CS & IT ENGINEERING

Operating System

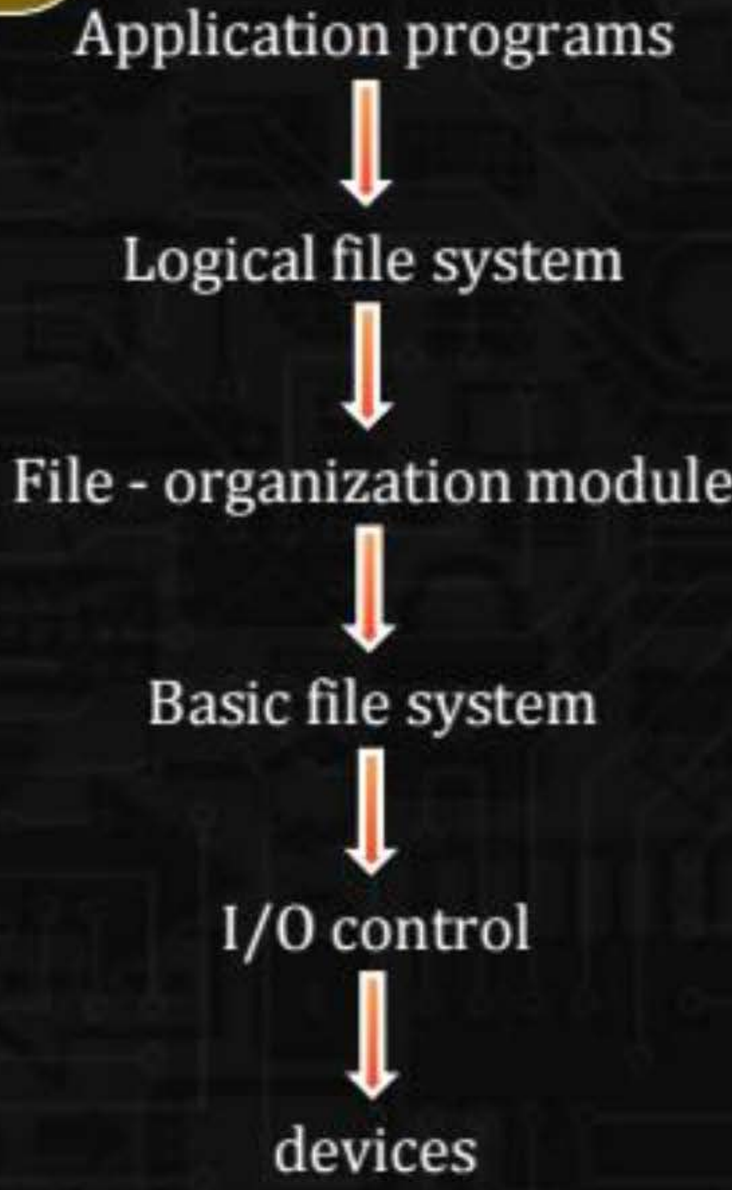
File System & Device Management

Lecture No. 3

By- Dr. Khaleel Khan Sir

- File System Implementation
- Allocation Methods
-

Layered File System



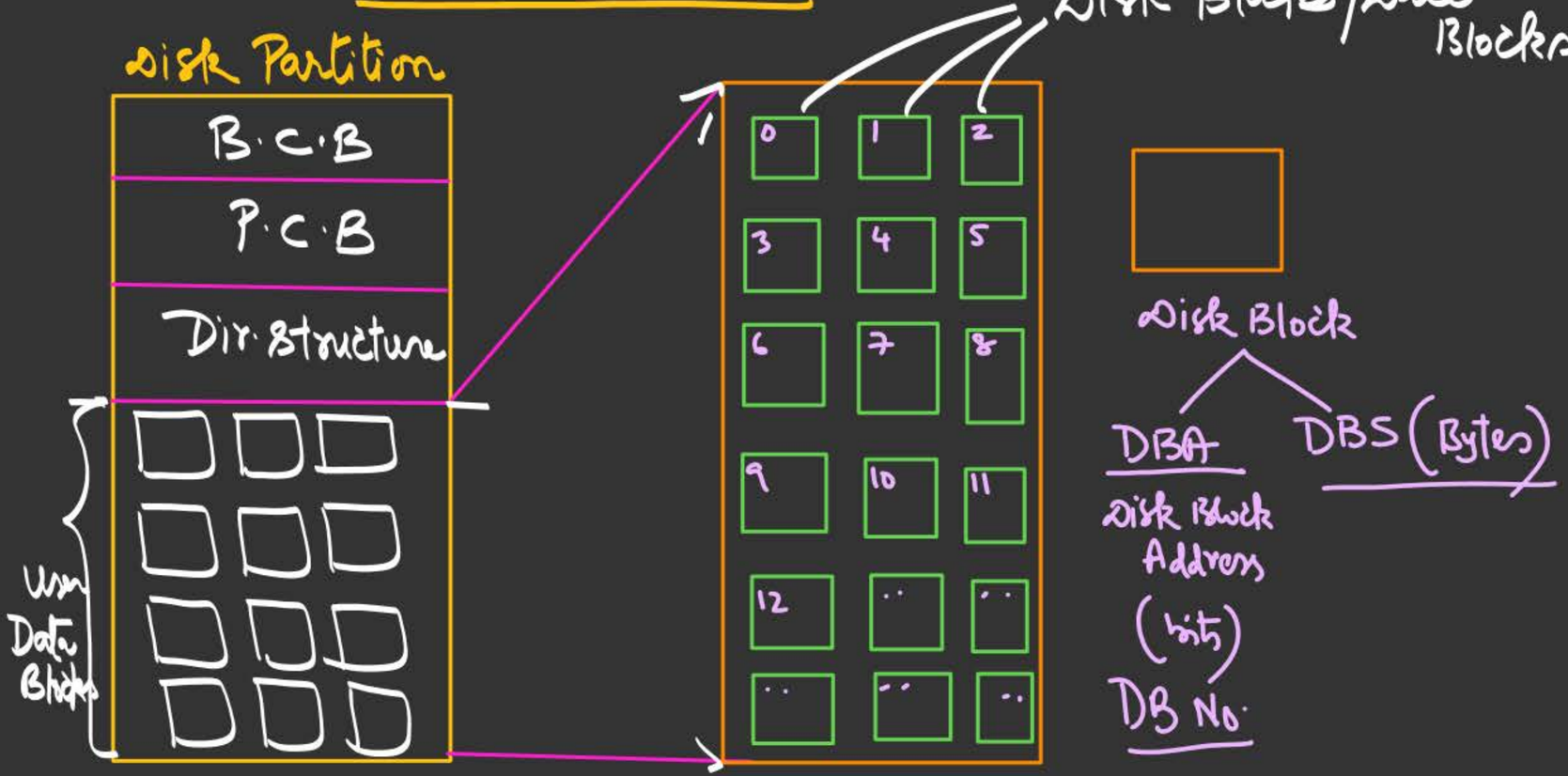
The I/O control level consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. A device driver can be thought of as a translator. Its input consists of high-level commands such as “retrieve block 123." Its output consists of low-level, hardware-specific instructions that are used by the hardware controller, which interfaces the I/O device to the rest of the system. The device driver usually

The Basic File System needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk. Each physical block is identified by its numeric disk address (for example, drive 1, cylinder 73, track 2, sector 10). This layer also manages the memory buffers and caches that hold various file-system, directory, and data blocks.

The File-organization module knows about files and their logical blocks, as well as physical blocks. By knowing the type of file allocation used and the-location of the file, the file-organization module can translate logical block addresses to physical block addresses for the basic file system to transfer.
The file-organization module knows about files and their logical blocks, as well as physical blocks. By knowing the type of file allocation used and the location of the file, the file-organization module can translate logical block addresses to physical block addresses for the basic file system to transfer.
a translation is needed to locate each block, the file-organization module also includes the free-space manager, which tracks unallocated blocks and provides these blocks to the file-organization module when requested.

The Logical File System manages metadata information. Metadata includes all of the file-system structure except the actual data (or contents of the files). The logical file system manages the directory structure to provide the file-organization module with the information the latter needs, given a symbolic file name. It maintains file structure via file-control blocks. A file control block (FCB) (an anode in UNIX file systems) contains information about the file, including ownership, permissions, and location of the file contents.

* Allocation Methods :



DBA = 16 bits ;

DBS = 1 KB ; ✓

⇒ Max. Possible file Size on this disk ;

Max. Possible File Size = Max. Possible (given) Disk Size

$$\text{Max Possible disk Size} = \left(2^{\text{DBA}}\right) * \text{DBS}$$

No. of Blks = 2^{16} = 64K

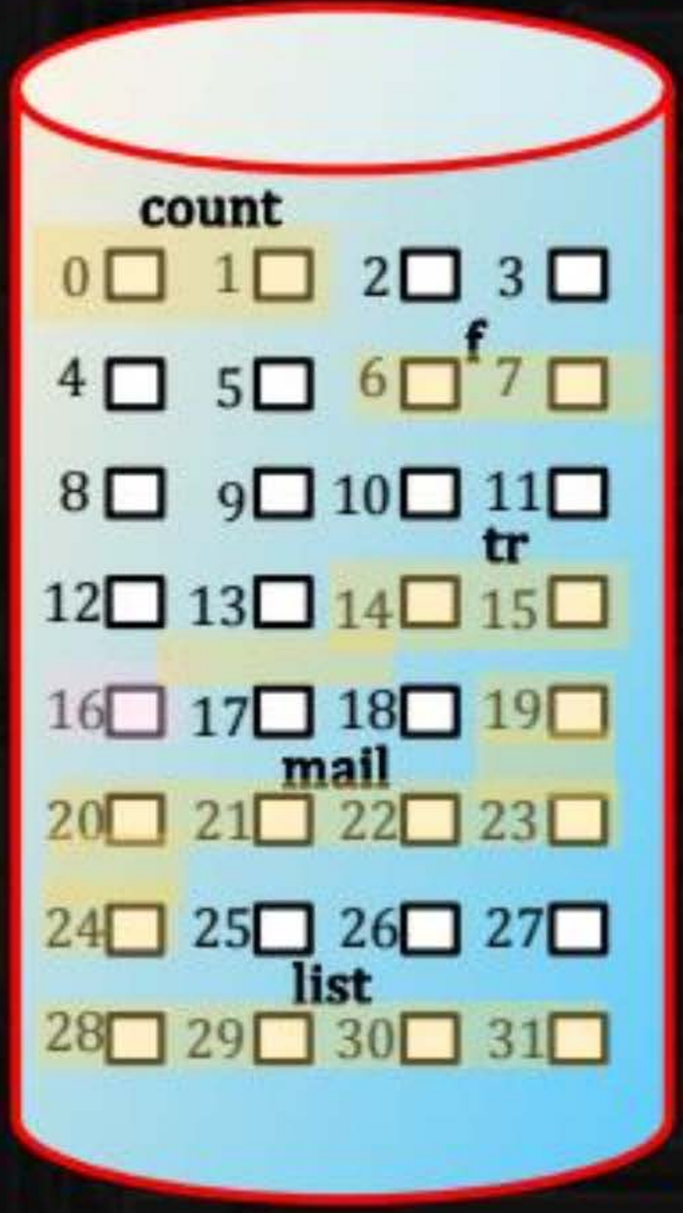
Disk Size = 64K * 1KB = 64MB

Contiguous Allocation of Disk Space

50 free

5 Blks

8 Blks



1.

directory

File	Start	Length
Count	0	2 +2
Tr	14	3
Mail	19	6
List	28	4
F	6	2

Performance of CG Allocation

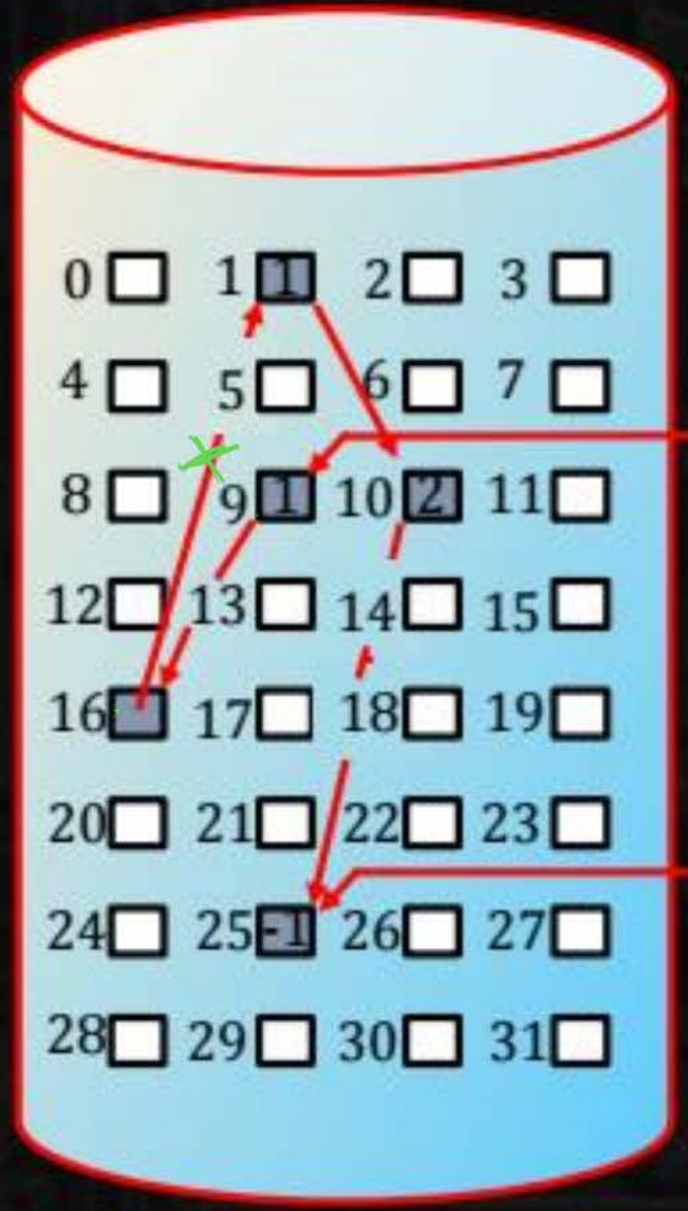
1) Internal fragmentation: ✓ (Last Block)
2) External " : ✓
3) Increasing file size : Inflexible
4) Type of access : Sequential ✓ | Random (direct) → (Faster)

NCG (linked)

1) Int. Frag: ✓ (Last Blk)
2) Ext. Frag: ✗
3) Inc. File Size: Flexible
4) Type of access: only Sequential (Slow)
5) Performance / Efficience: Ptrs Consume disk space
6) Space overhead:
7) Vulnerability of Ptrs: Ptrs Can get broken

Linked Allocation of Disk Space

We are creating linked list of data blocks.



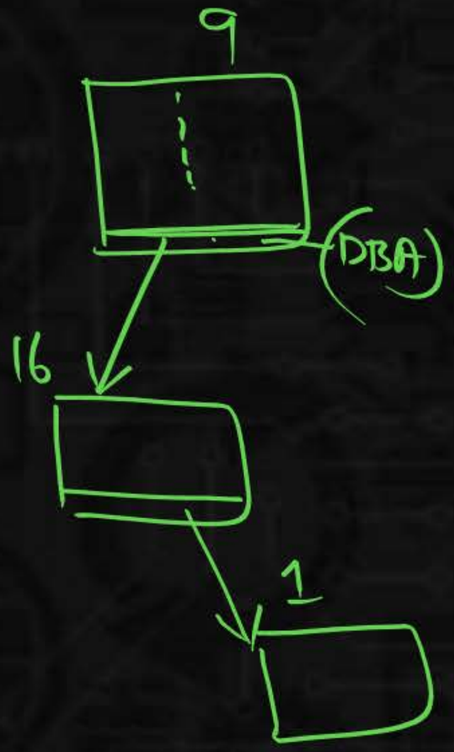
directory

File	start	end
Jeep	9	25

DBA : 32 bits (4B)

(*) Link breaks, File gets truncated;

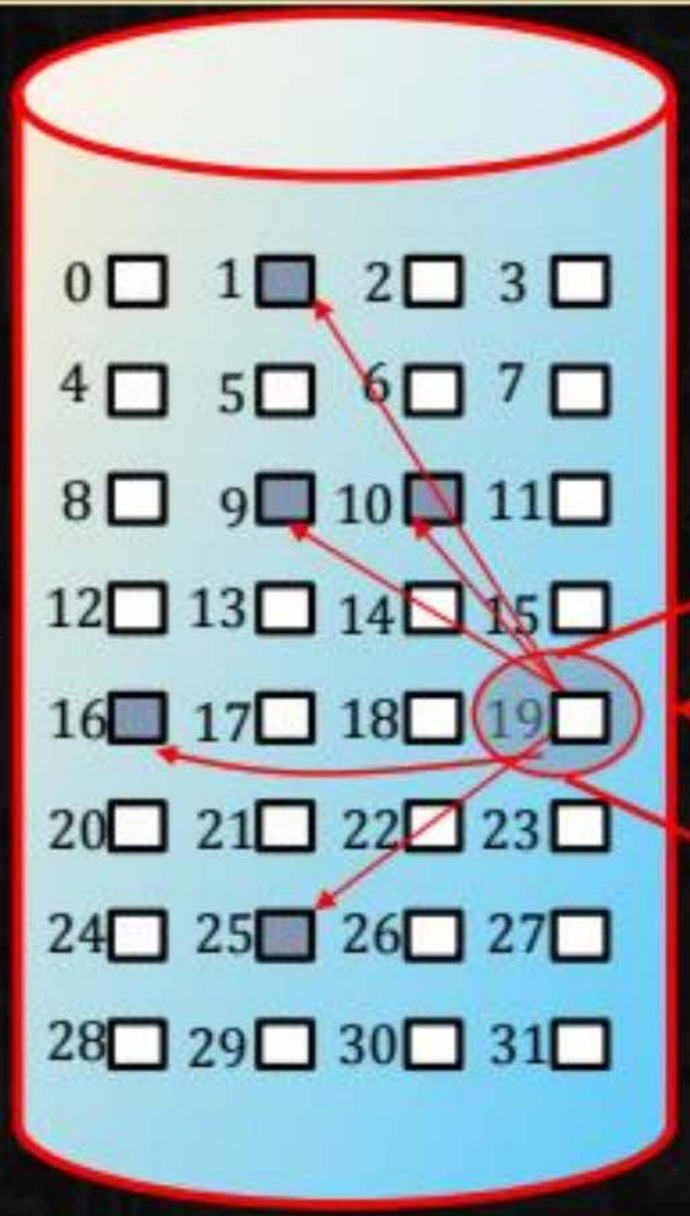
* To add new blks;



Indexed Allocation of Disk Space

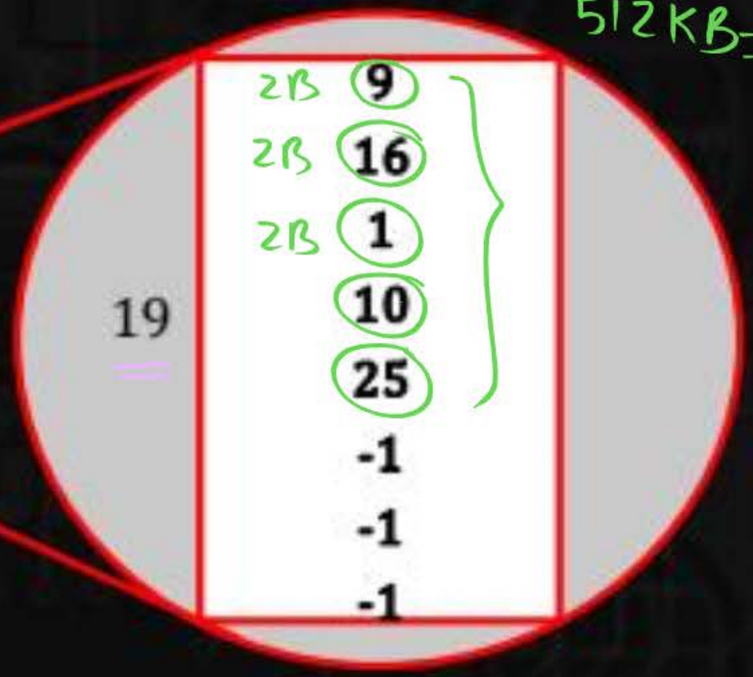
→ Each File has Index Block;

→ Index Blk holds Addresses of Data Blks of File;



directory

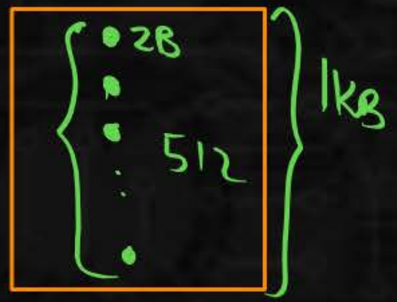
File	index block
Jeep	19



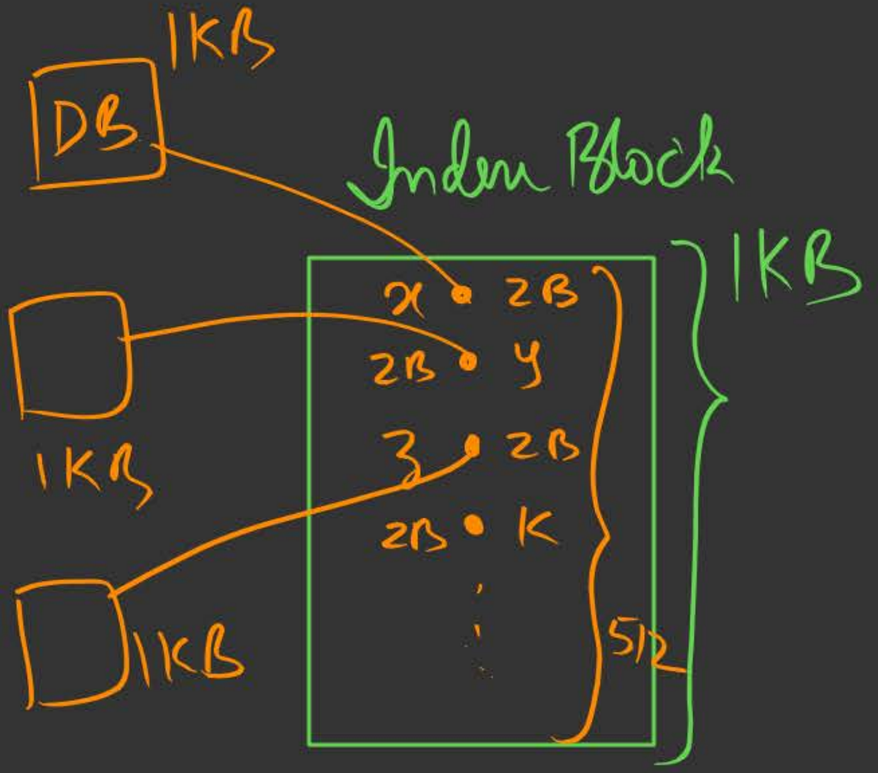
DBA = 16 bits
DBS = 1KB

(Max. File Size with 1 Index Block)

512KB



Index



No. of Addresses

$= \frac{1KB}{2B} = 512$

DBA = 32 bits

DBS = 4KB

Max. Possible file size with Index Block;

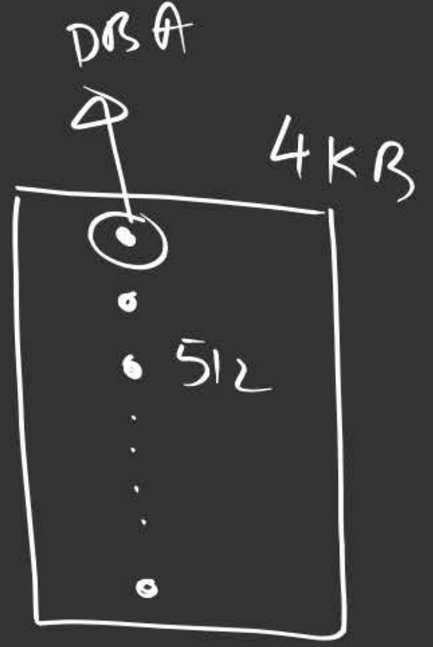
No. of Addresses $= \frac{4KB}{4B} =$ 1K

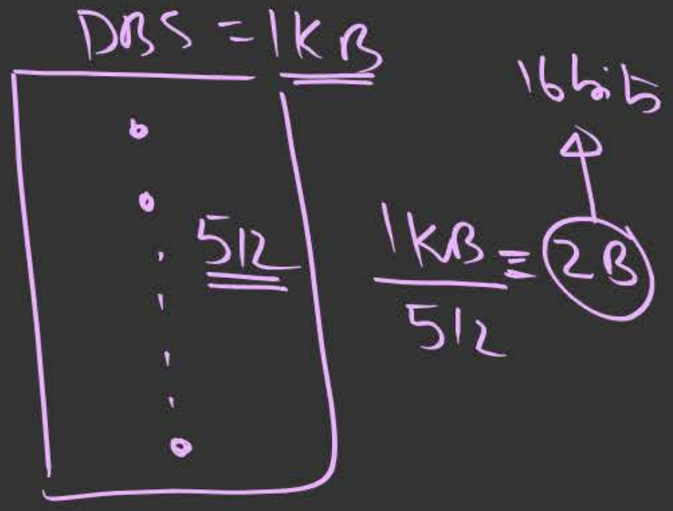
File Size = 1K * 4KB

= 4MB

A file System Supports a DBS = 4KB;
Each Block can hold 512 Addresses (Ptrs)
What is the Size of DBA in bits;

$$N_{Addrs} = \frac{DBS}{DBA}$$ ✓





$$DBA = \frac{DBA}{N_{Addrs}}$$

$$DBA = \frac{4KB}{512} = \frac{2^{12}}{2^{9}} = 2^{3}$$

$$= 8B$$

$$= 64 bits$$ ✓

Case Studies of O.S:

I. UNIX/LINUX:

Each file is assoc with a I-Node (Index)

File Name	I-Node
KK.C	23

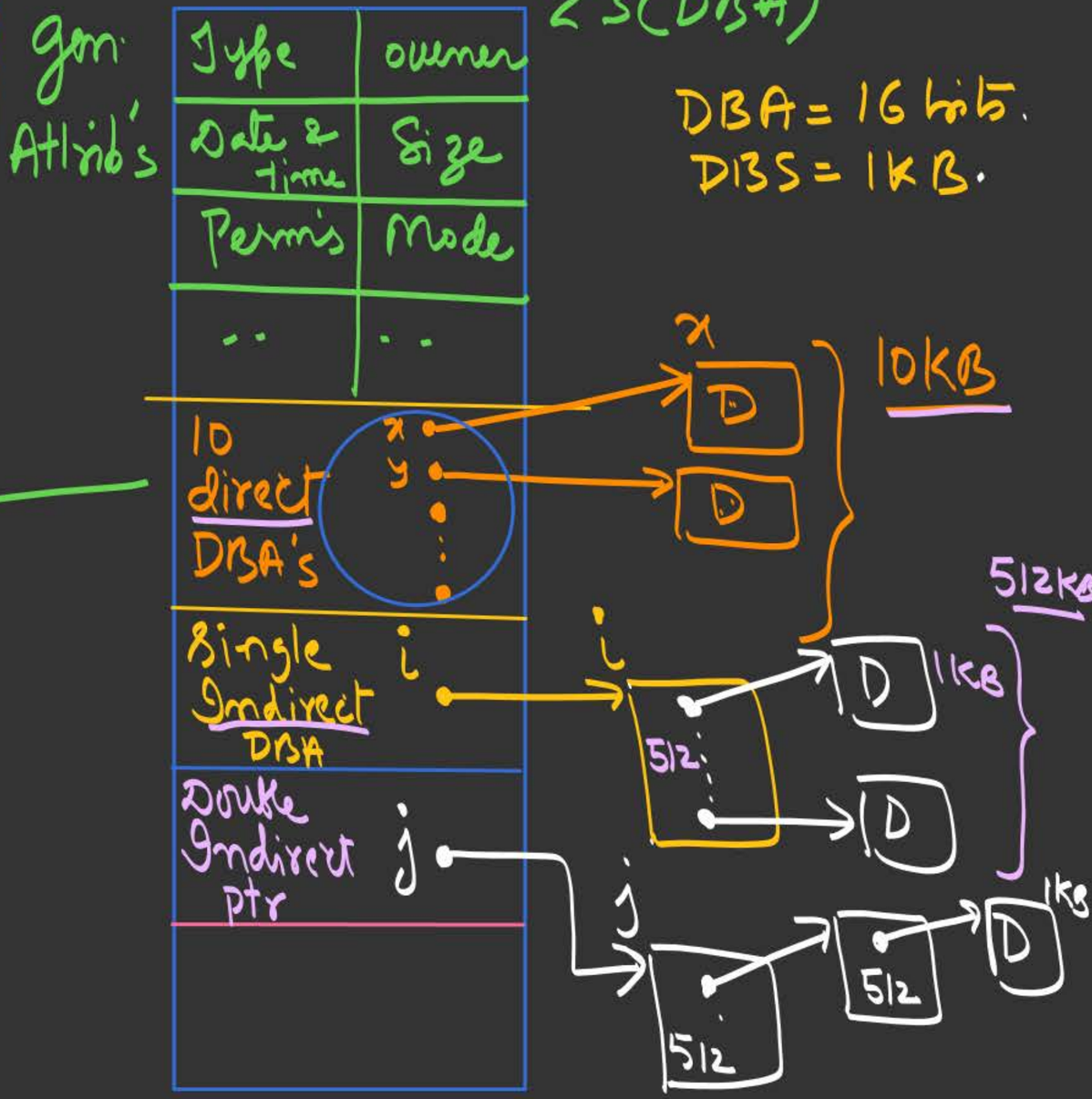
Directory

(Blocks in use by the file)

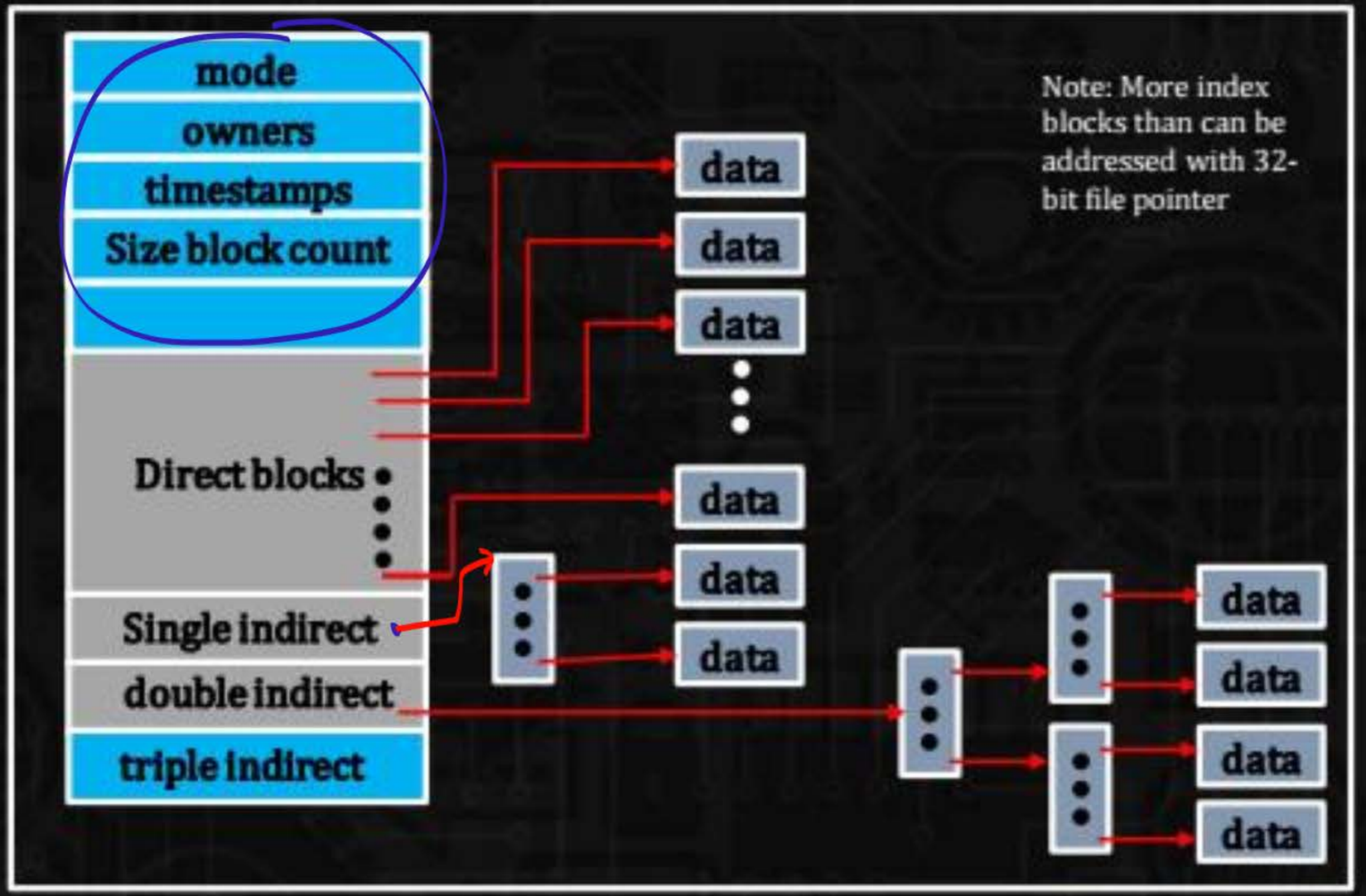
I-Node-Structure:

23 (DBA)

DBA = 16 bits.
DBS = 1KB.



The Unix I-Node



Double Indirect

$512 * 512 * 1KB$

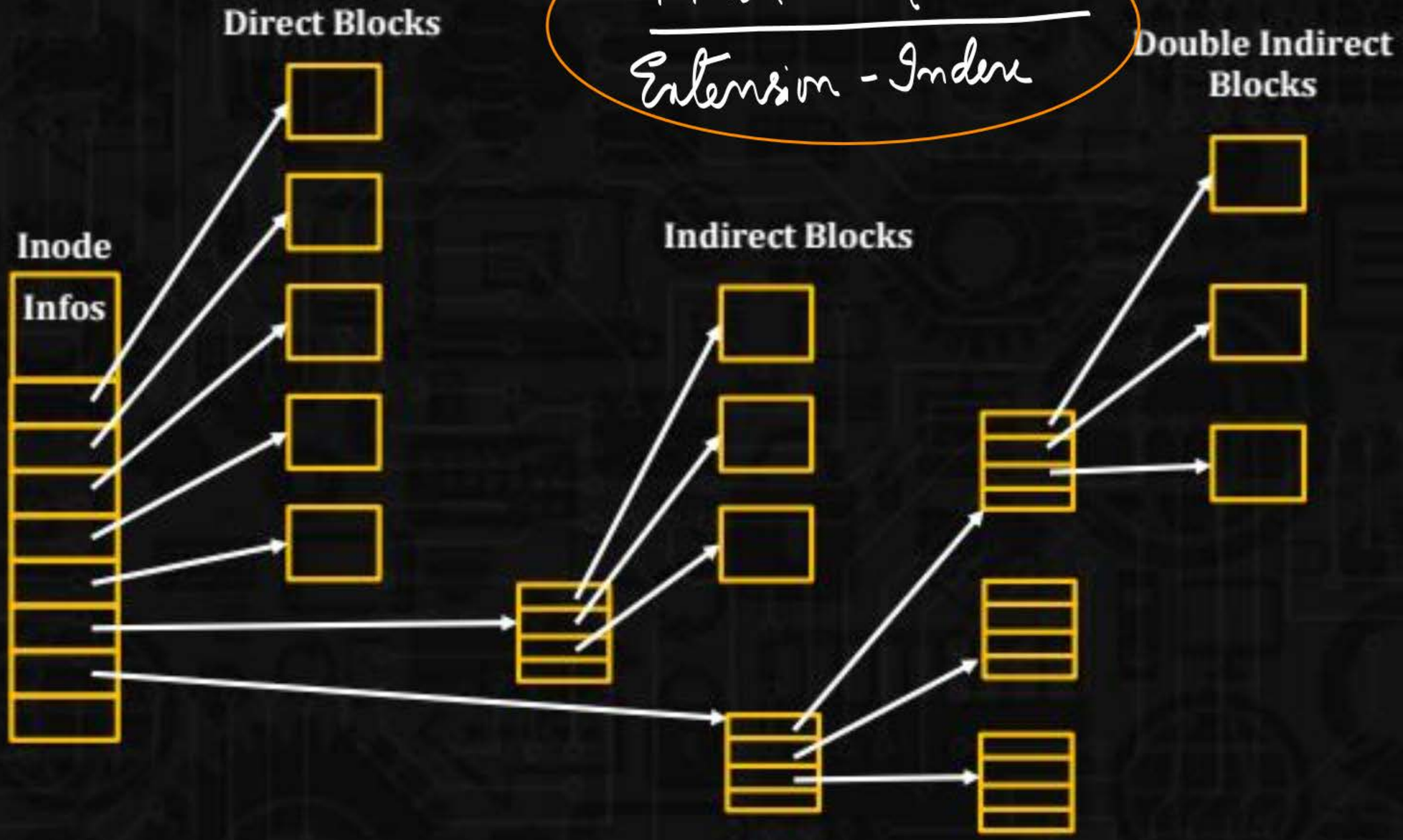
$2^9 \times 2^9 \times 2^{10}$

$= 2^{28} = 256MB$

Total File Size = 256.522MB

Multi-Level Index

Extension - Index



II. DOS/WINDOWS:

Block Info.

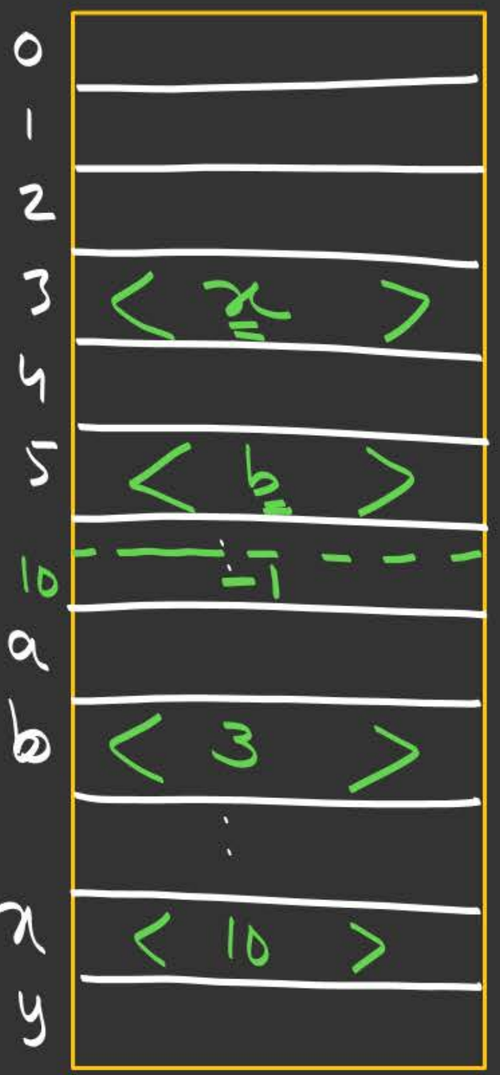
Other Attributes

File Name	. . . . .	First DBA	Last DBA
KK.C	. . . . .	5	10

Directory

<5, b, 3, x, 10>

Master File Table / F.A.T



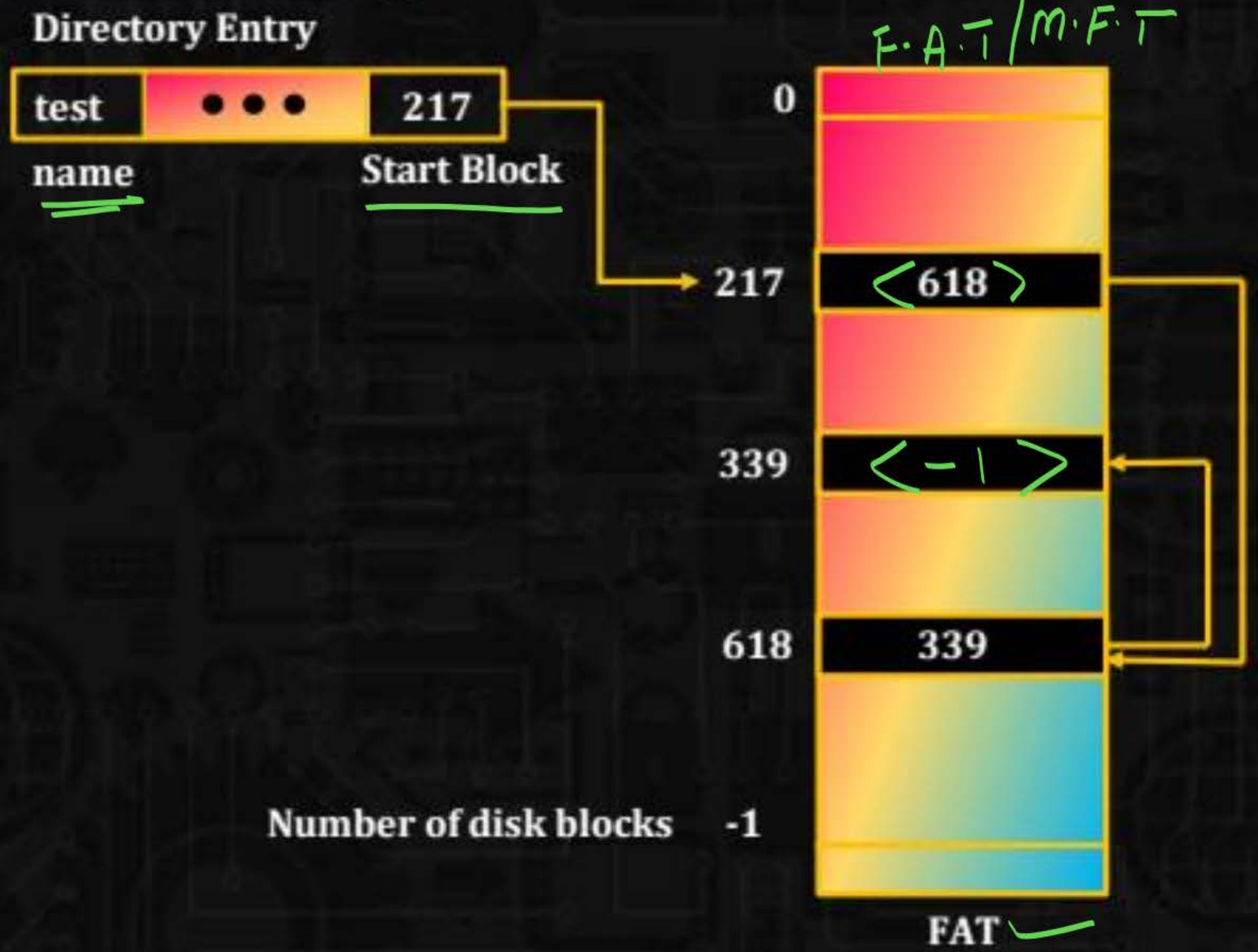
Linked Alloc

→ No. of entries in FAT / MFT = No of Blocks

→ FAT entry contains Address (DBA) of Next Data Blk

FAT-32 FAT entry Size (DBA)

File Allocation Table



Q.

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

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. In a file allocation system, which of the following allocation scheme(s) can be used if no external fragmentation is allowed?

I. Contiguous

II. Linked ✓

III. Indexed ✓

A. I and III only

B. II only

C. III only

D. II and III only ✓

Q.

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

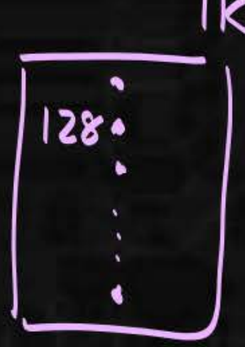
Disk Block Size is 1Kbytes & each Block can hold 128 Disk Block Addresses.

Calculate

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

(ii) Size of Disk Block Address? 64 bits

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



$$DBA = \frac{1KB}{128} = \frac{2^{10}}{2^7} = 2^3\ B = 8B = 64\ bits$$

$$= \underbrace{8*1KB}_{\text{direct}} + \underbrace{128*1KB}_{S.I} + \underbrace{\frac{128*128KB}{2^{24}}}_{D.I} + \underbrace{\frac{128*128*128KB}{2^{31}}}_{T.I}$$

$$= (8KB + 128KB + 16MB) + 2GB$$

$16.136MB = 0.016GB$

$= 2.016GB$

$\sim 2GB$ ✓

Q.

Consider a File System that stores 128 Disk Block Addresses in the index table of the Directory. Disk Block Size is 4 Kbytes. If the file size is less than 128 Blocks, then these addresses act as direct Data Block addresses.

However, if the File Size is more than 128 Blocks, then these 128 addresses in the Index table point to next level Index Blocks, each of which contain 256 Data block addresses. What is the Max File Size in this File System?

Q. 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 ___ GB (rounded off to 1 decimal place)

$$N_{\text{addresses}} = \frac{4KB}{4B} = 1K$$

$$\text{File Size} = \underbrace{12*4KB}_{\text{direct}} + \underbrace{1K*4KB}_{\text{SI}} + \underbrace{1K*1K*4KB}_{\text{DI}}$$

$$= 48KB + 4MB + 4GB$$

$$= 4.004GB \sim 4GB \checkmark$$

Q. 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

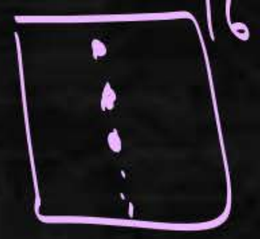
DBS = 128 B
DBA = 8 B

$N_{addr} = \frac{128}{8} = 16$

Size = 8 * 128 B + 16 * 128 B + 16 x 16 x 128 B

= 1 KB + 2 KB + 32 KB

= 35 KB ✓



A. 3 K Bytes

B. 35 K Bytes

C. 280 K Bytes

D. Dependent on the size of the disk

Q. The Data Blocks of a very large file in the Unix File System are allocated using

A. Contiguous allocation

B. Linked allocation

C. Indexed allocation

D. An extension of indexed allocation. ✓

Q. Using a Larger Block size in a Fixed Block Size File System leads to

A. Better Disk Throughput but Poorer Disk Space Utilization.

B. Better Disk Throughput and Better Disk Space Utilization

C. Poorer Disk Throughput but Better Disk Space Utilization

D. Poorer Disk Throughput and Poorer Disk Space Utilization

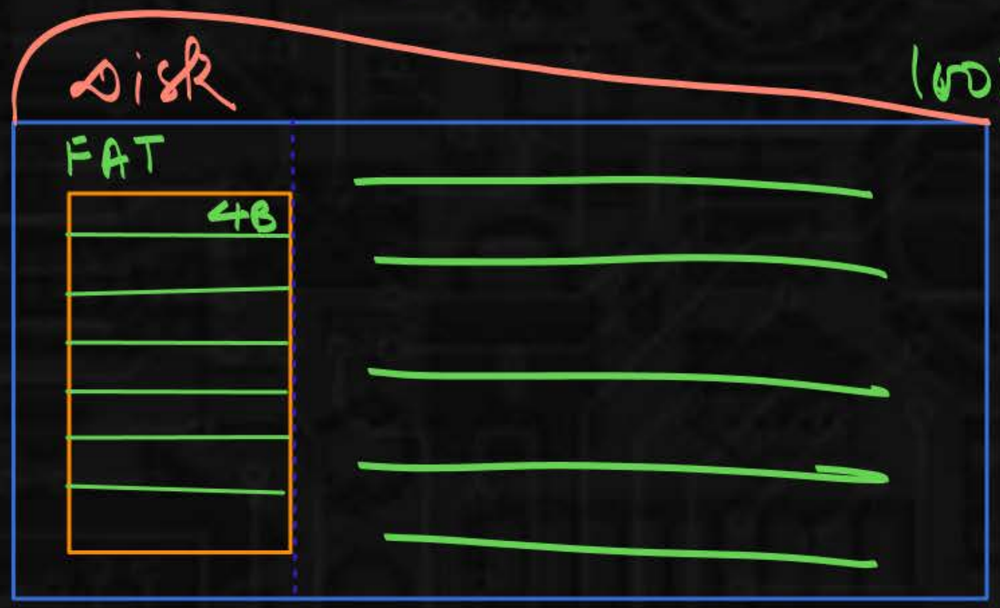
Q.

(*)

A FAT (File allocation table) based file System is being used and the total overhead of each entry in the FAT is 4 bytes in size. Given a 100×10^6 bytes' disk on which the file System is stored and data block size is ~~1 03~~ bytes, the maximum size of a file that can be stored on this disk in units of 10^6 bytes is 99.6. ✓

(NAT)

$\rightarrow 10^3$ Bytes



100×10^6 B

$N = \frac{100 \times 10^6}{10^3} = 100 \times 10^3$

DBS $= 10^3$ By

Max File Size = Give Disk Size − FAT Size

FAT Size $= 100 \times 10^3 * 4B$

$= 400 \times 10^3$ B

$= 4 \times 10^5$ B

$= 0.4 \times 10^6$ B ✓

$= 100 \times 10^6 - 0.4 \times 10^6 = 99.6 \times 10^6$

Q.

(*) H/W

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:

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.
Disk Block 2, 3 : Directory with 32-bit entry per File.
Disk block 4 : Data block 1.
Disk Block 5 : Data Block 2,3 etc;

(a) What is the Maximum possible number of Files?
(b) What is the Maximum Possible File size in Bytes?

9:00 am


THANK YOU GW
SOLDIERS !