→ Can be static (during life of system during life of process)

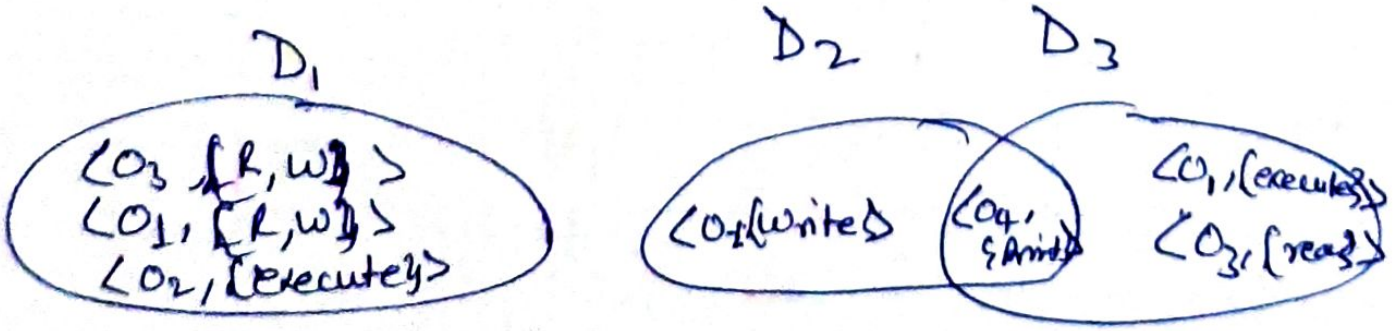
→ Or dynamic (changed by process as needed) — domain switching, privilege escalation.

→ "Need to know" a similar concept regarding access to data.

Domain Structure →

→ Access right = <object-name, rights-set> where rights-set is a subset of all valid operations that can be performed on the object

→ Domain = Set of access-rights



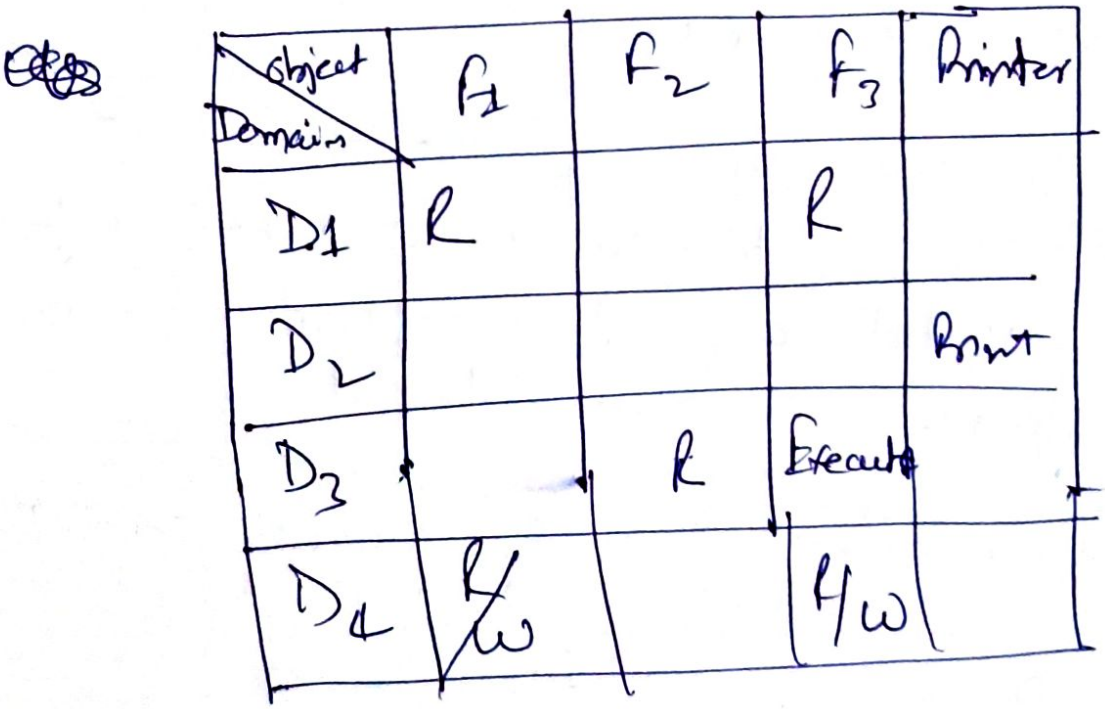
Access Matrix :-

→ View protection as a matrix (access Matrix)

→ Rows represent domains

→ Columns represent objects

→ Access (i, j) is the set of operations that a process executing in Domain$_i$ can invoke on object$_j$



object / Domain	f_1	f_2	f_3	Printer
D_1	R		R	
D_2				Print
D_3		R	Execute	
D_4	R/W		R/W	

Use of Access Matrix :- If a

→ Process in Domain D_i tries to do "op" on object O_j, then "op" must be in the access matrix.

→ User who creates object can define access column for that object

→ Can be expanded to dynamic Protection
  → Operations to add, delete access rights
  → Special access rights:
    → owner of O_i
    → Copy OP from O_i to O_j (denoted by "*")
    → Control – D_i can modify D_j access rights
    → transfer – switch from domain D_i to D_j
  → Copy and Owner applicable to an object
  → Control applicable to domain object

The Security Problem →

→ System secure if resources used and accessed as intended under all circumstances
  → Unachievable
→ Intruders (Crackers) attempt to breach security

→ Threat is Potential security violation

→ Attack is attempt to breach security

→ Attack can be accidental or malicious

→ Easier to Protect against accidental than malicious misuse

Security Violation Methods

→ Masquerading (breach authentication)

→ Pretending to be an authorized user to escalate Privileges

→ Replay Attack

→ As is or with message modification

→ Man-in-the-middle Attack

→ Intruder sits in data flow, masquerading as sender to receiver and vice versa.

→ Session Hijacking → Intercept an already-established session to bypass authentication

Levels to apply Security :

(1) Physical — Data centers, Servers, Connected terminals

(2) Human — Avoid social engineering, Phishing dumpster diving

(3) Operating System : Protection mechanisms, debugging

(4) Network : Intercepted Communications, interruption, DOS