# COMPUTER SCIENCE

Operating System

## File System & Device Management

Logical structure of disk

Lecture no:02


Logical structure of disk

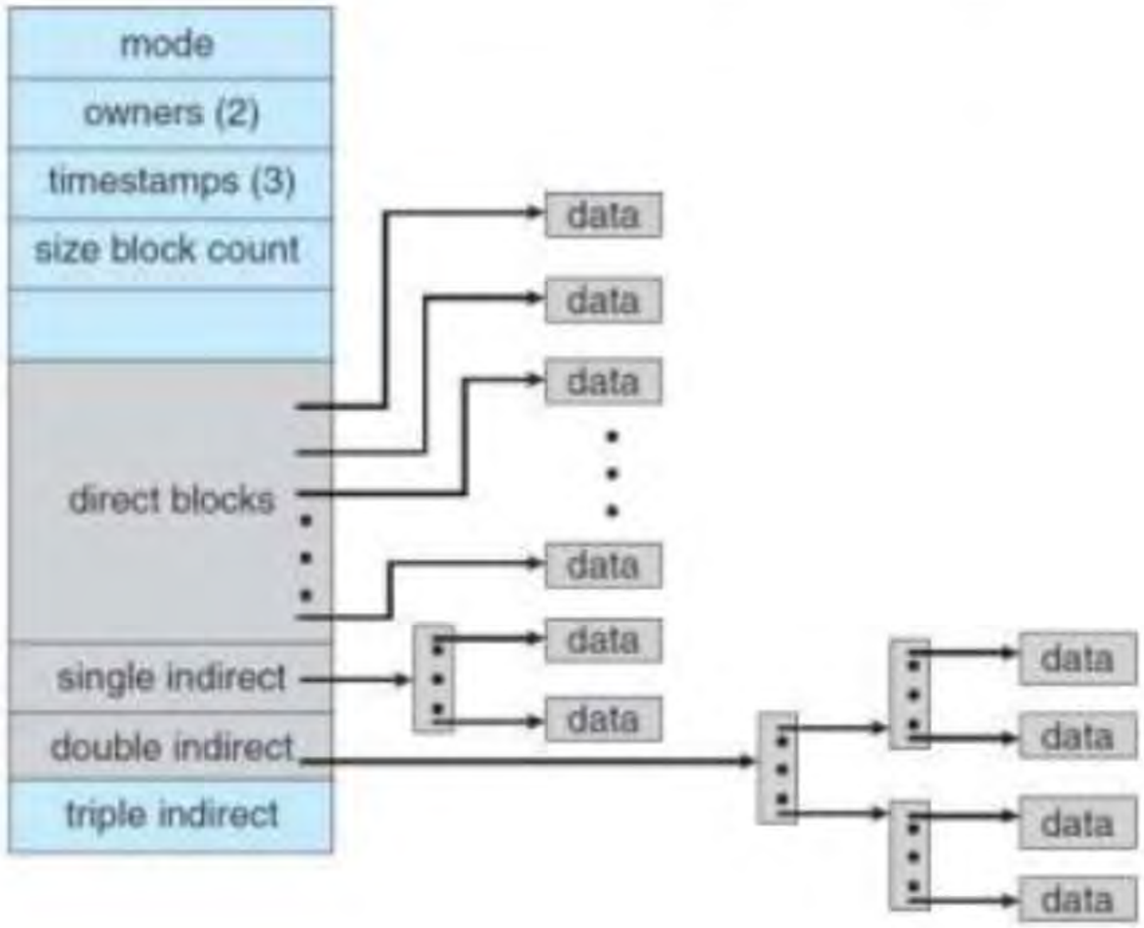
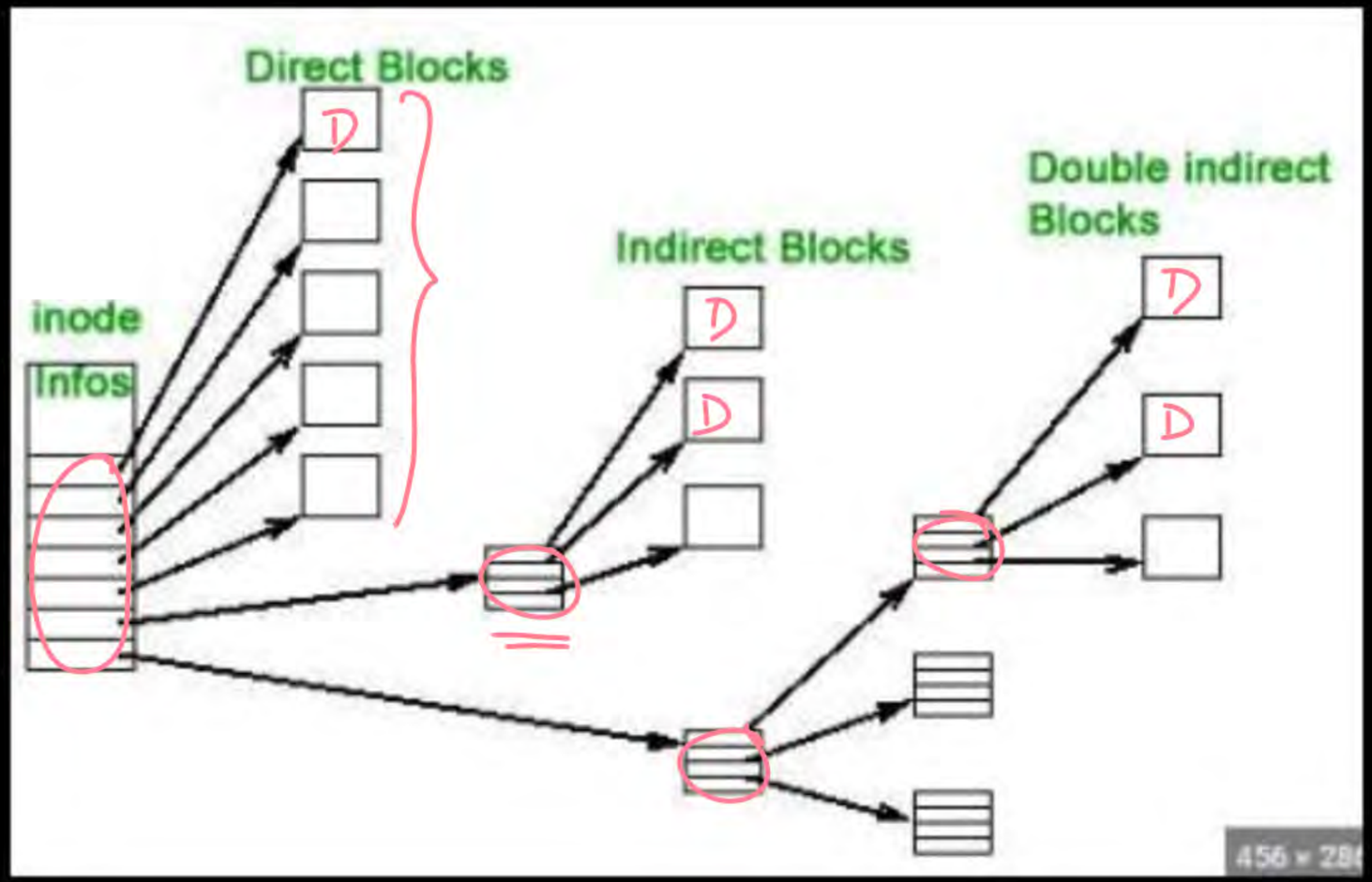



**Figure 12.9** The UNIX inode.

# Multi-Level Indexed


Direct Blocks
Indirect Blocks
Double indirect Blocks
inode
Infos
D
D
D
D
D

# II: DOS/WINDOWS Impl

: Tabular linked Alloc.

1000 . . 0 . . 999

| Gen Attributes | | |
|---|---|---|
| F. Name | - - - - - | First DBA |
| KRK | | x |

## Dir. Structure

<x, P, 3, t>

F.A.T <M.F.T>

FAT entry

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | < t > |
| | : |
| x | < P > |
| y | |
| | : |
| t | -1 |
| P | < 3 > |
| | : |
| n | |

→ FAT is orgnizd as a set of entries

→ No. of entries in FAT = No. of Blocks of disk

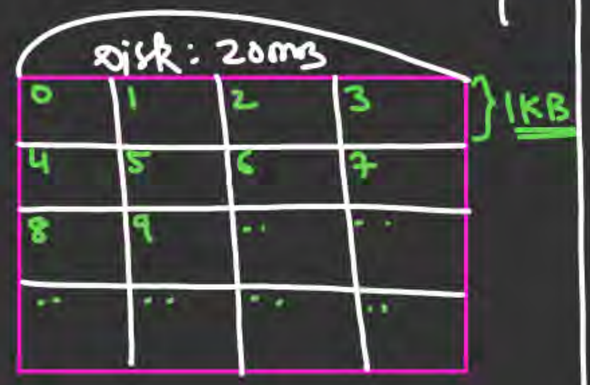
→ FAT entry contains address (DBA) of next Disk (Data) Blk in use by File;

→ FAT entry Size ~ DBA

# Disk Free Space Management

Disk Size : 20MB ;

DBS : 1KB
DBA : 16 bits ✓




No. of Blks = $\frac{20MB}{1KB}$ = {20K} ✓

DBA : 15 bits

Given disk size ≤ Max Possible disk size
20MB ≤ 64MB

Max. Possible disk size = 64MB

① Free linked list :


② Free list :


How many Blks are needed to Store 20K free DBA's

1 Blk (1KB) = 512 free DBA's

? ⟵ 20K "

$\frac{20K}{512} = 40$ Blocks

③ Bit Map | vector :

B → B.Bit < 0 → free, 1 → in use

: 1101001011 1
0101110001 1
⋮ ⋮ ⋯

} 20K bits

DBS : 1KB = 8K bits

Bit Map Size = 20K bits

1 Blk —— 8K bits

? ← —— 20K bits

$\frac{20K}{8K} = 2.5 \sim 3$ Blks

Bit Map : (Search Time)

All Blocks

<Free + in use>

Free list ✓

<Free Blks> (Faster for Alloc)

## 4) Counter Method: Majority of free Blks are contiguous

| First free DBA | Count of free Blks |
|---|---|
| 25 | 30 |
| 150 | 250 |
| 650 | 20 |
| 1000 | 180 |

File: FA : 15

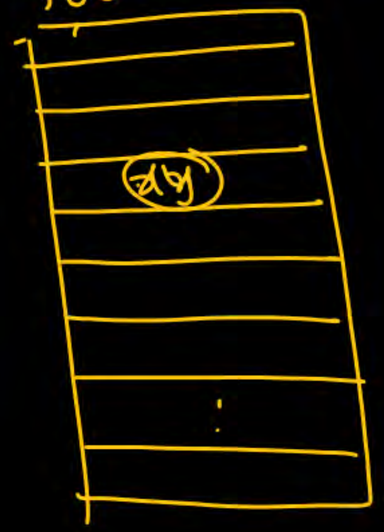
1. First Fit
2. Best Fit
3. Worst Fit

**Q.5** Consider a linear list based directory implementation in a file system. Each directory is a list of nodes, where each node contains the file name along with the file metadata, such as the list of pointers to the data blocks. Consider a given directory foo.

MSQ Which of the following operations will necessarily require a full scan of foo for successful completion?

(BC)

foo



fnum

(A) Opening of an existing file in foo ✗

(B) Creation of a new file in foo ✓

(C) Renaming of an existing file in foo ✓

(D) Deletion of an existing file from foo ✗

**Q.7** Consider a Unix I-node structure that has 8 direct Disk Block Addresses and 3 Indirect Disk Block Addresses, namely Single, Double & Triple.

Disk Block Size is 1Kbytes & each Block can hold 128 Disk Block Addresses.

Calculate

(i) Maximum File Size with this I-Node Structure?

(ii) Size of Disk Block Address?

(iii) Is this File Size possible over the given Disk?

DBA = $\frac{1KB}{128}$ = $\frac{2^{10}}{2^7} = 2^3 = 8B$

128

1KB

Direct: $8 * 1KB = 8KB$

S.I : $128 \times 1KB = 128KB$

D.I : $128 \times 128 \times 1KB = 2^{24} = 16MB$

T.I : $128 \times 128 \times 128 \times 1KB = 2^{21} \cdot 2^{10} = 2^{31} = 2GB$

0.136MB = 136KB

2.016GB

8B

0.016GB <= 16.136MB

Yes

Give(max) disk size = $2^{64} \times 2^{10} B = 2^{74} \cong$ 2GB

**Q.9** The index node (inode) of a Unix-like file system has 12 direct, one single-indirect and one double-indirect pointers. The disk block size is 4 kB, and the disk block address is 32- bits long. The maximum possible file size is 4 GB (rounded off to 1 decimal place)

DBS = 4KB; DBA = 32 bits = 4B

Direct : 12 × 4KB = 48KB = 0.048MB

S.I : 1K * 4KB = 4MB = 4.048MB

D.I : 1K * 1K * 4KB = 4GB

~ 4GB ✓

4KB = 1K


2

~~1KB~~

~~512~~

DBS = 1KB

DBA = ~~16 bits, 2B~~

512

**Q.10** A File System with 300 G Byte Disk uses a File descriptor with 8 Direct Block Addresses, 1 Indirect Block Address and 1 Doubly Indirect Block Address. The size of each Disk Block is 128 Bytes and the size of each Disk Block Address is 8 Bytes. The maximum possible File Size in this file System is

DBS = 128B; DBA = 8B;


$N = \frac{128}{8} = 16$

File Size: $8 \times 128B + 16 \times 128B + 16 \times 16 \times 128B$

$: 1KB + 2KB + 32KB = 35KB$

(A) 3 K Bytes

(B) 35 K Bytes ✓

(C) 280 K Bytes

(D) Dependent on the size of the disk

**Q.14** A File System with a One-level Directory structure is implemented on a disk with Disk Block Size of 4 Kbytes. The disk is used as follows:

4

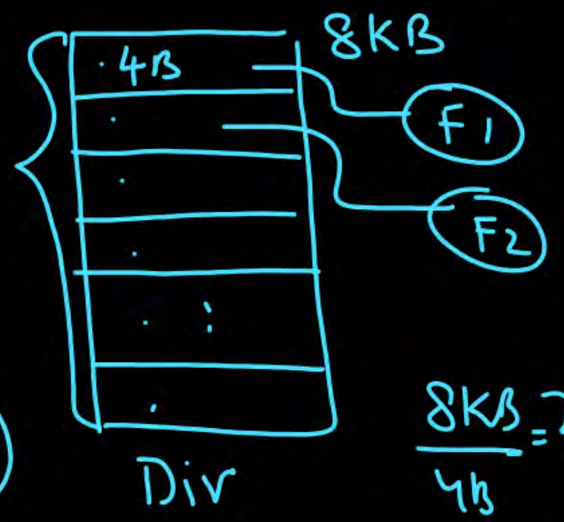
- Disk Block 0 : Boot Control Block
- Disk Block 1 : File Allocation Table, consisting of one 10-bit entry per Data Block, representing the Data Block Address of the next Data Block in the files.

DBA = 10 bits

- Disk Block 2, 3 : Directory with 32-bit entry per File.

1020

- Disk block 4 : Data block 1.
- Disk Block 5 : Data Block 2,3 etc;




$\frac{8KB}{4B} = 2K$

(a) What is the Maximum possible number of Files? 2K

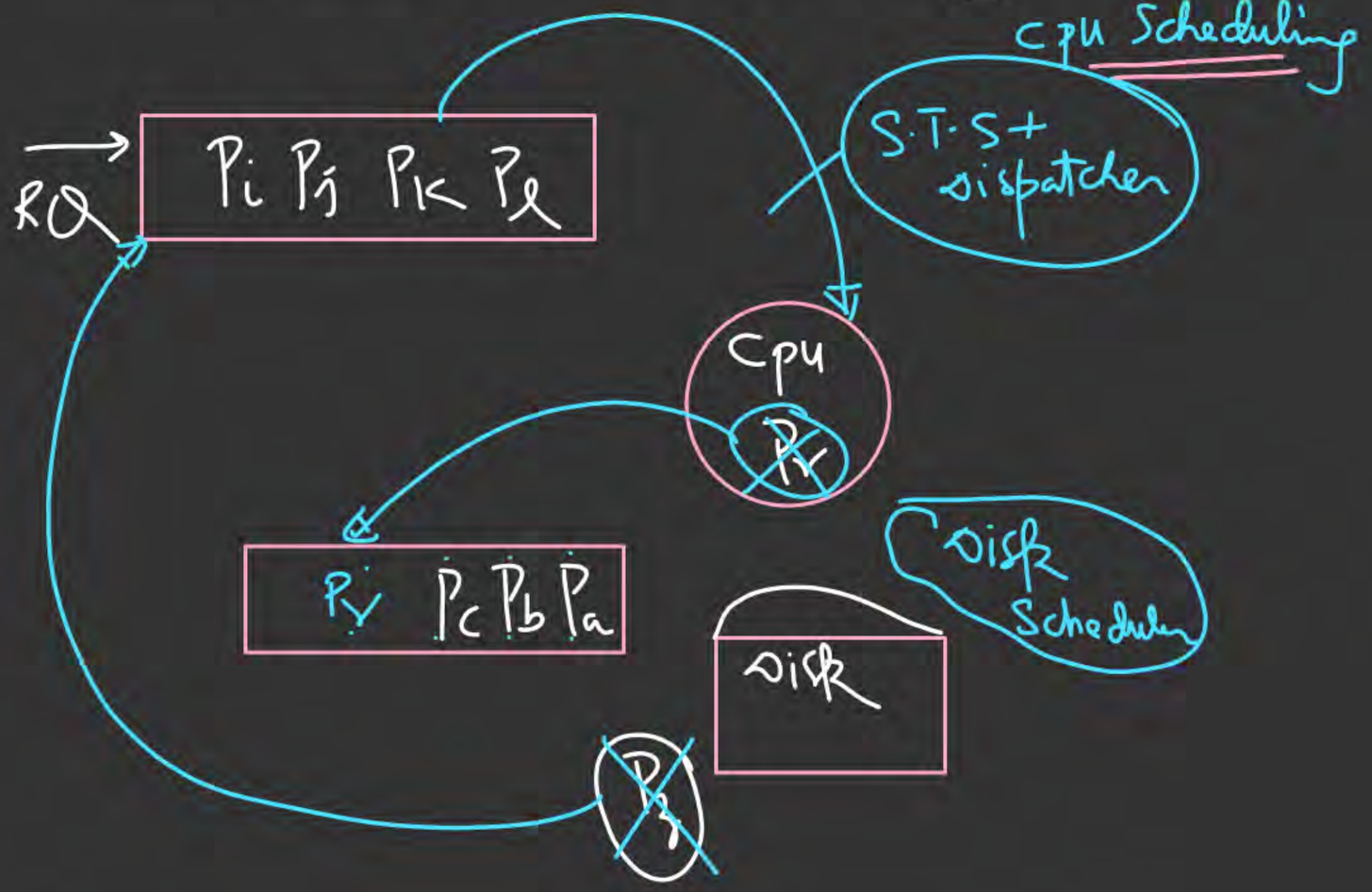
(b) What is the Maximum Possible File size in Bytes?

DBA = 10 ⇒ $2^{10} = 1024$

DBS = 4KB ⇒ $1020 \times 4KB = 4080KB$

# (I/O) Disk (Device) Scheduling:

## Need for Disk Scheduling:




Objective/Goal of Disk Scheduler

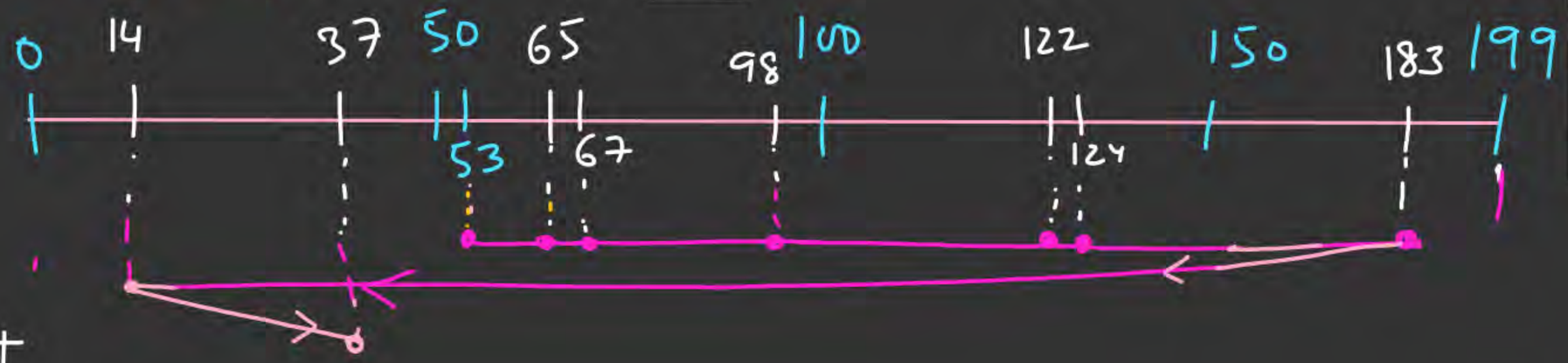
↓

Min. # of Seeks

(S.T)

Process Req's: 98 ; 183 ; 37 ; 122 ; 14 ; 124 ; 65 ; 67 <Track No's>
<FIFO>

Higher




## ① FCFS:

Total # of Seeks: 640 Seeks

## ② Shortest Seek Time First (SSTF)

closest Track Next

# of Seeks: (183-14) + (67-14) + (67-53)
: 236

## ③ SCAN

Elevator

: (199-53) + (199-14)
: 331

## ④ Look:

: (183-53) + (183-14) : 299 ✓

## ⑤ C-SCAN

: (199-53) + (199-0) + (37-0)
: 382

## ⑥ C-Look:

: (183-53) + (183-14) + (37-14) : 322

# THANK YOU!