CS & IT [redacted]

ENGINEERING

Operating System

File System

Lecture No. 02

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic File System

Topic Directory

Topic Disk Blocks

Topics to be Covered

Topic File Allocation Method ✓

Topic Unix i-node

Topic Disk Cylinder, Seek Time

Topic : Question

→ files

A system directory is kept in 4 disk blocks each of size 2Kbytes. It is a single level-directory and each directory entry is of size 32-bits. → 4B . No. of blocks in disk = 2^{20}

1. The maximum number of files possible in this system is ?
2. The maximum size of any file is?

4 blocks ⇒ 8 k bytes used to store directory

max. no of files = max entries stored = $\frac{8kB}{4B} = 2k = 2048$

No. of blocks remaining for file $= (2^{20} - 4)$

max file size $= (2^{20} - 4) * 2kB$

Topic : File Allocation Method

↓

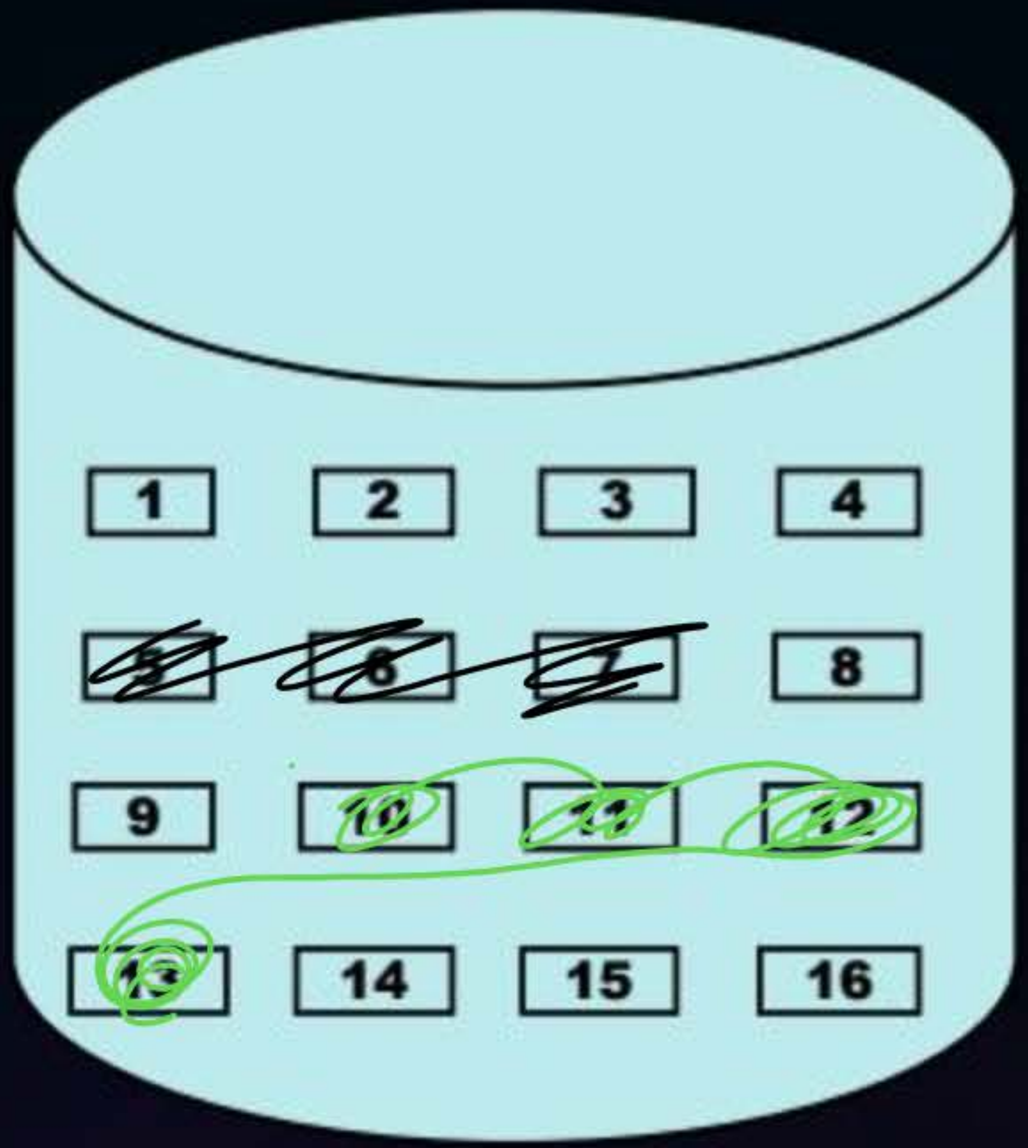
method to allocate disk blocks to store file.

Topic : File Allocation Method

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

Topic : Contiguous Allocation

↳ allocate consecutive blocks to a file.



Allocation table

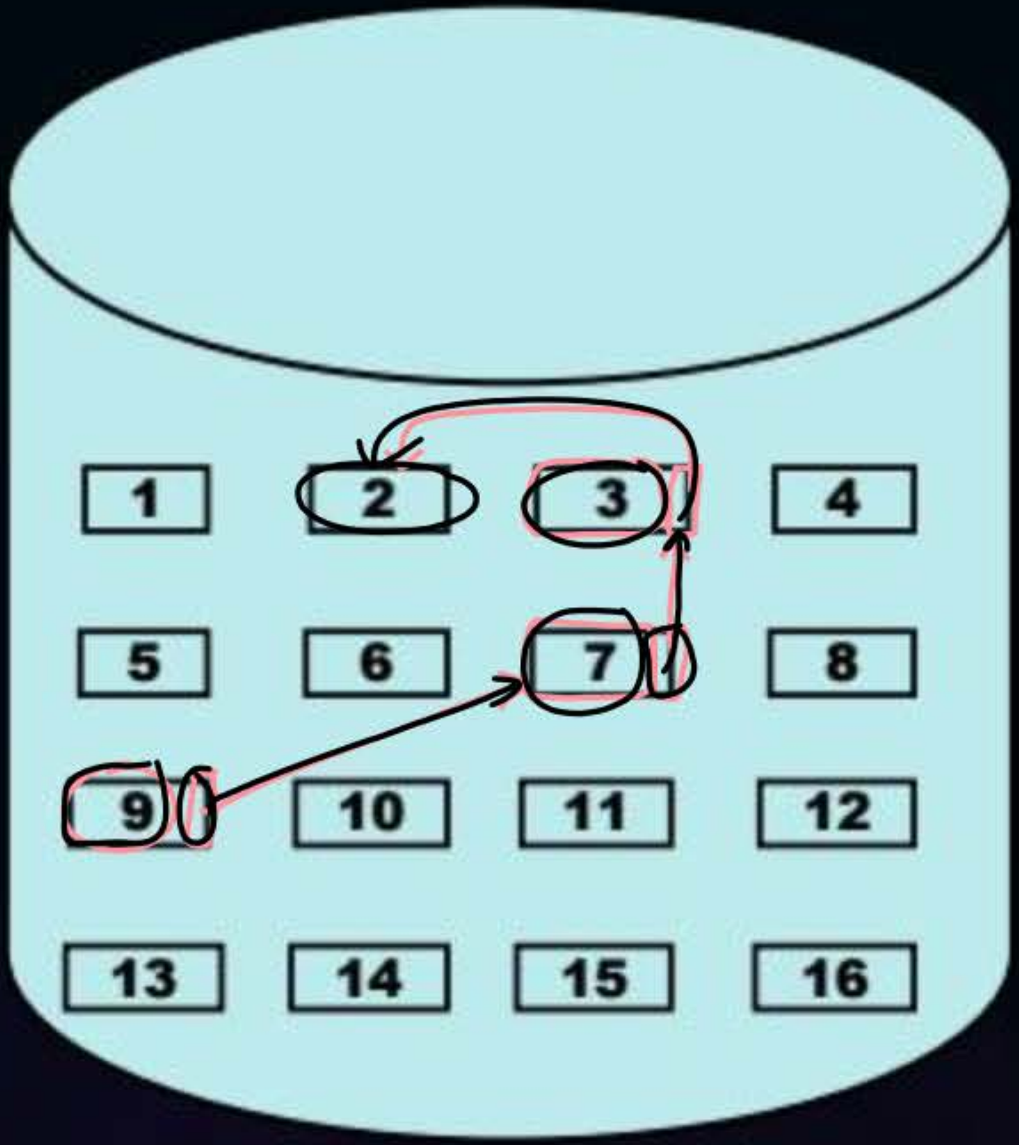
file name	file start block no.	no. of blocks to store file
COA. pptx	5	3
OSnotes.pdf	10	4

Topic : Contiguous Allocation

Performance:

1. Fragmentation: Internal, External ⇒ disadvantage
2. Increase in File size: Inflexible ⇒ disadvantage
3. Type of access: Sequential, Random/direct ⇒ advantage
4. Insertion in middle : Inflexible ⇒ Disadvantage

Topic : Linked Allocation



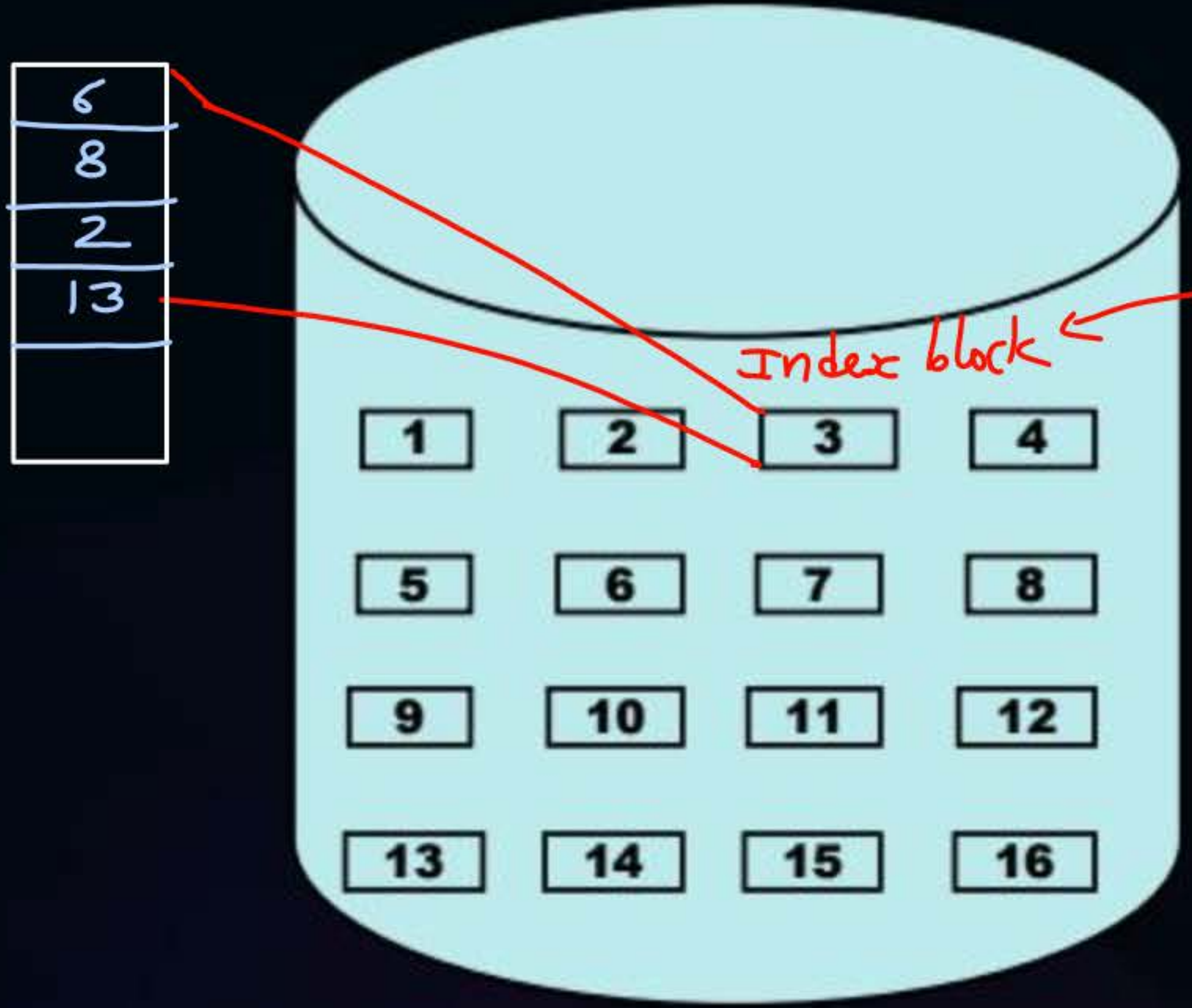
file name	start block no.	end block no.
osnotes.pdf	9	2

Topic : Linked Allocation

Performance:

1. Fragmentation: Internal ⇒ Advantage (no any external fragmentation)
2. Increase in File size: Flexible ⇒ Advantage
3. Type of access: Sequential ⇒ Disadvantage
4. Insertion in middle : Inflexible (because to reach to middle of the file then it will take time)

Topic : Indexed Allocation



file name	Index block
osnotes.pdf	3

Topic : Indexed Allocation

Performance:

1. Fragmentation: Internal
2. Increase in File size: Flexible
3. Type of access: Sequential, Random/direct
4. Insertion in middle: Flexible

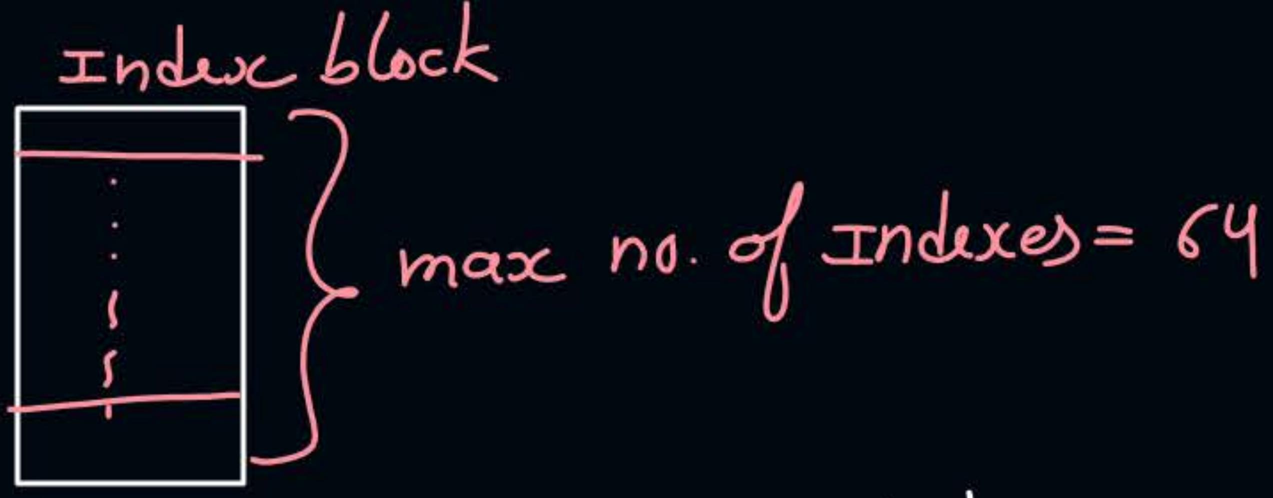
only disk block addresses are moved in index block.

Disadvantage :-

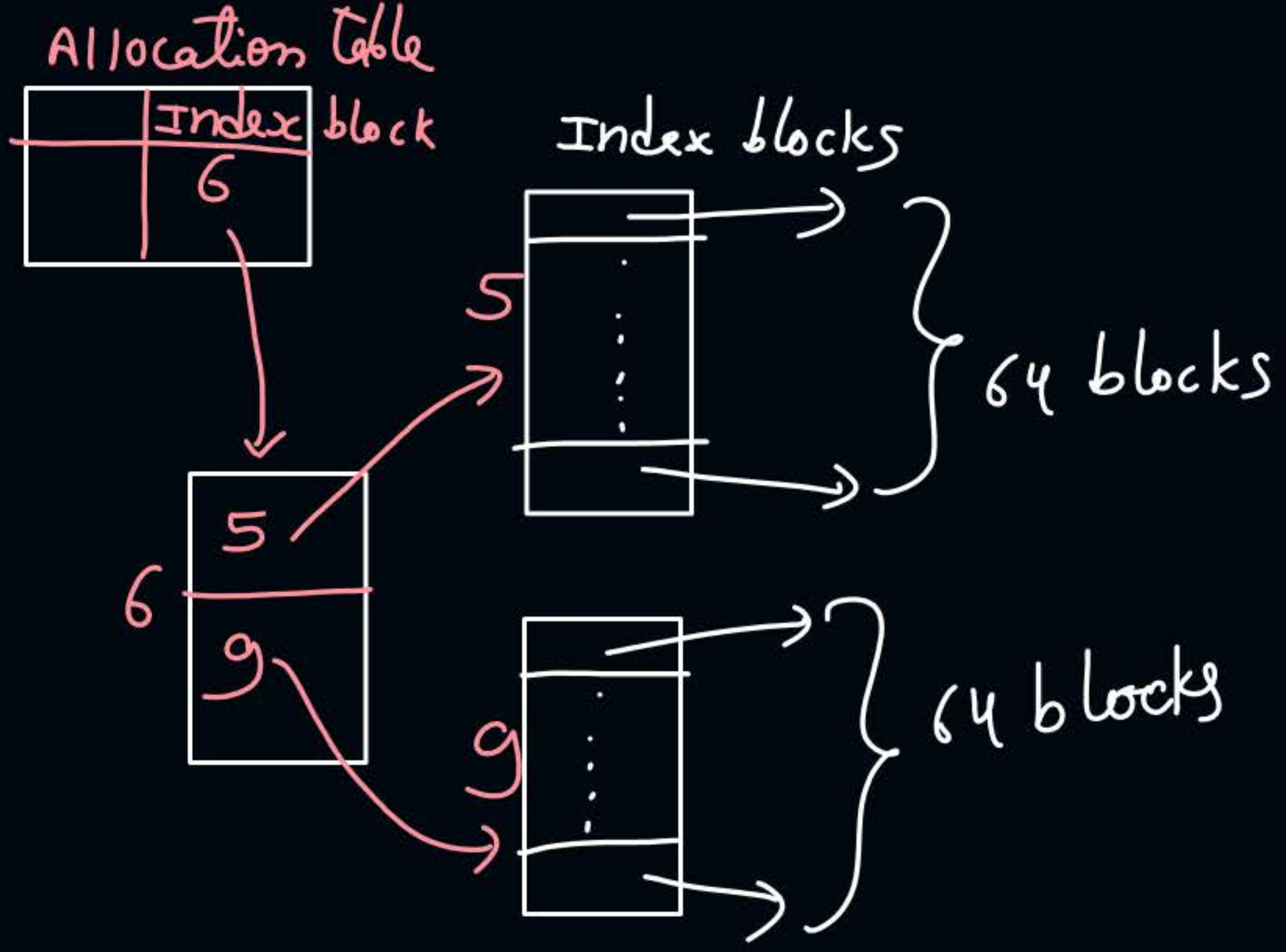
space is occupied for storing index.

⇒ If one block size is not sufficient to store indexes of a file then multilevel indexing is used.

ex:-



but file size = 128 blocks



Topic : Question

- Disk block address = 16 bits $= 2B$
- Disk block size = 1KB
- Index block = 1KB
- Maximum file size?

single level Indexing ✓

no. of indexes per block

$$= \frac{1kB}{2B}$$

$$= 2^9 = 512$$

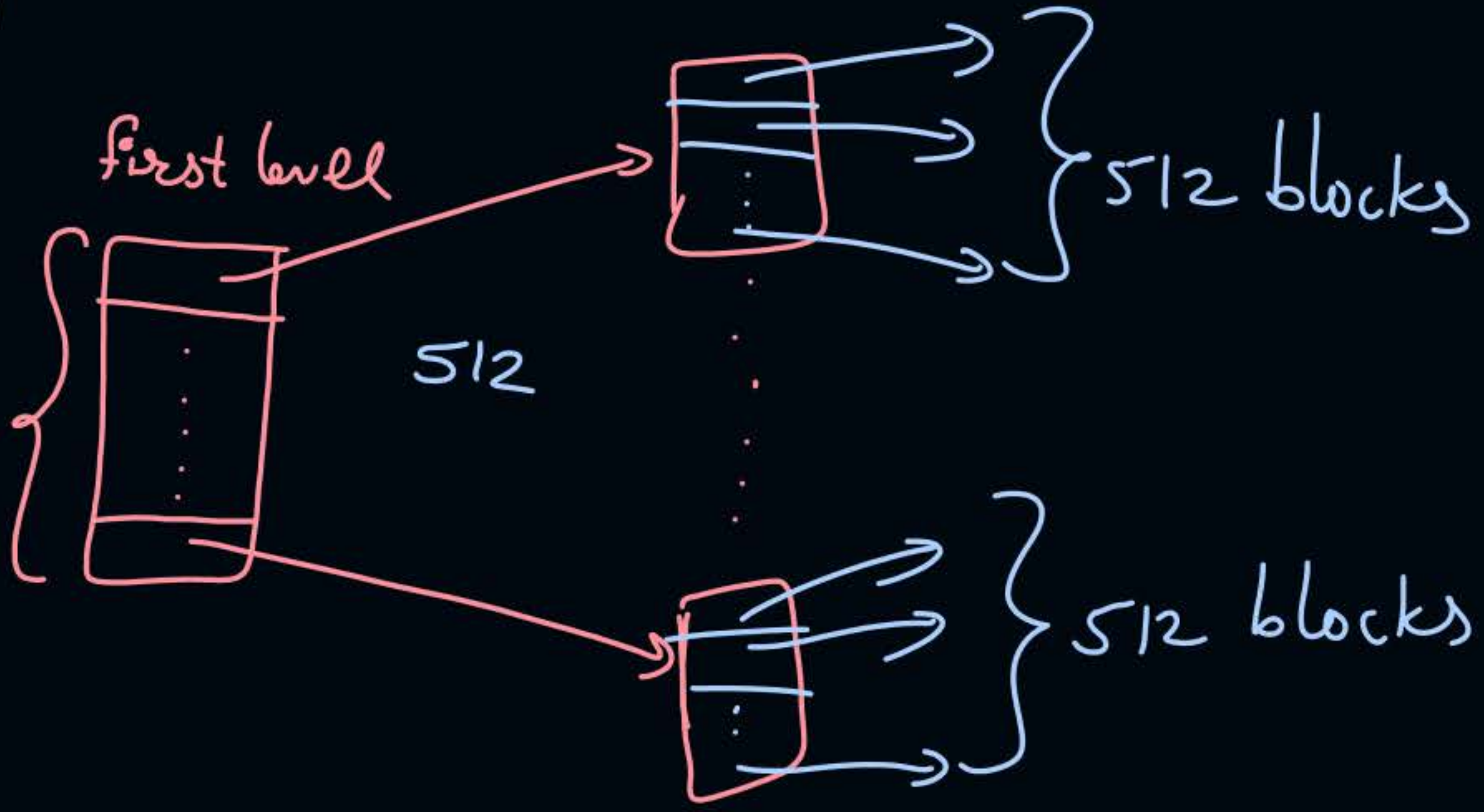
max file size $= 512 * 1KB$

$= 512KB$

- Disk block add. = 2B (16 bits)
- Disk block size = 1KB
- Index —"— = 1KB
- 2-level Indexing
- max file size = ______ ?

no. of indexes per block = $\frac{1kB}{2k}$

= 512



max no. of blocks used to store file

= 512 * 512

max file size = $2^9 * 2^9 * 1kB$

= 256 MB

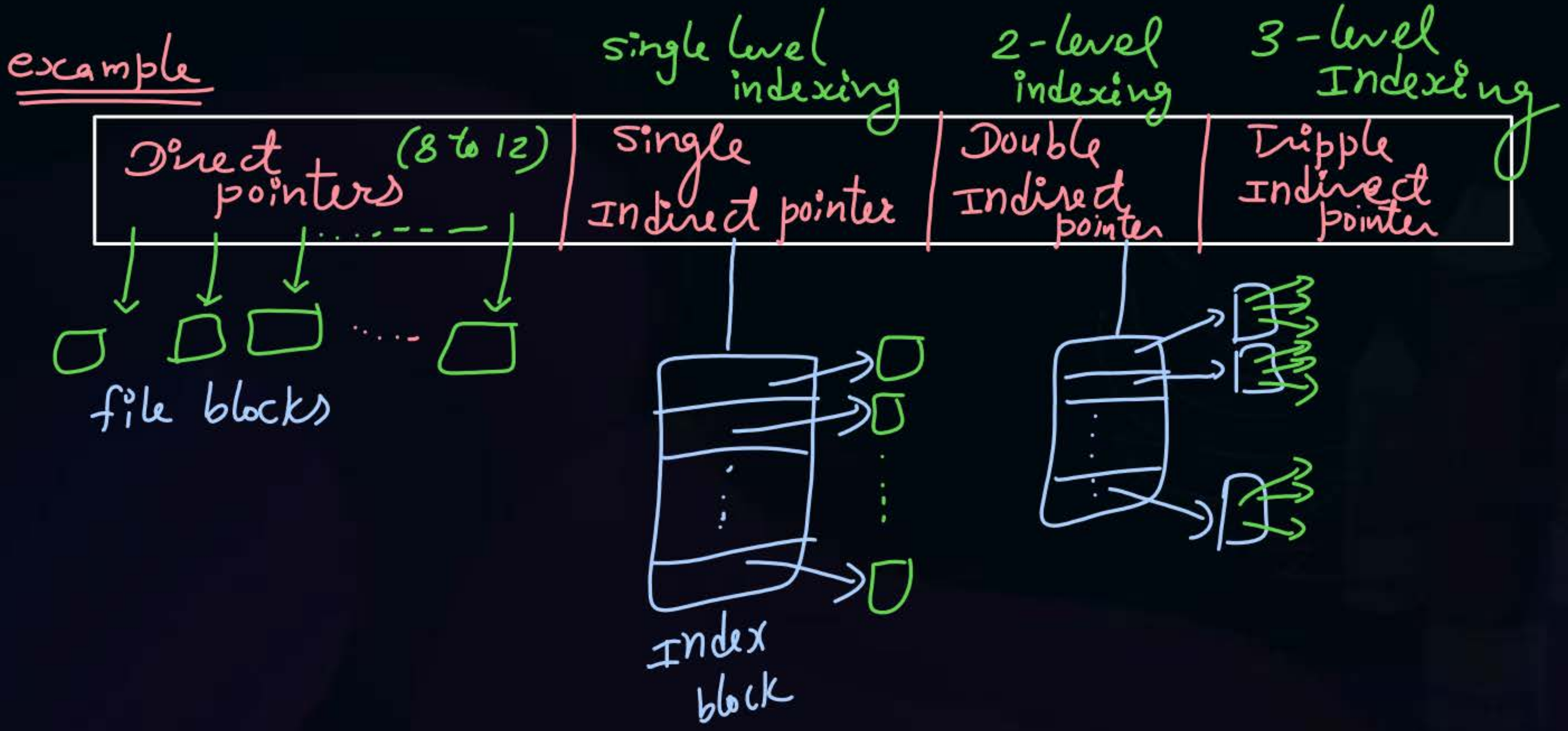
Topic : Master Boot Record

A master boot record (MBR) is a special type of boot sector at the very beginning of partitioned computer mass storage devices.

- Contains the information regarding how and where the OS is located in the hard disk so that it can be booted in the RAM.

Topic : Unix I-node Structure

The inode (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory.



Topic : Unix I-node Structure

The inode (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory.

- Each inode stores the attributes and disk block locations of the object's data.
- The number of Inode limits the total number of files/directories that can be stored in the file system.

Topic : Question

[GATE-2019]

4bytes

#Q. The index node (inode) of a Unix-like file system has 12 direct, one single-indirect and one double-indirect pointer. The disk block size is 4 kB, and the disk block addresses 32-bits long. The maximum possible file size is (rounded off to 1 decimal place) ___4.0___ GB?

no. of indexes per block $= \frac{4kB}{4B} = 1k = 2^{10}$

$$\text{max file size} = (12 * 4kB) + (2^{10} * 4kB) + (2^{10} * 2^{10} * 4kB)$$

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

$$\cong 4.004048\,GB = 4.0\,GB$$

Topic : Question

[GATE-2005]

#Q. In a computer system, four files of size 1 1050 bytes, 4990 bytes, 5170 bytes and 12640 bytes need to be stored. For storing these files on disk, we can use either 100 byte disk blocks or 200 byte disk blocks (but can't mix block sizes). For each block used to store a file, 4 bytes of bookkeeping information also needs to be stored on the disk. Thus, the total space used to store a file is the sum of the space taken to store the file and the space taken to store the book keeping information for the blocks allocated for storing the file. A disk block can store either bookkeeping information for a file or data from a file, but not both.

What is the total space required for storing the files using 100 byte disk blocks and 200 byte disk blocks respectively?

A 35400 and 35800 bytes

B 35800 and 35400 bytes

C 35600 and 35400 bytes

D 35400 and 35600 bytes

Topic : Question

Ans = 99.6

[GATE-2014]

#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 10^3 bytes, the maximum size of a file that can be stored on this disk in units of 10^6 bytes is.

no. of blocks in disk $= \dfrac{100 * 10^6 \text{ B}}{10^3 \text{ B}} = 10^5$ blocks

Total size of FAT entries $= 10^5 * 4\text{B} = 4 * 10^5 \text{ B}$

No. of blocks needed to store FAT entries $= \dfrac{4 * 10^5 \text{B}}{10^3} = 400$ blocks

$$\text{Max file size} = (10^5 - 400) * 10^3\ B$$

$$= (100 - 0.4) * 10^6\ B$$

$$= 99.6 * 10^6\ B$$

Ans = 99.6

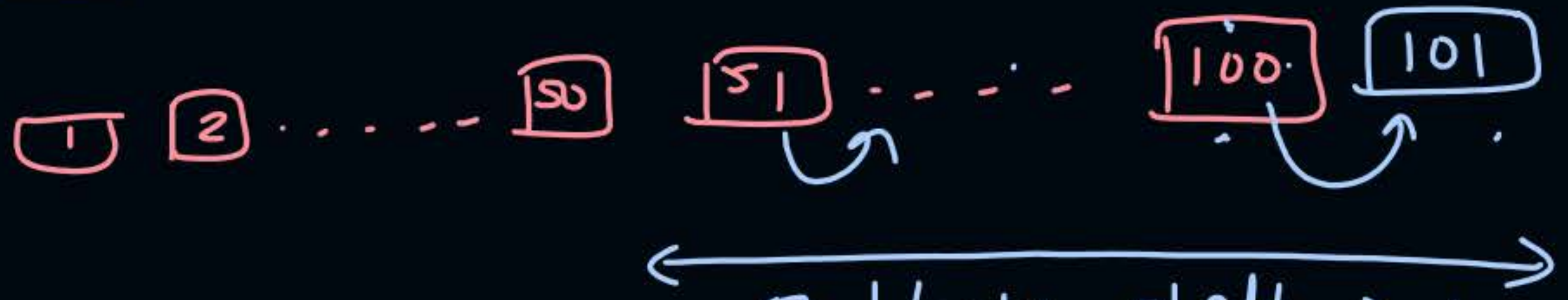
Topic : Question

[GATE-2022]

#Q. Consider two files systems A and B , that use contiguous allocation and linked allocation, respectively. A file of size 100 blocks is already stored in A and also in B. Now, consider inserting a new block in the middle of the file (between 50th and 51st block), whose data is already available in the memory. Assume that there are enough free blocks at the end of the file and that the file control blocks are already in memory. Let the number of disk accesses required to insert a block in the middle of the file in A and B are n_A and n_B respectively, then the value of $n_A + n_B$ is_______?

$n_A + n_B = 101 + 52 = \underline{\underline{153}}$ Ans.

Contiguous :-

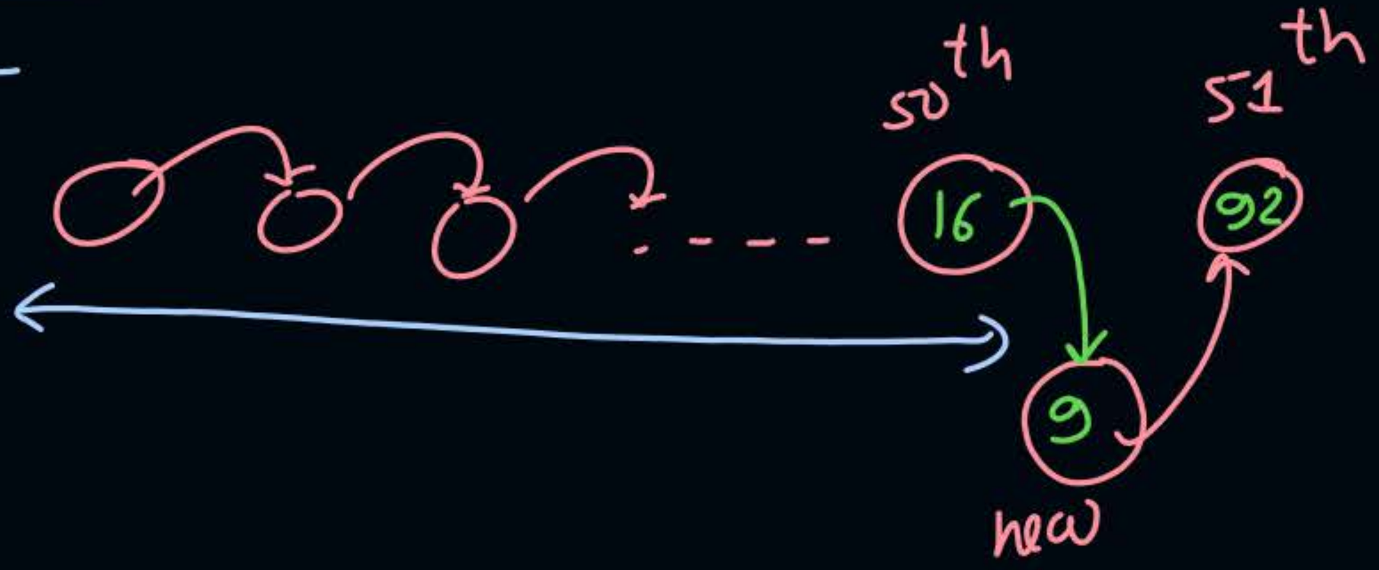


50 blocks shift ⇒ 100 block access

1 new block insert^n ⇒ 1 block access

(n_A) Total = 101

Linked :-



new block inst^n ⇒ 1

to reach to 50th block ⇒ 50

update pointer of 50th ⇒ 1

n_B 52

2 mins Summary

Topic File Allocation Method

Topic Unix i-node

Topic Disk Cylinder, Seek Time

Happy Learning

THANK - YOU