Unit-5 {File System}

Date

Chapter-10 → File System

- It has two distinct parts

1. Collection of file and storing of data
2. organizing the info about file in systems {Directory structure}

- Files are mapped from O.S onto physical devices.
  {book} {Library} {Shelf}

So, basically mapping determines, where the file is stored on ~~Operating System~~. the disk.

- from a user perspective, it's a smallest allotment of logical secondary storage {∵ data can't be written to secondary storage device unless they are written in a file}

e.g:- textfile, video-file, executable file, source-file.

- File attributes

1. Name → file name in Human readable format.
2. Identifier → a unique number that identifies file within System {Non-human readable ~~format~~ name}
3. Type
4. Location → This inf is pointer {reference address} to a device.
5. Size
6. Protection
7. Time date and user modification

Note: Once a name is given to file, it become independent of the process, user and system!

e.g:- Image1.png → whether you do editing, send this Image, or copy across network it still ~~become~~ called same as Image1.png.

Spiral

Date

- Info related to file is stored in directory structure {like name, Identifier} {after, this will point other attributes} so, when file is accessed this info is also brought into memory peacemeal.
  ↳ {Immediately}

• File Operations

- There are 6 minimal operations, which are then combined to perform other operations.

1. Creating a file : Two steps necessary
   → whether space in file system is found.
   → entry for new file must be made in directory.

2. Writing a file : To write a file we do a system call, specifying name of file and what to be written in file. Then a system assigns write pointer, which must update whenever write occurs

3. Reading a file : again system call, specifying name and memory address of file, then O.S search in directory and a read pointer is maintained {to keep track of current position of file}

{∵ process is either reading or writing, so both operation maintain same pointer Pre-Process Current position file pointer to save space and time}

4. Repositioning within a file or Seek : It refers to moving current position file pointer to a specific location; without doing actual I/O operation.

5. Deleting a file : Removes the directory entry {or reference} and marks the space as available

6. Truncating a file : Erase the contents of a file, but keep its attributes

Spiral

ind → Independent

Date

• File operation and Tables

1. Open() and close() system calls :
   - open() → Called to access file, it prevents constant directory searching by creating entry in open-file table for quick reference to file's details.
   - close() → Removes the file from table once it's no longer in use. (create() and delete() work only with closed files

2. File tables :
   - Pre-process table : Tracks files opened by a specific process {includes file pointers, access rights etc}.
   - System-wide table : Contains process ind. information {such as file size and location on disk}

3. What happens in case of simultaneous file action ?
   - If multiple process opens same file, entries in pre process table points to a shared system wide table entry.
   - Open Count : It tracks how many processes have open the file, and when one process close file it reduce count by -1, and when reaches 0, file entry is removed.

4. Key elements store for open files :
   - file pointer : Track where the process last read or write
   - file-open count : ↑ above defn →
   - Disk-location : Stored to avoid repeated disk access.
   - Access-rights : ensure process can only perform allowed operations {In memory}

Spiral

Date

- **File locking** : • It prevents multiple processes from accessing or modifying file simultaneously
  • Useful in scenarios, where several process share files {e.g. System logs files}

→ Types of file locks:

- Shared Lock: Multiple processes can access file concurrently (reader lock)
- Exclusive Lock: Only one process can access the file at a time. (writer lock)

→ Locking Mechanism

- Mandatory locking: O.S enforces the lock. If one process locks a file, others are blocked from accessing it until lock is released {e.g. Windows}
- Advisory Locking: O.S doesn't enforce the lock. It's up to software to manually request the lock before accessing file {like text editor} {e.g. In UNIX}

→ Precautions with file lock

- With Mandatory locking, process holding exclusive locks, should release them quickly to avoid blocking other processes
- Measures must be taken to prevent two processes from waiting indefinitely for each other's lock, which could lead to → "Deadlock"

Spiral

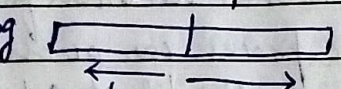
• fixed length logical records: i.e, each record is of uniform size {If data not fill space padding is done} and have logical arrangement

Date

Access Methods

1/2) Sequential Access:

• It's the simplest method, in which file info is processed in order, one record after the other



(rewind) ← → read_next(): reads next portion of file and automatically update a file pointer or

write_next(): appends to end of file and advances to new end of file.

2/2) Direct Access or relative Access:

• Here, file is made up of fixed length logical records, that allow programs to read and write ~~programs~~ records rapidly in no particular order.

• Since, file is viewed as a sequence of numbered blocks/records so you can randomly access any block, like block: 7, 53, 82 etc

• files are access using relative block numbers {start from 0}, this makes file access independent of physical disk locations

↓

like, if a logical record is of length ~~(L)~~ L {bytes}, and you wanted to access N^{th} {record}, so system calculate it's position as {L * N} bytes from file's start.
so we directly access that record.

• Operations: read(n), write(n),
position_file(n)
↳ put pointer to n^{th} block number

Spiral

RAID → redundancy array of Ind. disk

Directory and User Structure

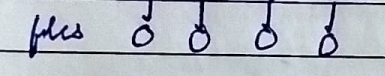
Date

- Any entity containing a file system known as **Volume**.
- Volume also contain info about the files in the system
  - It may be Single device, Multiple device linked in RAID
  - ↳ This info in kept in entries in **Device Directory**
    - It records information for all files on that Volume (eg:- name, size, type, location etc)

→ following operation on Directory:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

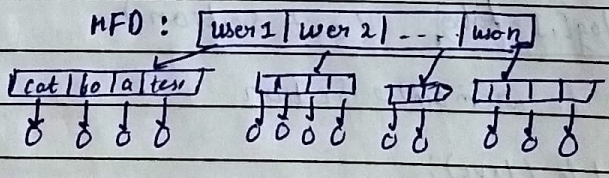
(a) **Single level** Directory: → • All files stored in one directory



- High potential for name conflicts
- All users share one space

(b) **Two level** directory: → • It assigns each user a separate User file Directory (UFD)

These UFD's are indexed by Master file Directory (MFD), which is organised by username or account numbers



- ∵ each user has its own UFD, preventing name collision across user.
- facilitates file management for each user independently
- But, restricts inter file sharing! {require mechanism such as search paths)
  - user b/ text.abc
- System files are stored in special directory called 'User 0'.
- A sequence of directory that O.S follows when locating files called **search path** {local UFD → System directory}

Spiral

Date

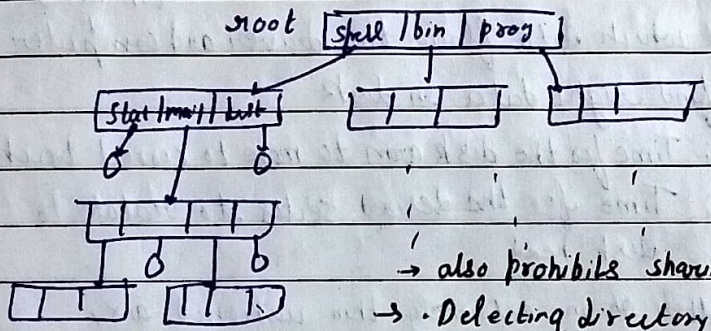
(c) Tree Structured directory : extension of above, that allows a hierarchy of directories and sub-directories.

• Tree has root directory and every file in a system has unique path name.

• Directory are treated as special file, distinguish by a bit in entry (0 = file, 1 = directory or subdirectory)

Abs. path → Starts from a root and follows a path down to specified files

relative path → Start from current directory



→ also prohibits sharing of file or directory

→ Deleting directory with files requires careful policies (e.g. recursive deletion)

↳ rm -r

(d) Acyclic Graph directory : It allows file or directories to exist in multiple locations, enabling sharing, while avoiding cycles in directory structure

[It's not keeping duplicating copies of files or dir to each user]

• shared directory ensure consistency, as all users access the same file.

• A file may now have multiple absolute path names due to sharing

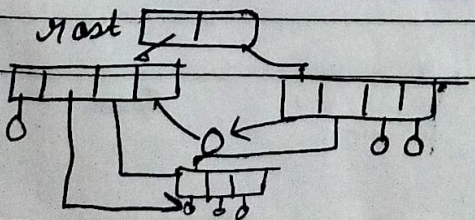
• Link Implementation → It's a pointer to another file or directory

Symbolic → Hard link

(contain path to original file) [Directly open to the file data blocks]

• Deletion → • Immediate deletion of file (but leaves pointers)

→ • Preserve file until all references are deleted



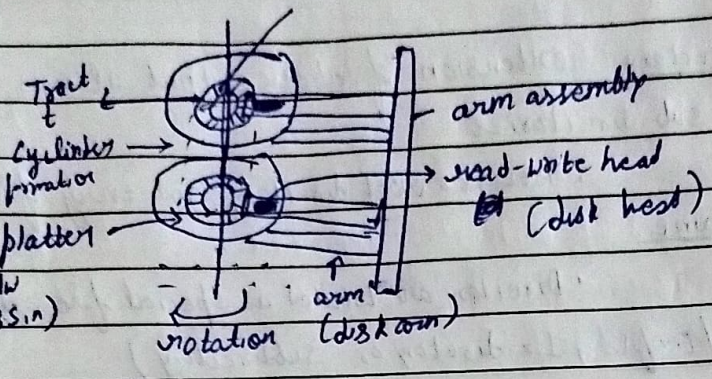
(Acyclic graph structure

Spiral

Mass Storage Structure

Date

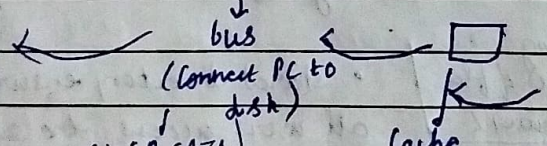
(i) Magnetic disk:



both surface are covered with magnetic material (Store inf. by recording it magnetically on platters)

- Transfer rate : rate at which data flows blw driver and computer
- Positioning time (or Access time) : Time to find right data on disk
  - Seek time : Time for the disk arm to move to correct track
  - Rotational Latency : Time for the desired sector to rotate to disk head (in RPM)

∵ disk head flies over platter on an extremely thin cushioned air, sometimes it will make contact with disk surface. This accident is called head crash. (∵ hard disk can't be repaired, ∵ It is replaced)



(ii) SSD → fast, consume less power, no moving parts, but less capacity than large disk

(iii) Magnetic Tape → hold large data, but access time is very slow.

Spiral

10.4 Date

- bandwidth: Total byte transferred / (Time of first request for service − Completion of last transfer) or Total Time

Goal → minimize access time and increase bandwidth

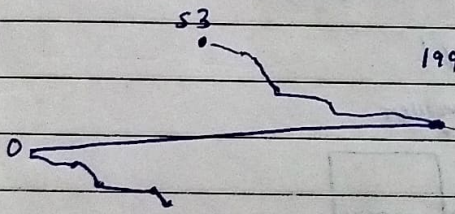
(a) FCFS Scheduling: • first come-first served • generally slow.

(b) SSTF " : • Choose request closest to current head position
(Shortest seek time first) • leads to starvation, reduces head time, but not optimal.

(c) SCAN Scheduling: • from head start → move to first end (0 or 199) → then move to other end, while servicing request come in way
(elevator algo) (mentioned left or right) while servicing request in a way

C-SCAN

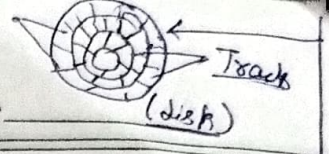
(d) ~~CSCAN~~ Scheduling: • from head start → move to one end while servicing any request in way → Immediately come to other end (No service) → then move other end, servicing request



(e) C-Look: → Same → only move end up to final request is available (i.e. 183) → Imd movement only up to final request (183) → the move other end

Optimal generally → CLOOK and SSTF
Heavy load → CLOOK
Moderate load → SSTF

Spiral



Date

• Disk Scheduling Algorithm

It's an arm (It has Rlw head)

There are diff disk-scheduling algo :-

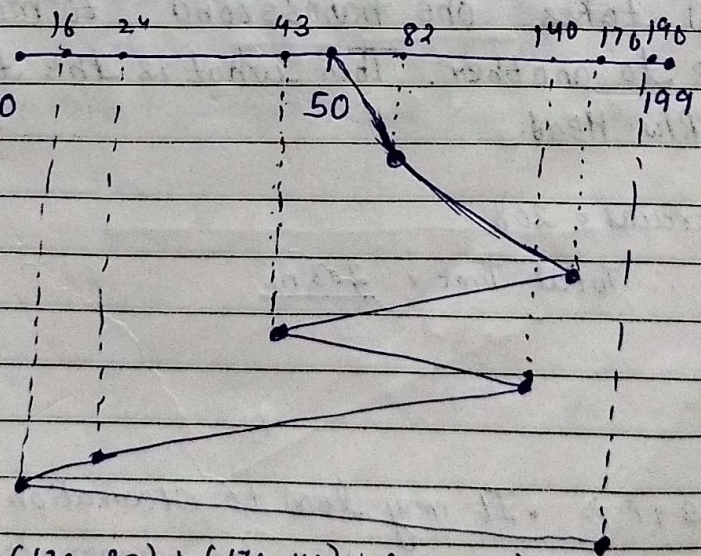
(a) FCFS
(b) SSTF (Shortest Seek Time first)
(c) CSCAN (Circular Scan)
(d) SCAN
(e) LOOK
(f) CLOOK (Circular Lock)

- The goal is to minimize the seek Time.
- Seek Time → Time taken to reach up to the desired track

(a) FCFS

Q A disk contains two hundred tracks (0 to 199).
Request queue contains the following track numbers:
82, 170, 43, 140, 24, 16, 190
Current position of Rlw head is 50, calculate Total number of track movements made by Rlw head.

Sol



⇒ (82-50) + (170-82) + (170-43) + (140-43) + (140-24) + (24-16) + (190-16)

or

(170-50) + (170-43) + (140-43) + (140-16) + (190-16)

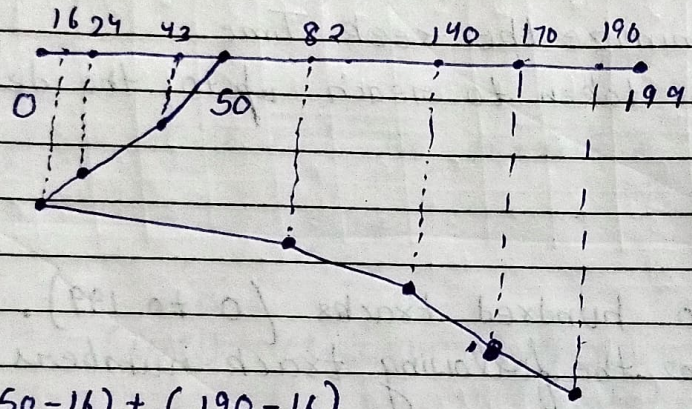
⇒ 642

(Total no of Spiral track movement made by Rlw

Date

- adv of FCFS → There is no Starvation.
- disadv of FCFS → Take more seek Time

(b) SSTF



⇒ (50-16) + (190-16)

⇒ 208 {Total Tracks}

Q If R/W. Head takes one nanosecond to move from one track to another. Then what is the total time taken by R/W. Head.

Soln ∴ Total tracks = 208

∴ Total Time = 208 ns

- Disadv of SSTF ⇒ • It may lead to starvation.
  - Overhead is also generated, to find the movement of R/W Head onto the next track. {Calc. which is nearest}

Spiral

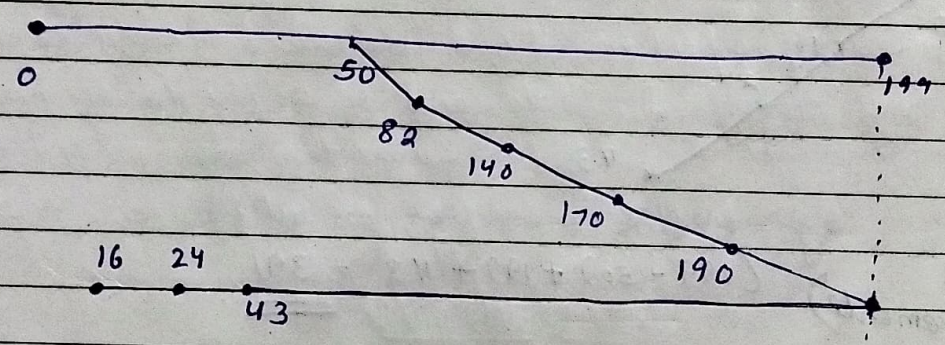
Date

adv → In most of the cases, it leads to optimal result.

(c) SCAN Algo

- It is also known as the elevator algorithm.
- Direction will always be given in this.

- Direction: Move to Larger Side



Total track movements: 332
(190-50) + (199-16)

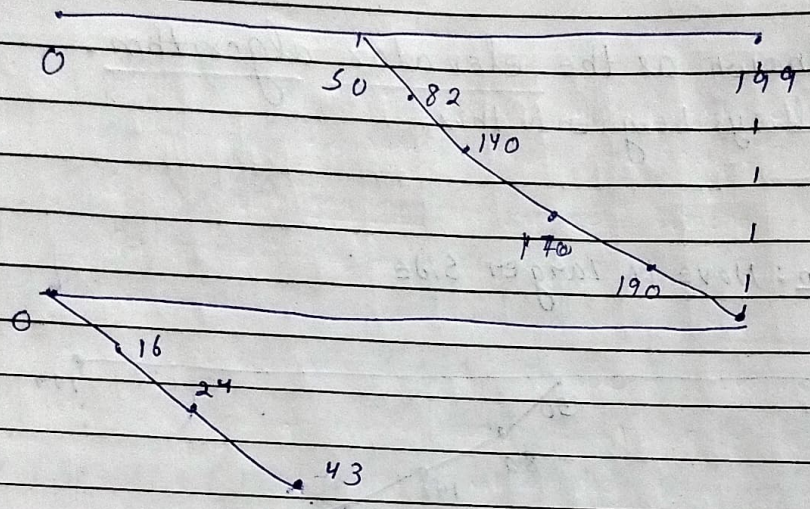
{We don't go to zero, but will go to end}

- Direction: Move to Smaller Side

Date

(d) C-SCAN Algo
(Circular Scan)

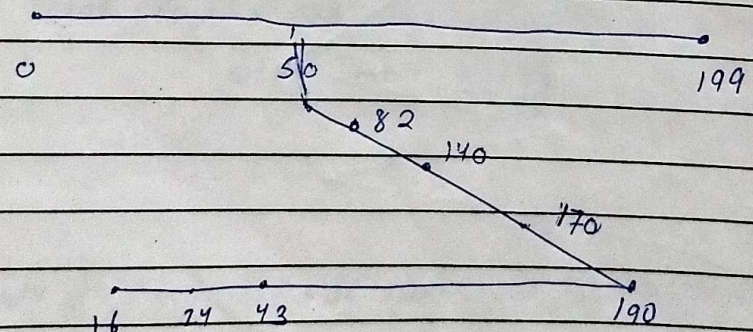
- Move towards larger side



(Track movements) $(199 - 50) + 199 + 43 = 391$

(e) LOOK Algo

- Move towards larger side



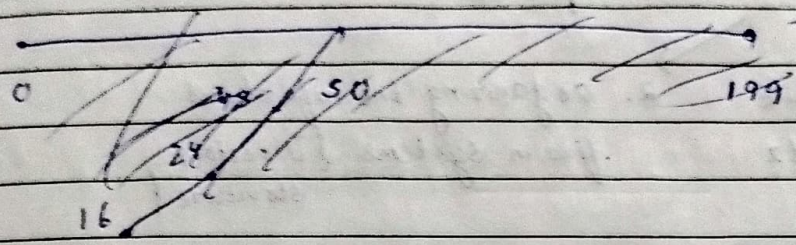
$(190 - 50) + (190 - 16)$

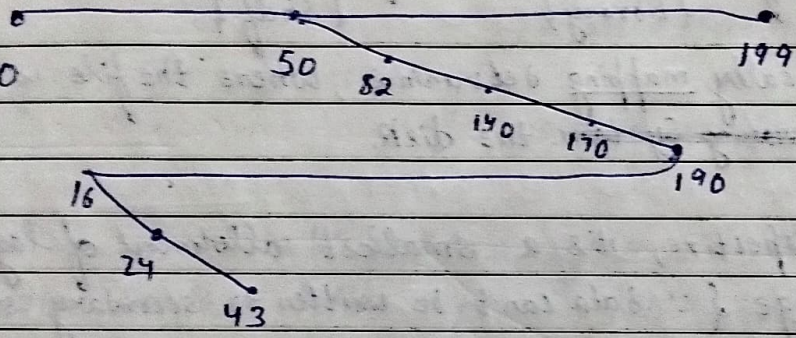
$\Rightarrow 314$

Spiral

Date _____

f) CLOOK (Move towards larger side)





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

i) 341