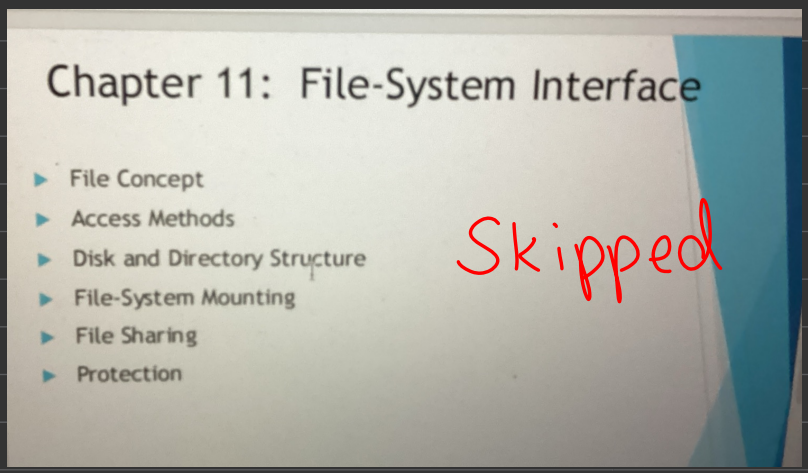
Sumedh

- Topics Covered

1. Swapping
2. Contiguous allocation / Fragmentation / Segmentation
3. Paging
4. Demand Paging (Page fault)
5. Page Replacement
6. Thrashing
7. Disk Structure / Attach / Schedule / Manage
8. RAID



- **Swapping**

A process is moved frm main mmry to temp storage (Backing store) n bought again to come running is swapping

This method allows the system to handle more processes than actually it can handle

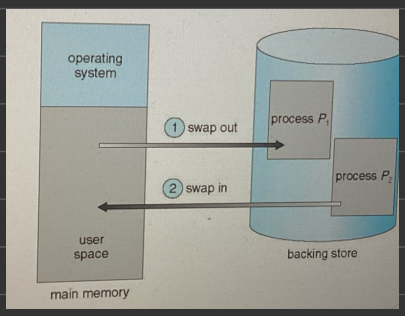
Backing Store : A storage fast enough to hold mmry copies of all process

Roll out, roll in : A algo where low priority processes kicked out to replace high ones

Transfer time : Time Taken to transfer data, depends on size of mmry

Pending I/O : If a process has pending I/O it complicates swapping

Need not necessary process comes to same add after swap



- **Context Switching with swapping**

When the process in next mmry isn't in memry than the OS swaps out one process & swaps in another

Process with pending I/O can't be swapped

Double buffering: To avoid data issue, it transfers thru kernel space

eg : A 1000 mb prcs swapped at 50 mb/s takes 2000 ms to swap out & 2000 ms to swap another

- Contiguous Allocation / Fragmentat^n / Segmentat^n

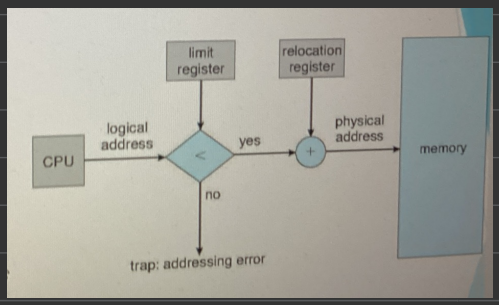
  - Contiguous Allocation

    - A basic way of allocating mmry by placing each process in a single contiguous block

    - Two - mmry sections : One for OS (low mmry)
      One for user processor (high mmry)

    - Realocation reg : A process with reallocated reg is used to interferance of procs

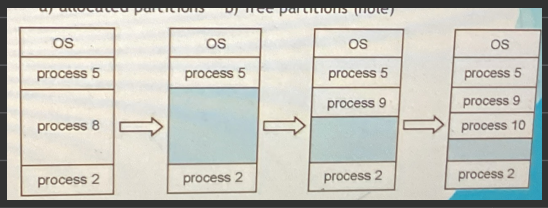
    - Base & limit reg : define allowed mmry range for each process



  - Multiple partition allocation

    Here, mmry divided into multiples partitions to support multiple process

    Mmry holes are unused spaces, filled by new processes



In order to allocate mmry efficient, there are 3 methods

1. First - fit : First available hole
2. Best - fit : Smallest hole that fits
3. Worst - fit : largest hole

- **Fragmentation**

External - Free mmry exists but isn't contiguous causing wasting of space

Internal - Allocted mmry is slightly more than needed, creating unused space

To remove ext. frag we move main mmry & create one large free block

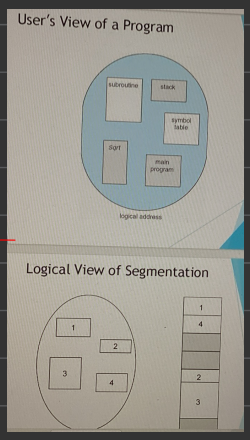
Compaction : Shuffle mmry elems to create free parts, but this can't be done dyanamically

- **Segmentation**

Organises mmry in segments based on logical divisions

Allows user to see mmry as seperate logical units such as :-

- main prog
- func
- method
- obj
- stack
- arrays



Architecture

- logical add : Contains segment no & offset
- segment table : maps logical add to physical add
  - Base - Contains the start add of prcss
  - limit - specifies length

Segment table Base reg (STBR) : points to segment table's loc

" " length reg (STLR) : indicates no. of segments

Protection bits are associated with segments
- each entry gives a bit

- Paging

Non-contiguous allocation : A process's memory can be spread across diff areas in physical mmry so it doesn't have to be continuous

Benefits : Prevent gaps
Solve issues with diff sizes

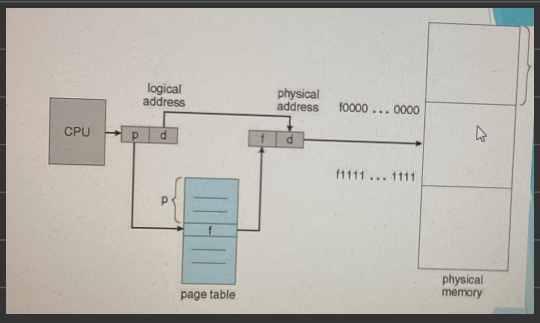
Concepts : Mmry divided into equal part called frames
Each page fits into a frame

limit : Can still lead to wastage of space (int. frag)

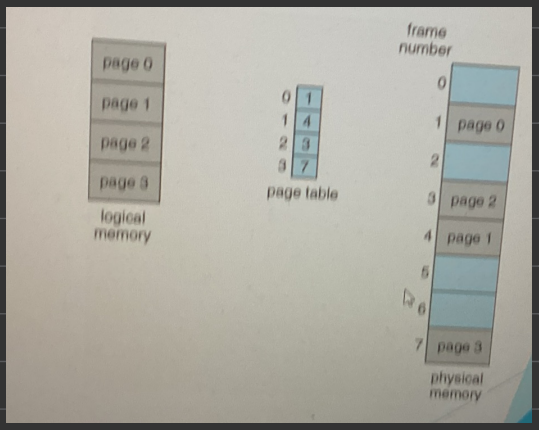
- Add translation

Page number (p) : Identifies which page to refer
Page offset (d) : Specifies exact location

Paging h/w :



Paging model :

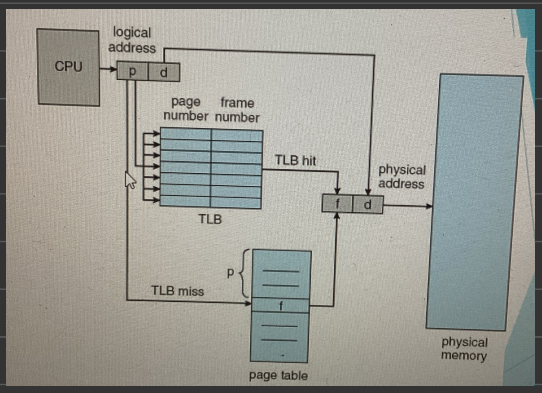


- Implementation of page table
  - Stored in main mmry
    PTBR (page table base reg) : points to table
    PTLR (length) : indicate size
  - Each access req 2 lookups, one for page & one for data
  - Solution : TLB (Translation look-aside buffer)
    Gives a fast-lookup

Features

- Some TLB store identifier to manage multiple process
- Usually small
- On a miss, entry used in future
- Replace old entries

Paging hlw with TLB :



- Effective Access time

  └ TLB takes minimal time to search but may not always find page

Formula on TLB (hit / miss)

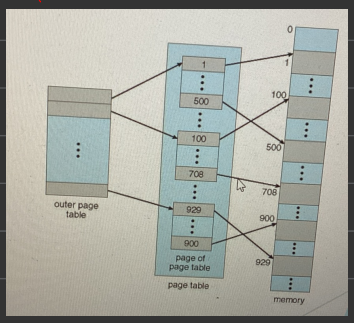
$$EAT = (1+\varepsilon)\,\alpha + (2+\varepsilon)(1-\alpha)$$

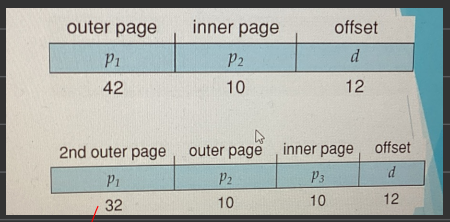
miss hit

- Mmry protection
  - Each frame has a bit indicating the process to read | writ
  - Bit may give read | write only permissions

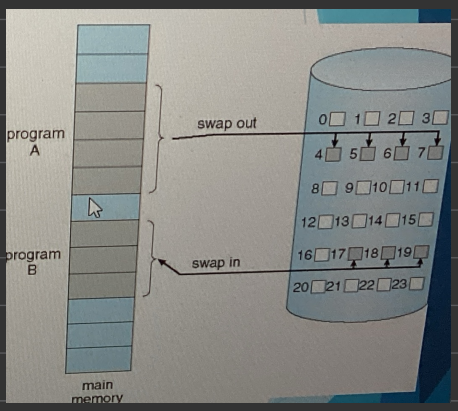
- More types of paging

1. Hierarchal Paging
2. Two-lvl paging
3. Hashed paging
4. Three-lvl paging
5. Inverted paging



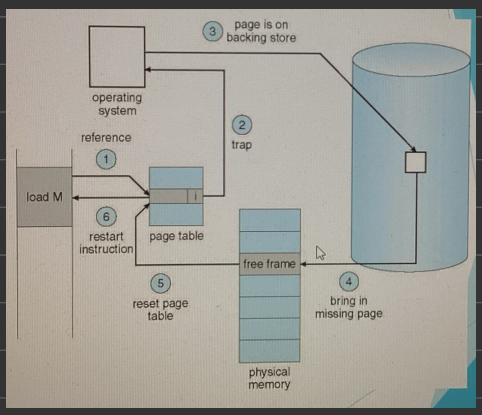


- Demand Paging (Page fault)
  - Loads only necessary pages of a prog as they're needed, rather than loading the entire prog
  - This approach saves mmry & I/O
  - Also improves response time & supports users
  - When a page isn't in mmry, the OS loads it (pager)
  - Pager predicts which pages needed & only brings those
  - If page isn't in mmry, it's load frm storage



- Page fault
  - occurs when page needed by a prog isn't in mmry

1. OS checks if it's valid
2. find a free frame
3. loads page in frame
4. Updates tables
5. Restart Instruc



Steps :
1. reference
2. trap
3. page in back store
4. missing page
5. reset page table
6. Restart instruc

- aspects of demand paging
  - Pure DP mean no page initially in mmry
  - Each instruc fetch can lead to multiple page fts
  - H/w support req for DP

- performance of DP

it more slp than wit the abov slp in hw

  - Involves 3 main activities : handle interrupt , read page & restart process
  - The page fault rate (p) how often it occurs (0 1)
  - EAT combines Time & page fault overhead

- **Page Replacement**

Purpose : Prevents excessive mmry allocation by adding a page-replacement mechanism in the system

Modify (Dirty) Bit : Helps reduce the need for data transfers

Benefit : Supports large virtual mmry by enabling more efficient use

Without PR, system will run out of mmry quickly

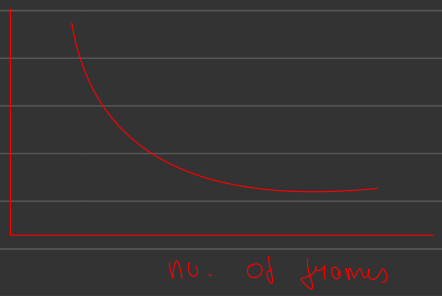
- **Page & Frame replacement algorithms**

Frame-allocation algo : determines no. of frames assigned to each prcss & which " to replace

Page-Replacement algo : Aims for lowest page-fault rate by tracking seq.

The algos are :-

1. FIFO
2. Optimal
3. LRU
4. Enhanced 2nd Chance
5. Counting
6. Page buffering

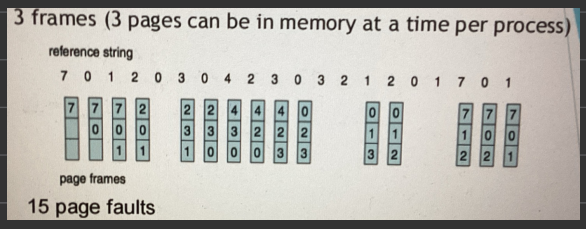


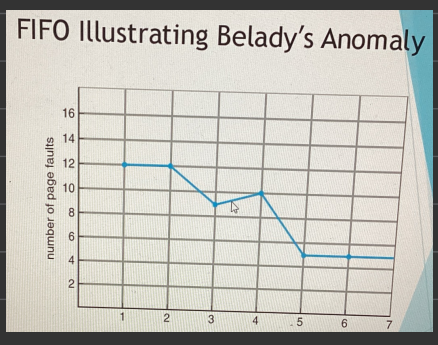
- **FIFO (First In First Out)**

Mech : Replaces the oldest loaded page

Adding more frames can cause more pge flts

It's used to track order of pges

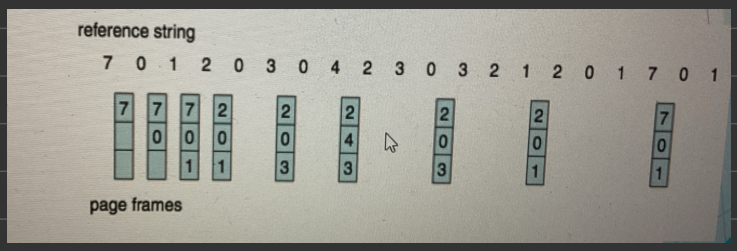




- **Optimal Algorithm**

Mech : Replaces the page that won't be used for the longest future period

limit : theoritical since relying on future



- **Last Recently Used (LRU)**

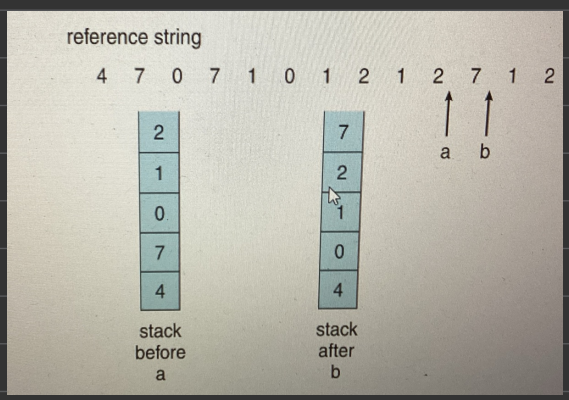
Mech : Replaces the page that has not been used for the longest time

Use : Can used counters / stack to record time

Adv : Generally, effective but complx to implement

Counter Implementation : Each page entry has a counter, incremnts the clock & record time

Stack " : Every page is refered to item in stack from Top, Top increments recording time



- **Enhance Second Chance**

Uses both the reference & modify bits to classify pages

(0, 0) : neither recently used nor modified
(0, 1) : " " " but "
(1, 0) : recently used not modified
(1, 1) : both

- **Counting Algorithm**

Least frequently used (LFU) : Replaces page with fewest responses

Most " " (MFU): " " " most "

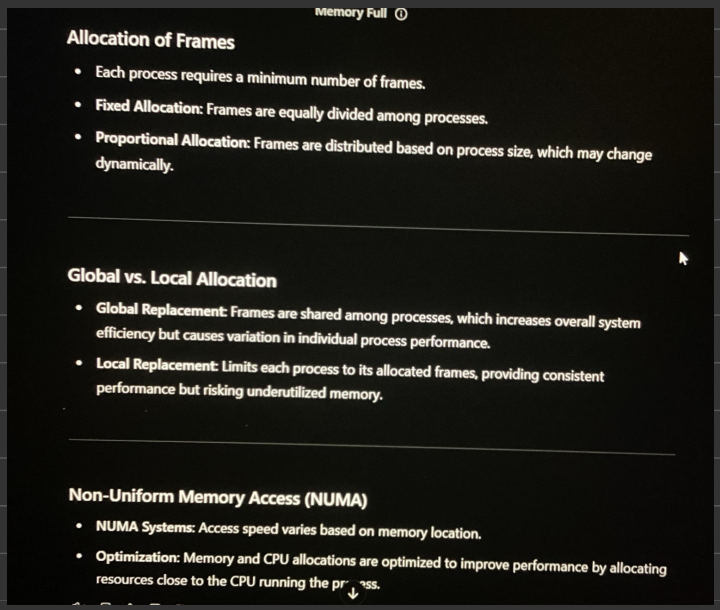
- **Page - Buffering**

Mech : Maintains a pool of free frames to speed up page replacement

Pages migh remain in free frame until they're needed again

- **Applications**

1. Guessing future pg access
2. Better knowledge
3. Save respons time
4. Bypass buffering
5. Prevent locking



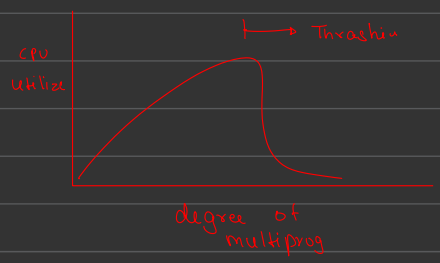
- **Thrashing**

Thrashing happens when a process doesn't have enough pages causing a high page-fault rate

It leads to constant swapping of pages in & out of mmry because pges that were just removed again are quickly needed agin

This causes :-

- Low CPU utilizatn
- OS mistakens making worse problems
- More procsses being added, inc thrashing further



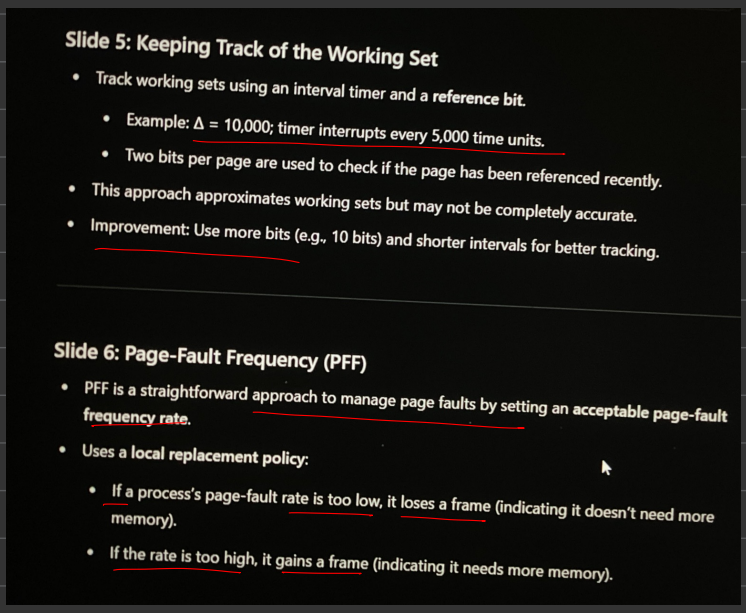
- **Demand Paging & thrashing**
  - DP works based on locality model
  - Thrashing occurs when the size of these locality models exceed
  - To reduce thrashing, we use local / priority based page replacement strategies

- **Working Set Model**
  - The " " " (Δ) is a recent set of page references eg last 10000 instruc
  - Working set (wss) of a process = Total pages accessed

if Δ too small : may not capture full set of pages
Δ too large : may include pges not currently in use
$\Delta = \infty$: It includes entire pros

D = Σ (wss) : represents total demand

* Thrashing occurs if D exceeds the available mmry frames



Slide 5: Keeping Track of the Working Set

- Track working sets using an interval timer and a **reference bit**.
  - Example: Δ = 10,000; timer interrupts every 5,000 time units.
  - Two bits per page are used to check if the page has been referenced recently.
- This approach approximates working sets but may not be completely accurate.
- Improvement: Use more bits (e.g., 10 bits) and shorter intervals for better tracking.

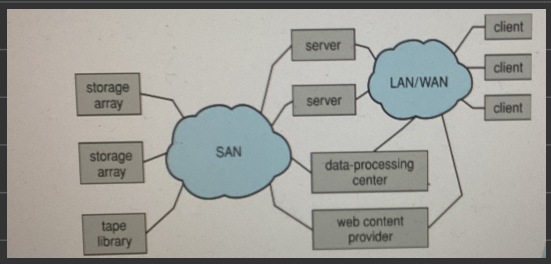
Slide 6: Page-Fault Frequency (PFF)

- PFF is a straightforward approach to manage page faults by setting an **acceptable page-fault frequency rate**.
- Uses a **local replacement policy**:
  - If a process's page-fault rate is too low, it loses a frame (indicating it doesn't need more memory).
  - If the rate is too high, it gains a frame (indicating it needs more memory).

- Disk Structure / Attachment / Scheduling / Management
- Disk Structure
  - Disk drives are organised as large 1D arrays of small storage unit logical block
  - Formatting sets up these logical blocks
  - Disk Structure : starts frm outermost cylinder & mover inward
  - Mapping logical add to phy add is easy but issue can arise with bad sectors
  - Disks may have diff no. of sectors per track

- **Disk Attachment**
  - Disks connect to comps via I/O ports & buses
  - **SCSI** : A type of bus that can connect upto 16 devices at a time
  - SCSI devices have initiaters (start tasks) & targets (perform tasks)
- **Fibre Channel** : A high speed connection used in SAN (Storage area network) to link multiple Storage devices

SAN :
- Its one /more storage arrays
- Hosts also attach to switch
- Multiple hosts " to multiple array

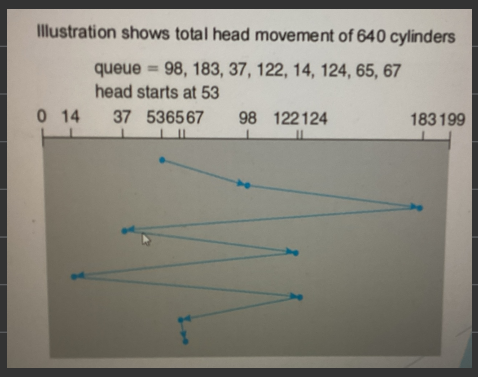


- **Disk Scheduling**
  - OS manages disk to ensure quick access & efficient use
  - Minimize seek time
  - The OS schedule disk task to minimize delays
  - Disk I/O req from : OS
    System prcs
    User prcs
  - Types of Disk scheduling Algo
    1. FCFS
    2. SSTF
    3. SCAN
    4. C-SCAN
    5. C-LOOK
- Diff algo are used to decide best order

1. FCFS

(First come, first Serve)

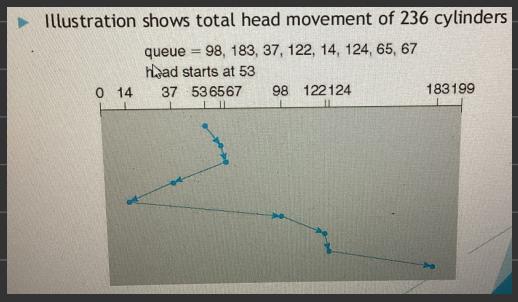
Processes req. in order, but res in unnecessary head movement



2. SSTF

(Shortest seek time first)

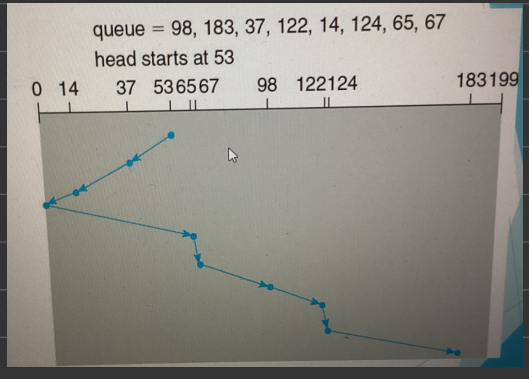
- Chooses the closest req to minimize head, movement
- Some req. may wait longer
- Comonly used n simple



3. SCAN

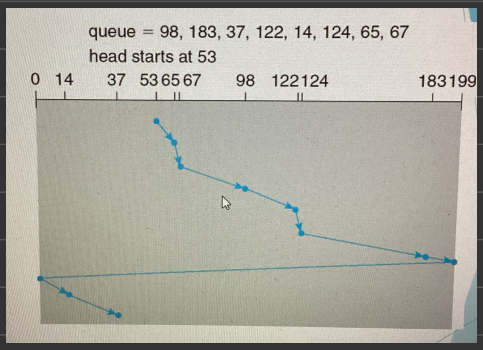
(Elevator Algo)

- The head moves accross the disk
- Serving req in one direction then reverse
- Used in high-demand disk



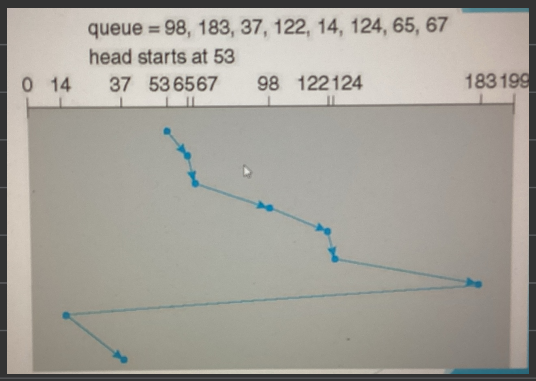
4. C-SCAN

- Moves in one direction
- Once reached the end, returns w/o servicing req
- Provides fair WT
- Used in high-demand disk



5. C-LOOK

- Like LOOK a version of SCAN, C-LOOK version of C-SCAN
- Go to the last req. in each direction



- **Disk Management**
  - low-level formatting prepares the disk by dividing it into sectors, store units
  - **Partioning** organise the disk into section
  - **logical formatting** creates a file system structure
  - **RAW disk** access allows apps like db to manage their own disk

- **Raid Structure**

  - RAID (Redundant Array of inexpensive disk) is a system that uses multiple disks together

  - To improve reliablity & data rebundancy

  - Using multiple disks means longer time before a failure (mean time to failure)

  - Mean time to repair (MTTR) : It's a time taken to fixed a failed disk

  - During this time, another failure could risk data

  - Mean time to data loss : It depends on foll factors
    - If disk independant
    - long mean time to failure

  - disk striping : Split data across multiple disks, treating them as a single unit

  - Raid lvls : Raid has six lvls with different ways of organizing data for speed

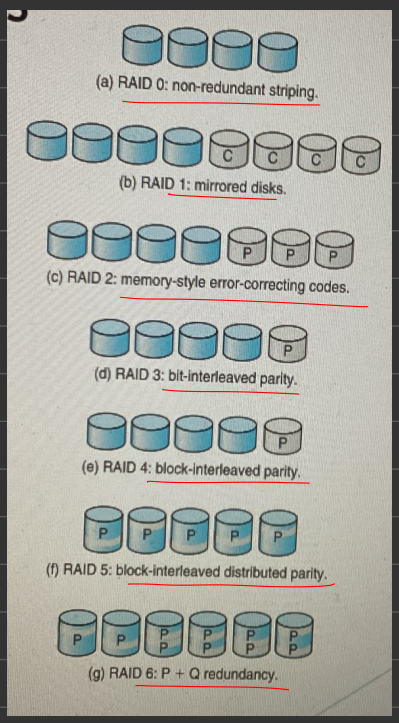
  - Redundant data storage : This setup improves performance & safety

    1. Mirror /shadowing (RAID 1) : Keep exact copy of each disk
    2. Stripped Mirror (RAID 1+0) : Combine speed & reliability
    3. Block interleaved (RAID 4, 5, 6) : Uses fewer backups

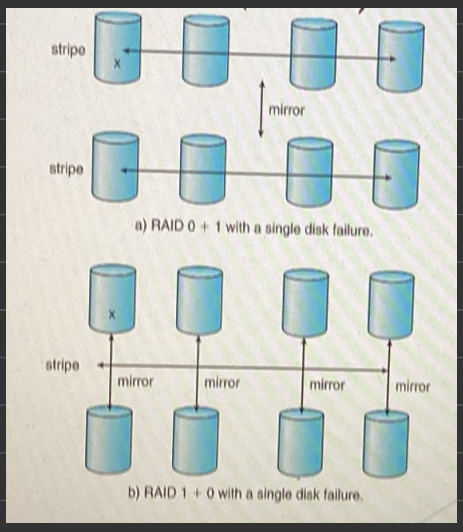
  - RAID can still fail if storage array fails, so data replication is common

- RAID lvls

RAID 1
2
3
4
5
6



- Raid (0+1) & (1+0)



- Other features

1. Snapshot : Capture the state of file at a specific time, helpful for backup

2. Replication : Automatically duplicates data to other locations for backup

3. Hot Spare Disk : An used disk that automatically takes over if another fails