CS & IT ENGINEERING

Operating System

File System & Device Management

Lecture No. 05

By- Dr. Khaleel Khan Sir

- File System Implementation
- Allocation Methods
- Problem Solving

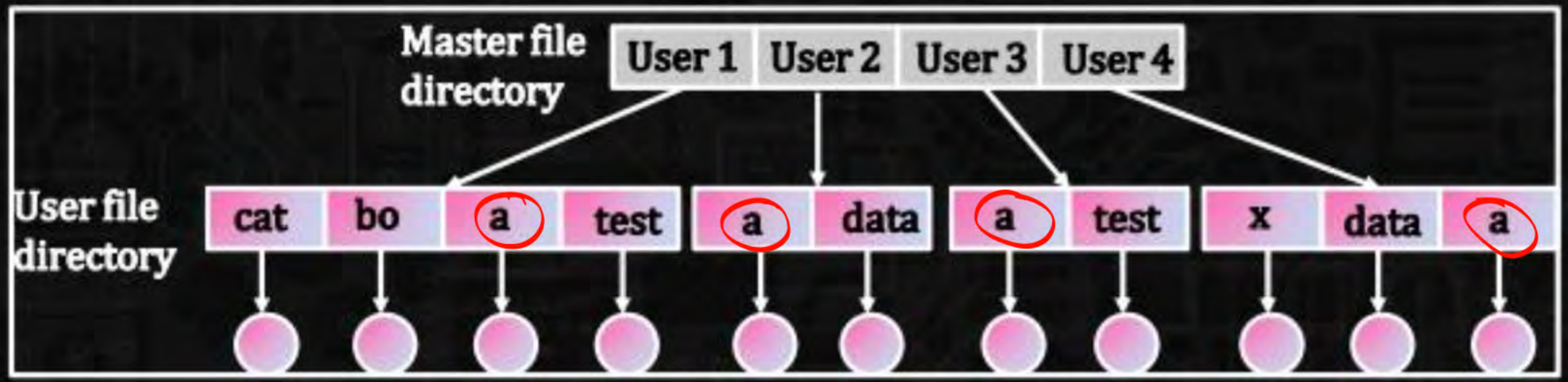
A single directory for all users

Naming problem

Grouping problem

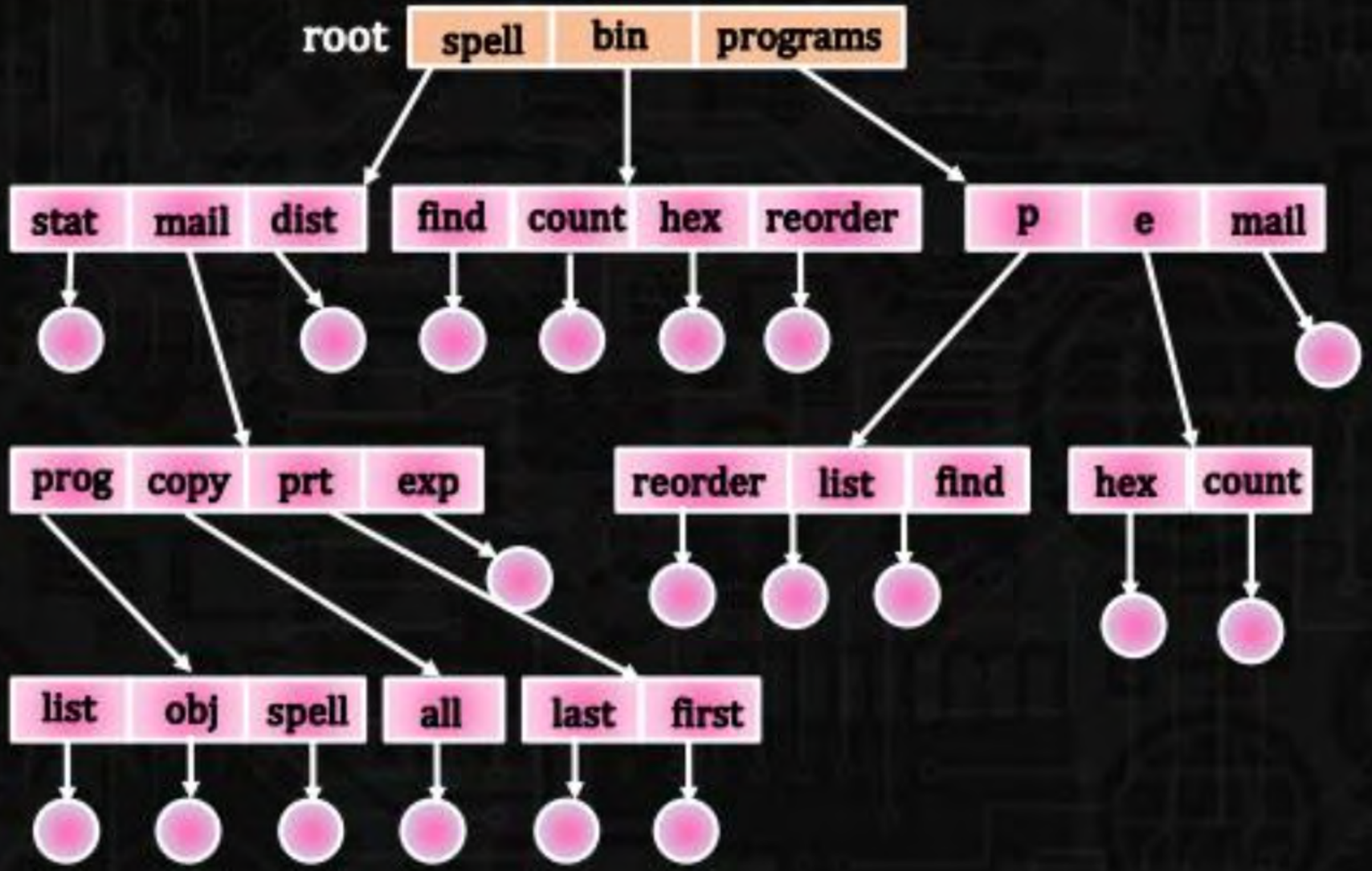
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories

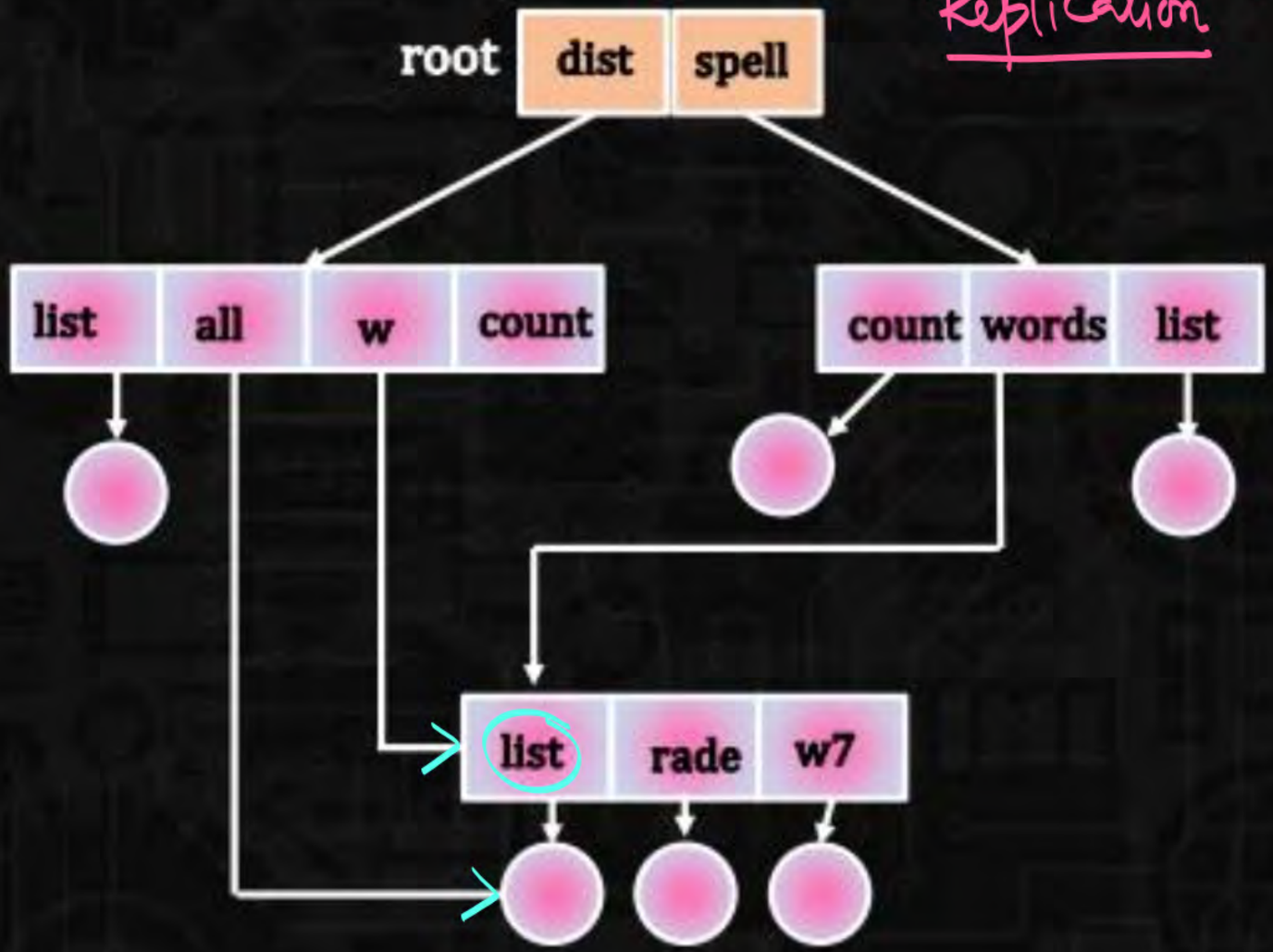


Acyclic-Graph Directories

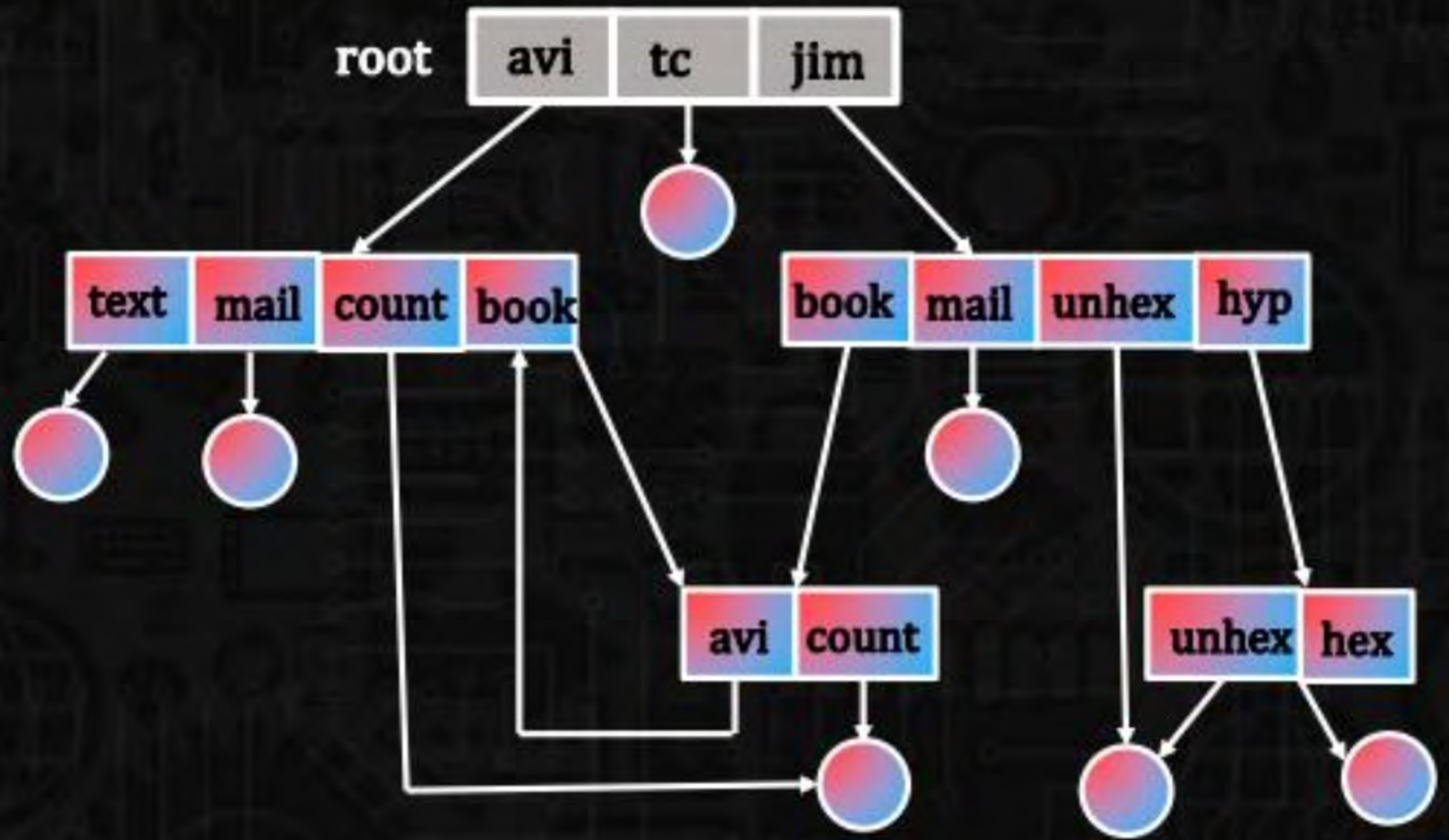
(File Sharing)

Replication

Links (Count)

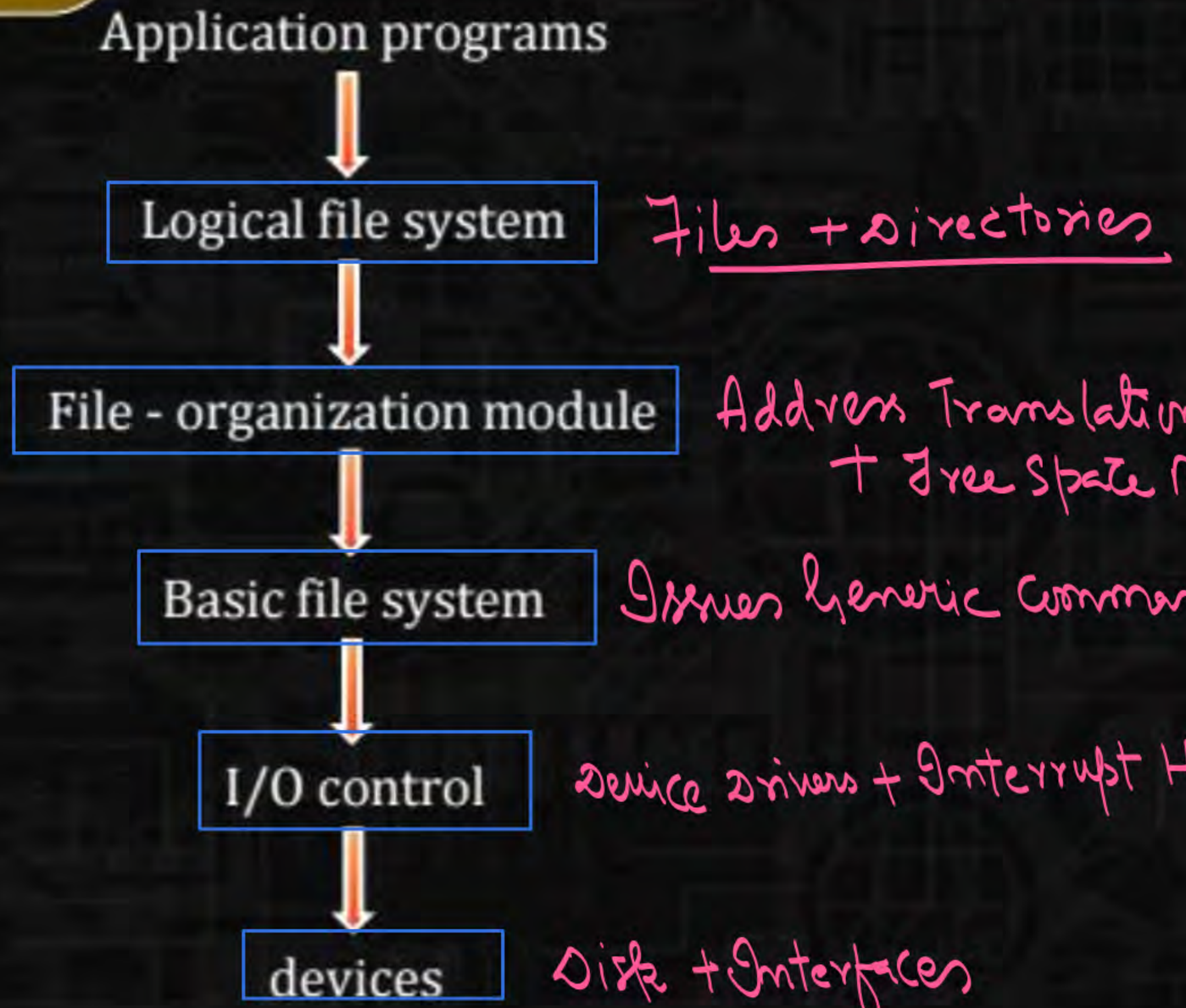


General Graph Directory



Layered File System

File System Implementation

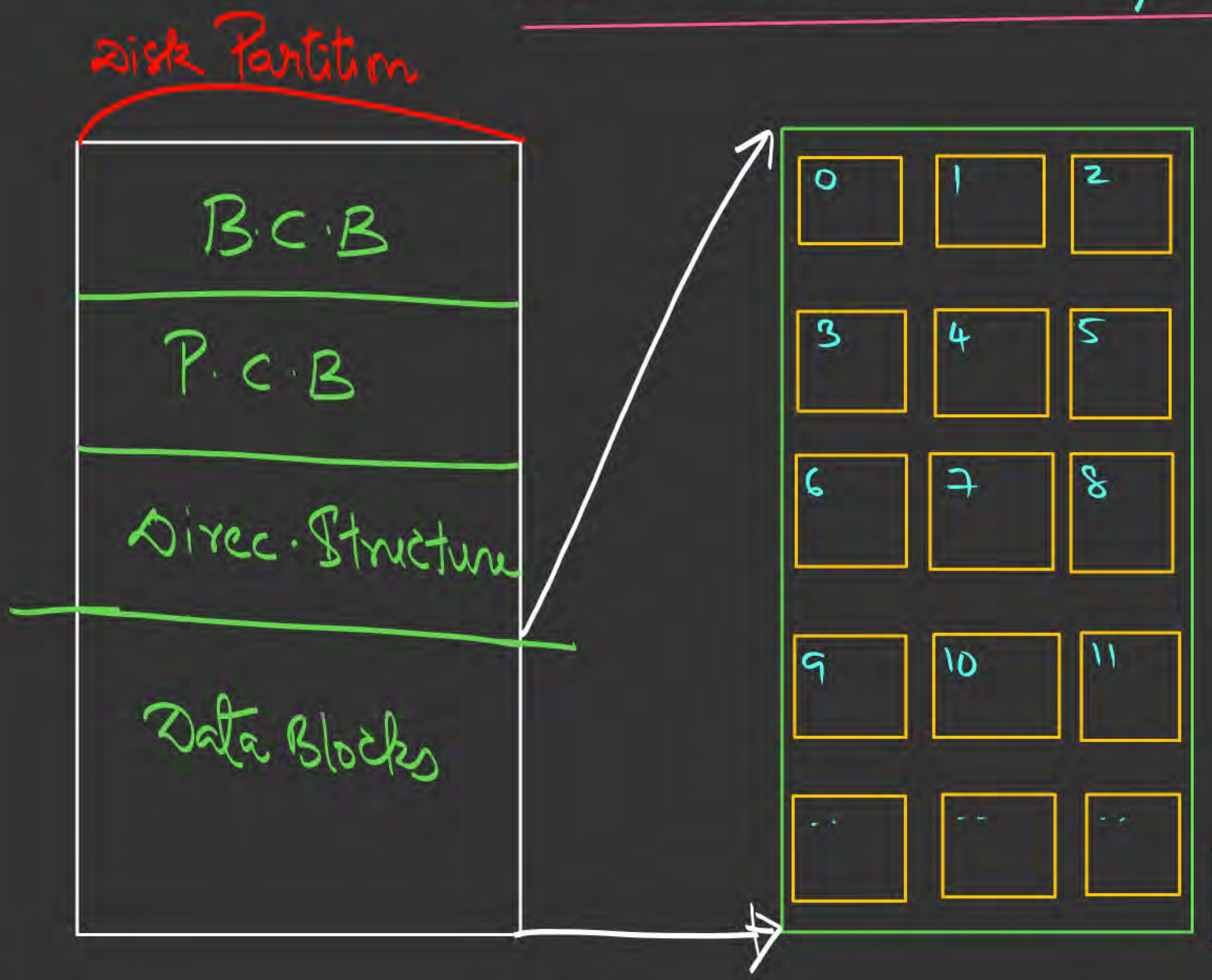


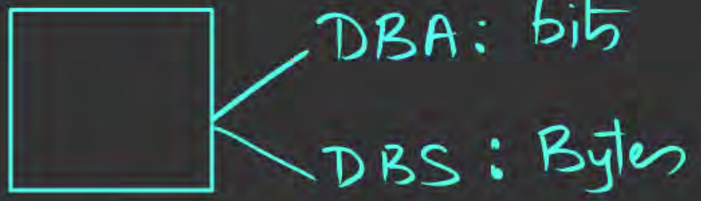
The I/O control level consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. A device driver can be thought of as a translator. Its input consists of high-level commands such as "retrieve block 123." Its output consists of low-level, hardware-specific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

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. A block

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 tile-organization module also includes the free-space manager, which tracks unallocated blocks and provides these blocks to the file-organization module when requested.
Finally, 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 inode in UNIX file systems) contains information about the file, including ownership, permissions, and location of the file contents. The

* File Allocation Methods / Disk Space Alloc Strategies





DB
<Data Block
Disk Block>

Ex: DBA: 16 bits; DBS = 1KB

Max File Size = 2^{16} * 1KB

Disk Size = 64MB

Max Disk Size = 2^{DBA} * DBS

File : 15 KB

DBS = 2 KB

$N_{BlKS} = \frac{15KB}{2} = 7.5$

$= 8$ Blks

Allocation?

a) Contiguous Allocation

Performance :

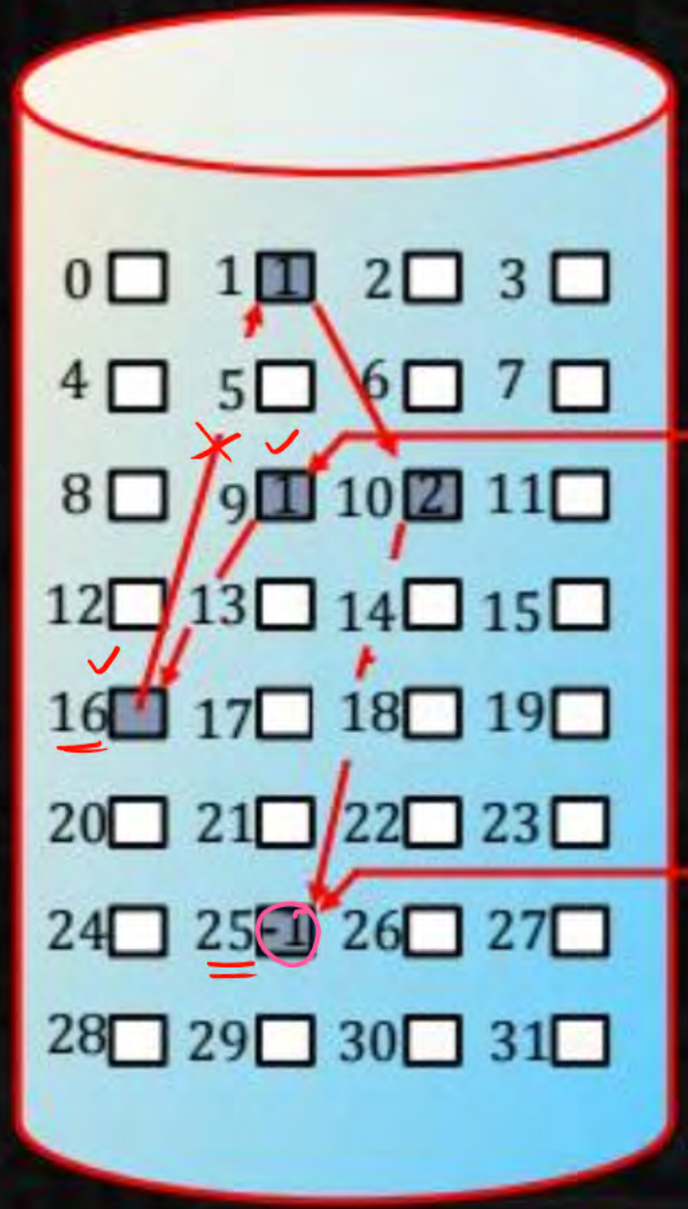
1) Internal Frag : ✓ (In last BlK)
2) External Frag : ✓
3) Type of Access : Both (Seq + Random) (Faster)
4) Inc. File Size : (Inflexible)

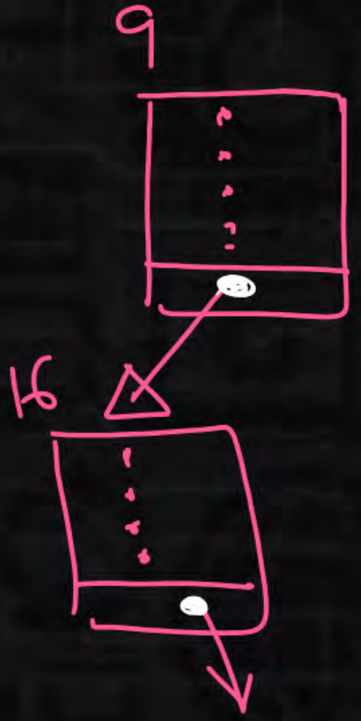
Contiguous Allocation of Disk Space

directory

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

Linked Allocation of Disk Space : (Real/Physical linked list)





directory

File	start	end
Jeep	9	25

1. Int. Frag : ✓

2. Ext. Frag : - No -

3. Type of access: Sequential (Slow)

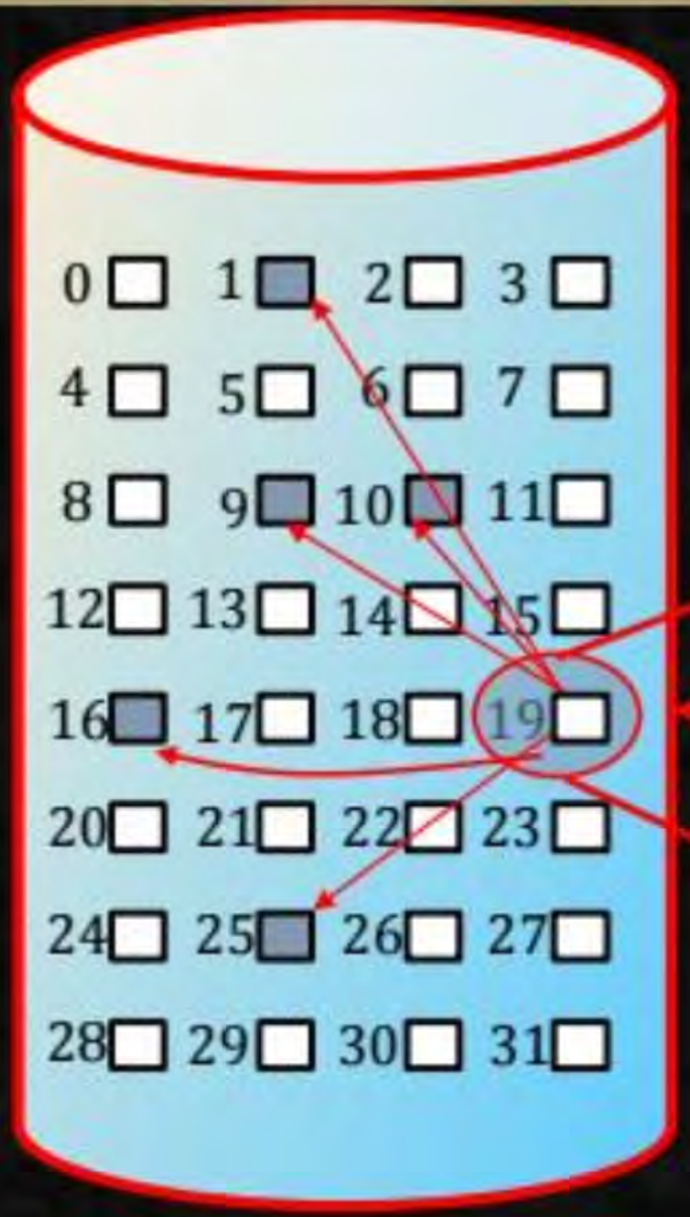
4. Inc File Size: Flexible

5. Space ovhd : Some amount of disk space is consumed by ptrs/links;

6. Vulnerability of Ptrs : If ptr/link breaks the file gets truncated

Indexed Allocation of Disk Space

Each file is associated with an Index Block

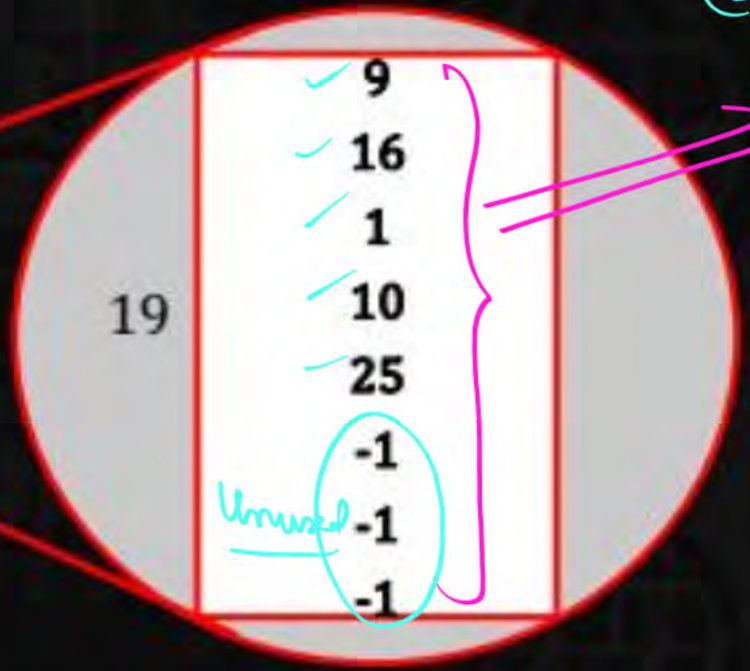


directory

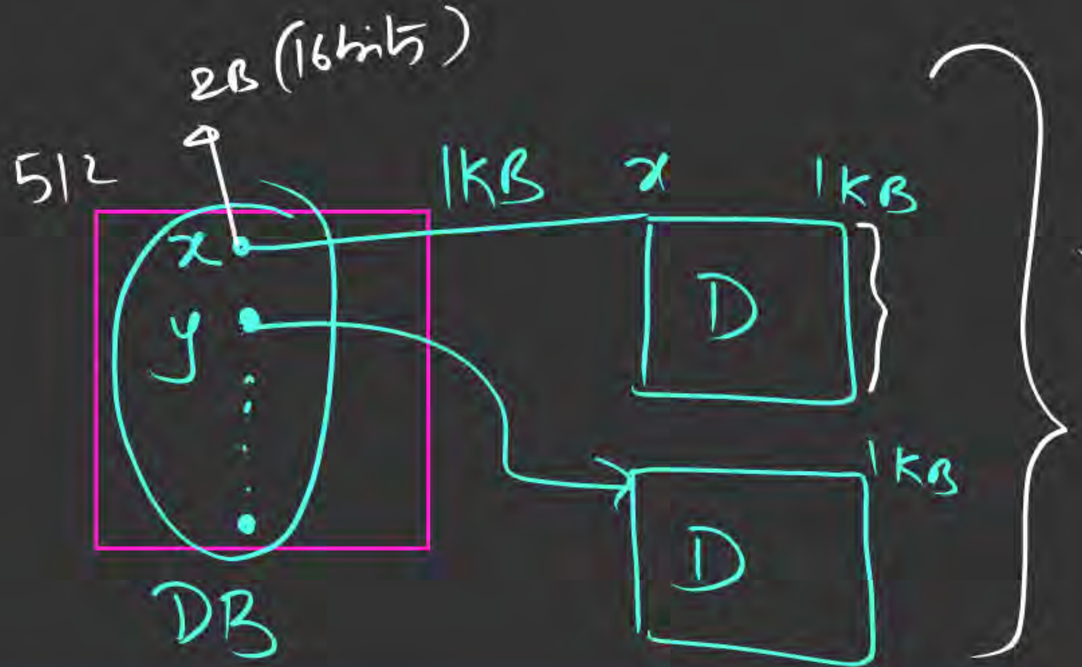
File	index block
Jeep	19

DBA = 16 bits
DBS = 1 KB

Max File Size = (1 - Index Blk)



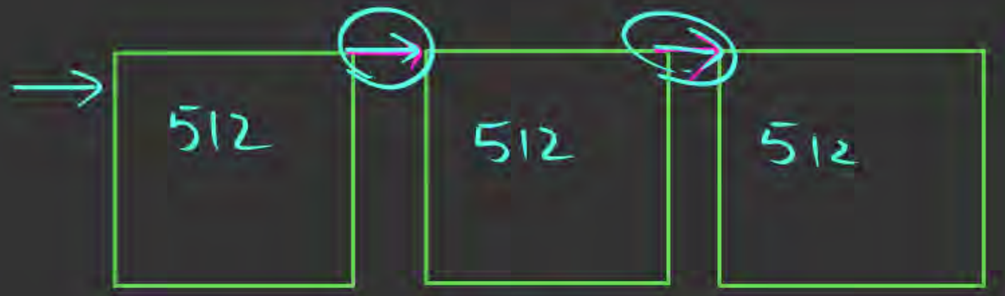
Addresses of Data Blks in use by the file



Total File Size = 512 * 1KB

= 512KB.

No. of Addresses that can be stored in one Index Blk $= \frac{1KB}{2B} = 512$



3 * 512KB = 1.5MB

1) I.F : ✓

2) E.F : ✗

3) Inc File Size : Flexible

4) Type of Access : Both

5) Reliability : is more In Comp. to linked Alloc.

6) Space overhead : is there but less
(Index-Blks)

Case-Studies :

1) UNIX | LINUX

File Name	I-Node #
KK.C	23

1 (Index) → 512KB

Loc. Attrib's

Total File Size = 512 * 512KB

D.I = $2^9 \times 2^9 \times 2^{10}$

= 2^{28} = 256MB

Final File Size

256.522MB

Unix-I-Node Structure

23

Size	owner
Type	Pr. Info
Date	...
10 direct DBA's	
Single Indirect DBA	
Double Indirect DBA	

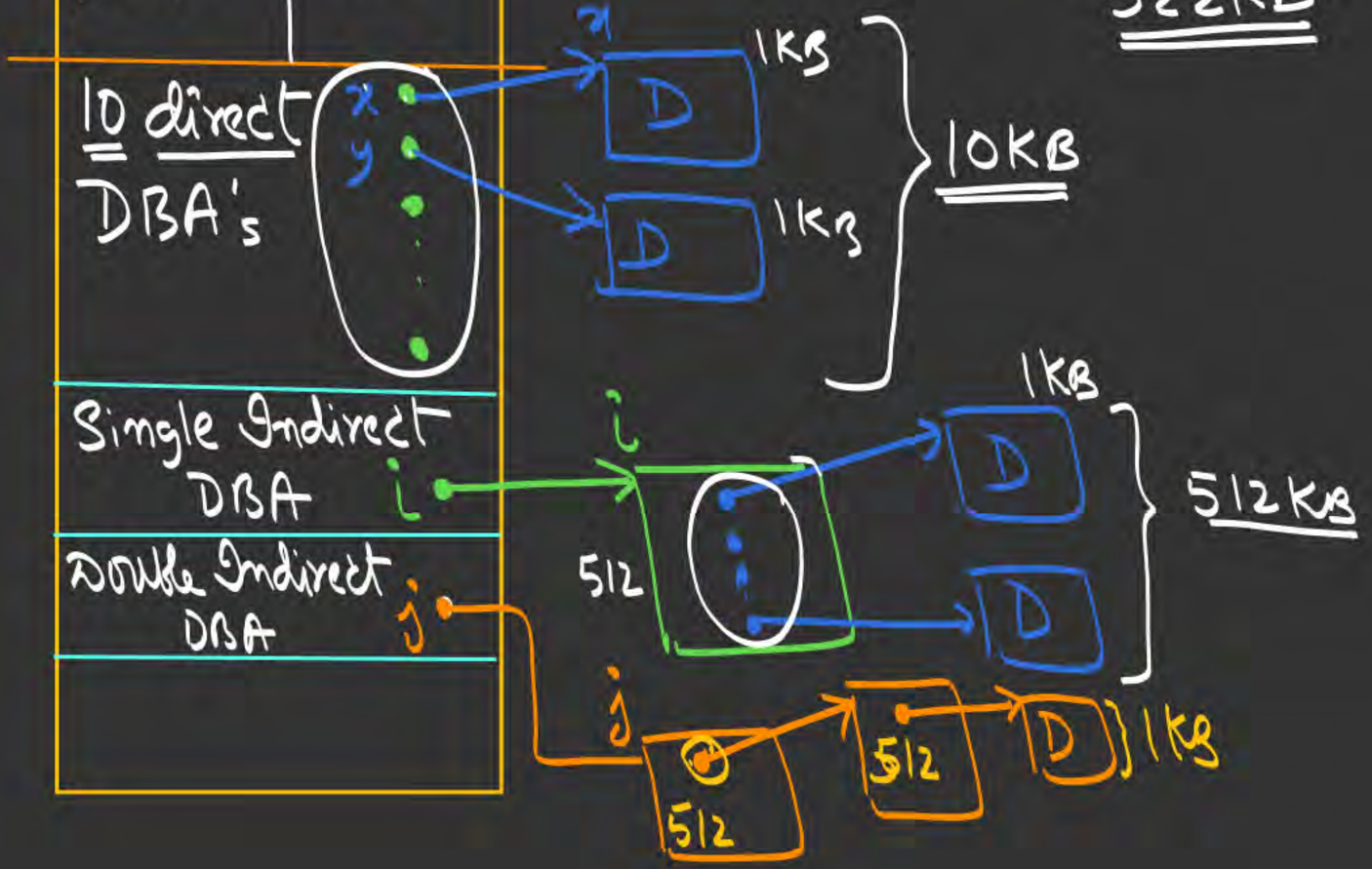
Gen_Attrib's

64MB

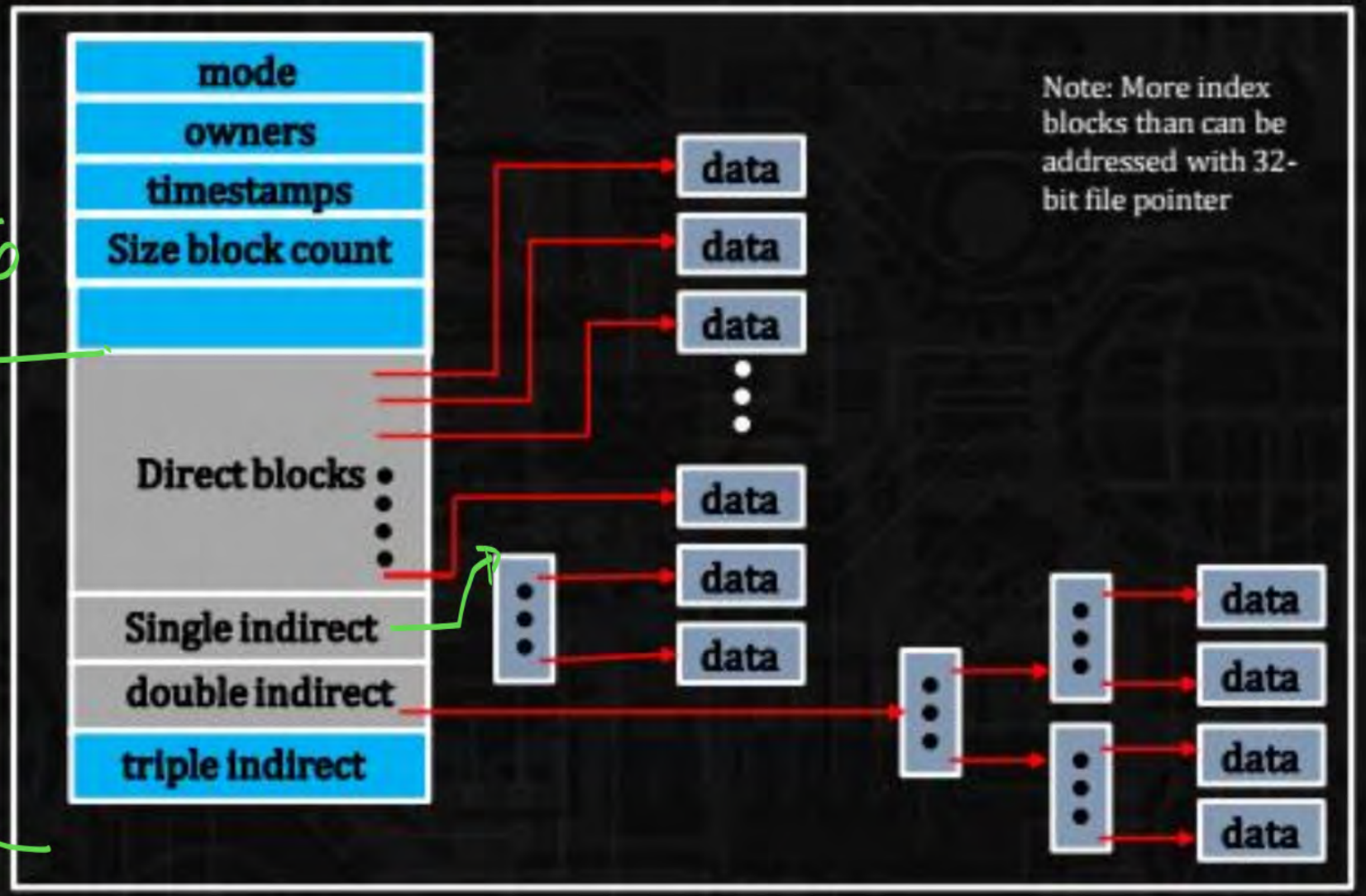
DBA = 16bits

DBS = 1KB

522KB



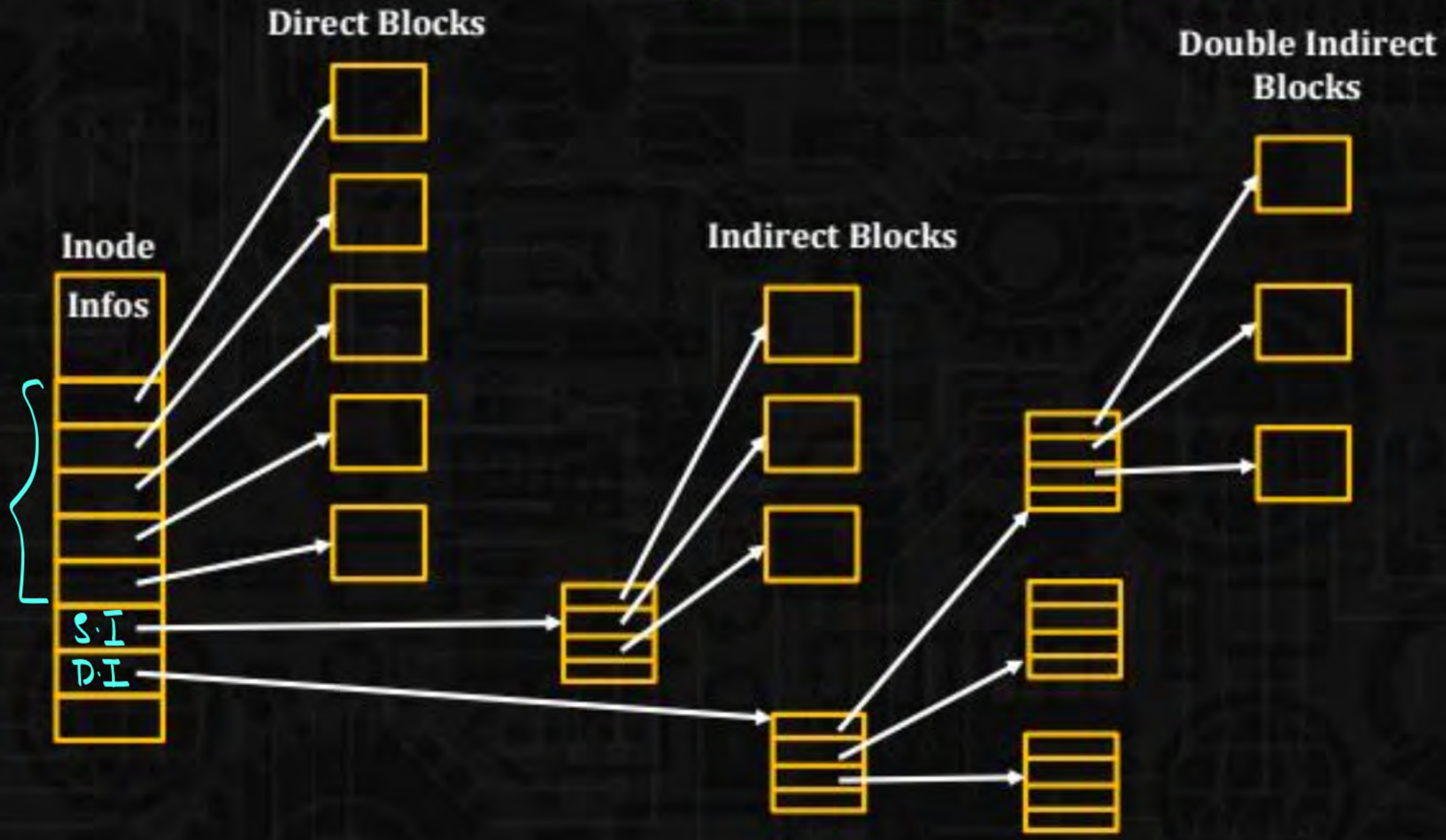
The Unix ~~Code~~ I-Node



Gen Attributes

Location

UNIX - I-node Structure



Case-Study-II: <DOS/WINDOWS>

	Gen Attrib's	Loc Attrib's
File Name	. . . . .	First DBA
KK.C	. . . . .	y

Directory

<y; 3; z; 2> → linked allocation

DB Blocks

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

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

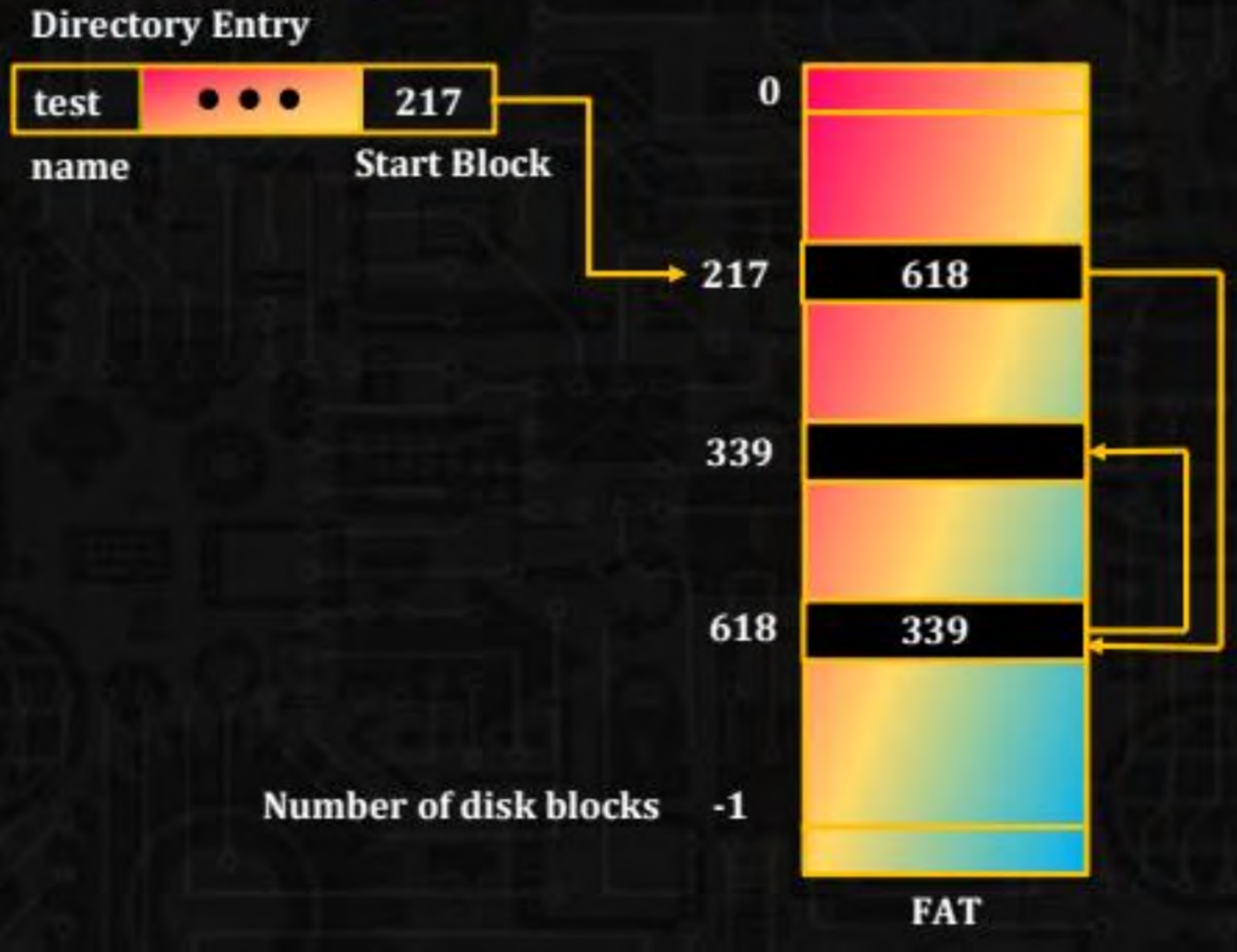
FAT → 32 entry Size (DBA)

→ No. of entries in FAT = No. of Blks on disk

→ FAT entry contains Address(DBA) of next Data Blk in use

→ FAT entry Size ~ DBA

File Allocation Table



Q.

MSQ

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?

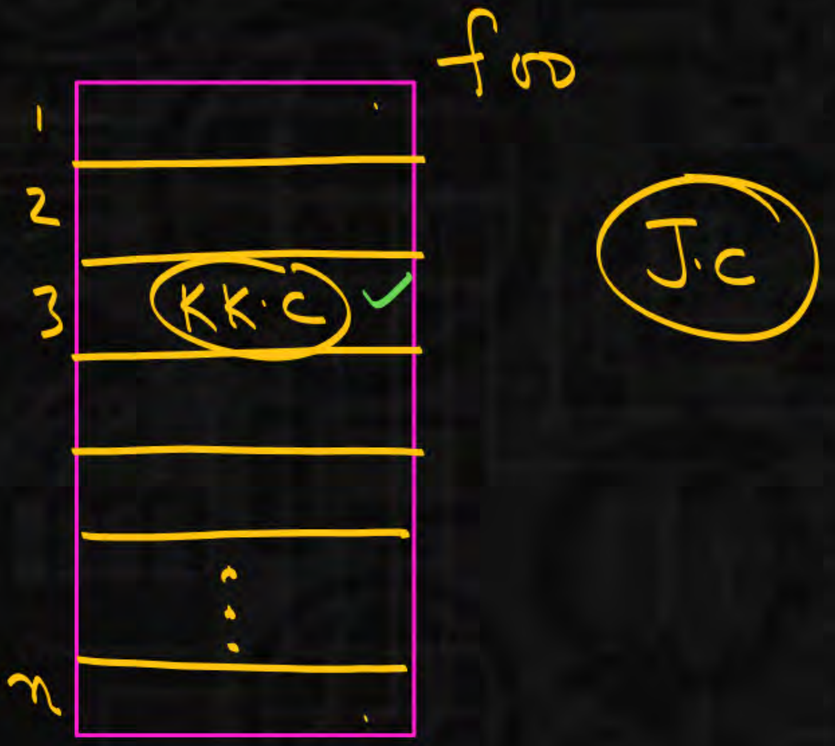
<B, C>

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.

H/W

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?

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.

H/W

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.

H/W

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)

Q.

PYQ

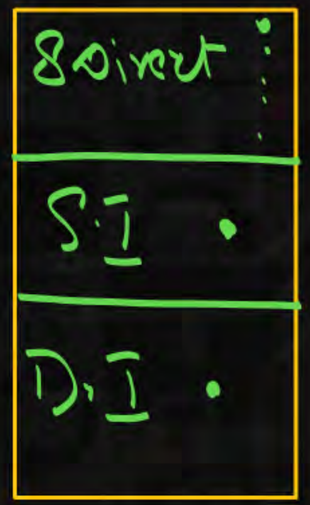
A File System with 300 G Byte Disk uses a File descriptor with 8 Direct Block Addresses, 1 Indirect Block Address and 1 Doubly Indirect Block Address. The size of each Disk Block is 128 Bytes and the size of each Disk Block Address is 8 Bytes. The maximum possible File Size in this file System is

A. 3 K Bytes

B. 35 K Bytes

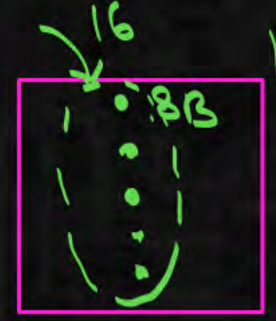
C. 280 K Bytes

D. Dependent on the size of the disk



DBA = 8 Bytes

DBS = 128 Bytes



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

$\therefore 8 \times 128B + 16 * 128 + 16 \times 16 \times 128B$

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

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.

(multi-Level)

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

More I·F

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.

H/W

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 103 bytes, the maximum size of a file that can be stored on this disk in units of 10^6 bytes is _____.

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?


THANK YOU GW
SOLDIERS !