OPERATING SYSTEMS

Storage Management - 9

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Storage Management - 9: Allocation Methods

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 4

Unit 4: Storage Management

Mass-Storage Structur - Mass-Storage overview, Disk Scheduling, Swap-Space Management, RAID structure. File System Interface - file organization/structure and access methods, directories, sharing File System Implementation/Internals: File control Block (inode), partitions & mounting, Allocation methods.

Case Study: Linux/Windows File Systems

OPERATING SYSTEMS

Course Outline

37	Mass-Storage Structure: Mass-Storage overview	12.1	82.1
38	Disk Scheduling – FCFS, SSTF, SCAN, C-SCAN, LOOK	12.4	
39	Swap-Space Management, RAID Structure	12.6,12.7	
40	File Concept, File Structure, Access Methods	10.1-10.2	
41	Directory and Disk Structure	10.3	
42	File-System Mounting, File Sharing, Protecting	10.4-10.6	
43	Implementing File-Systems: File control Block (inode), partitions & mounting	11.1,11.2	
44	Disk Space Allocation methods: Contiguous, Linked, Indexed	11.4	
45	Case Study: Unix/Linux File systems	11.8	
46	NFS	16.7	

Topic Outline

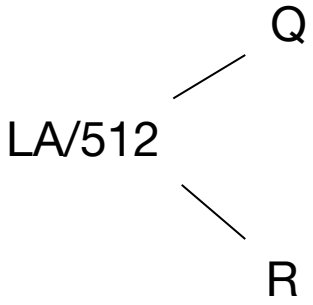
- Contiguous Allocation
- Linked allocation
- Indexed Allocation

Contiguous Allocation

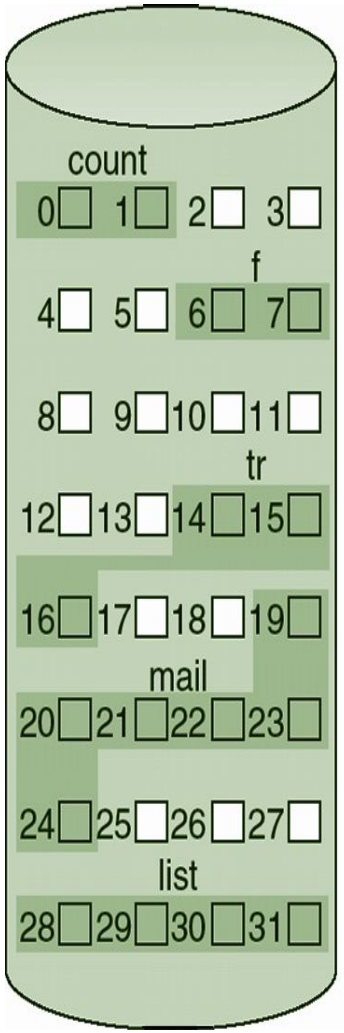
- An allocation method refers to how disk blocks are allocated for files
- Contiguous allocation – each file occupies set of contiguous blocks
  - Best performance in most cases
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Problems include finding space for file, knowing file size, external fragmentation, need for compaction off-line (downtime) or on-line

Contiguous Allocation

- Mapping from logical to physical



- Block to be accessed = Q + starting address
  Displacement into block = R

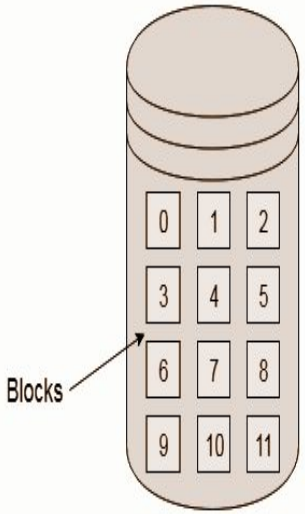


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous Allocation

- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents



File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Directory

Contiguous Allocation

Contiguous Allocation

Advantages:

1. In the contiguous allocation, sequential and direct access both are supported.
2. For the direct access, the starting address of the kth block is given and further blocks are obtained by b+K,
3. This is very fast and the number of seeks is minimal in the contiguous allocation method.

Disadvantages:

1. Contiguous allocation method suffers internal as well as external fragmentation.
2. In terms of memory utilization, this method is inefficient.
3. It is difficult to increase the file size because it depends on the availability of contiguous memory.

Linked Allocation

- Linked allocation – each file a linked list of blocks
  - File ends at nil pointer
  - No external fragmentation
  - Each block contains pointer to next block
  - No compaction, external fragmentation
  - Free space management system called when new block needed
  - Improve efficiency by clustering blocks into groups but increases internal fragmentation
  - Reliability can be a problem
  - Locating a block can take many I/Os and disk seeks

Linked Allocation

- In this scheme, the directory entry contains the pointer of the first block and pointer of the ending block.
- These pointers are not for the users.
- For example, a file of six blocks starts at block 10 and end at the block. Each pointer contains the address of the next block.
- When we create a new file we simply create a new entry with the linked allocation.
- Each directory contains the pointer to the first disk block of the file. when the pointer is nil then it defines the empty file.

Linked Allocation

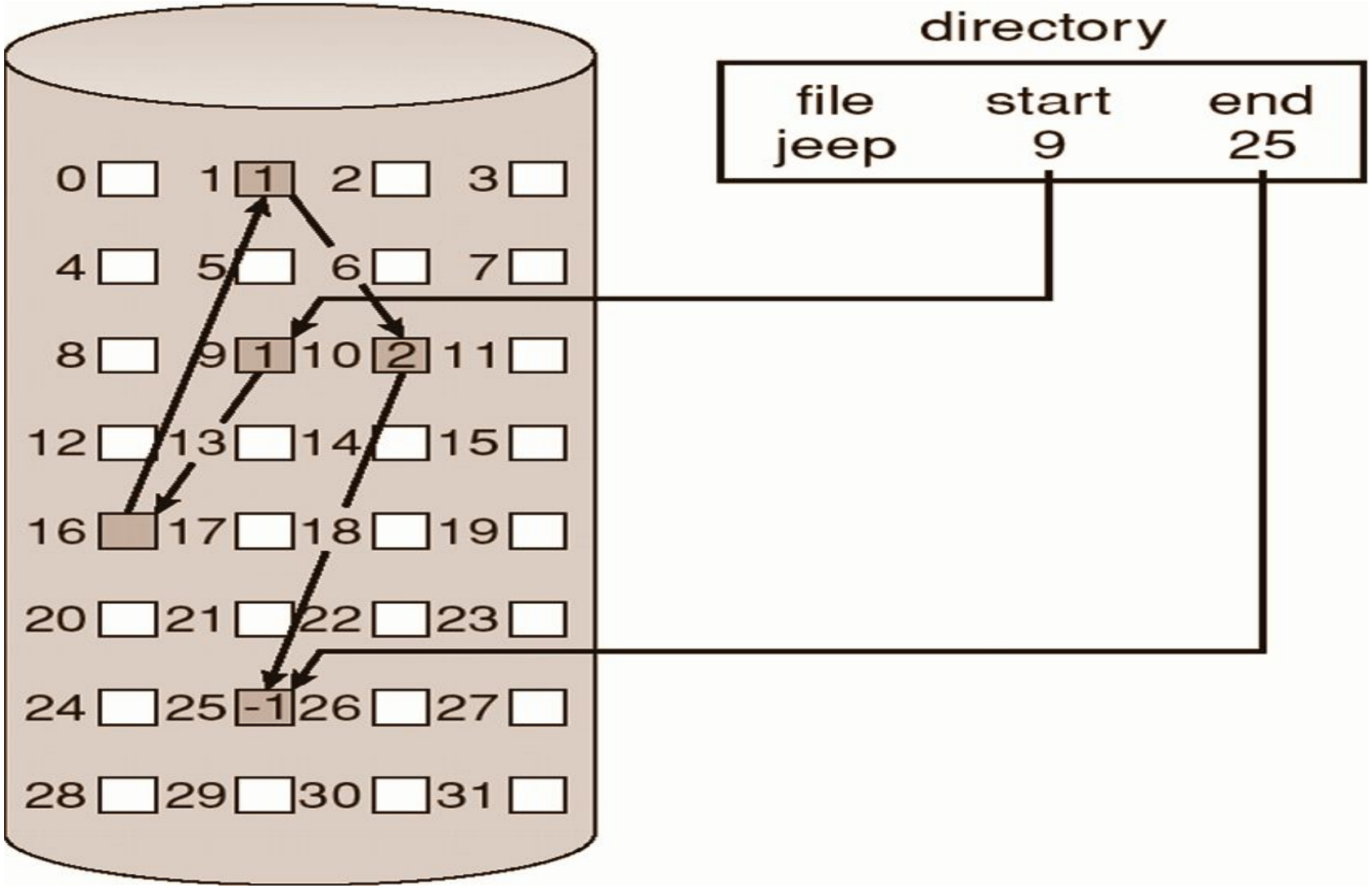
Advantages:

1. In terms of the file size, this scheme is very flexible.
2. We can easily increase or decrease the file size and system does not worry about the contiguous chunks of memory.
3. This method free from external fragmentation this makes it better in terms of memory utilization.

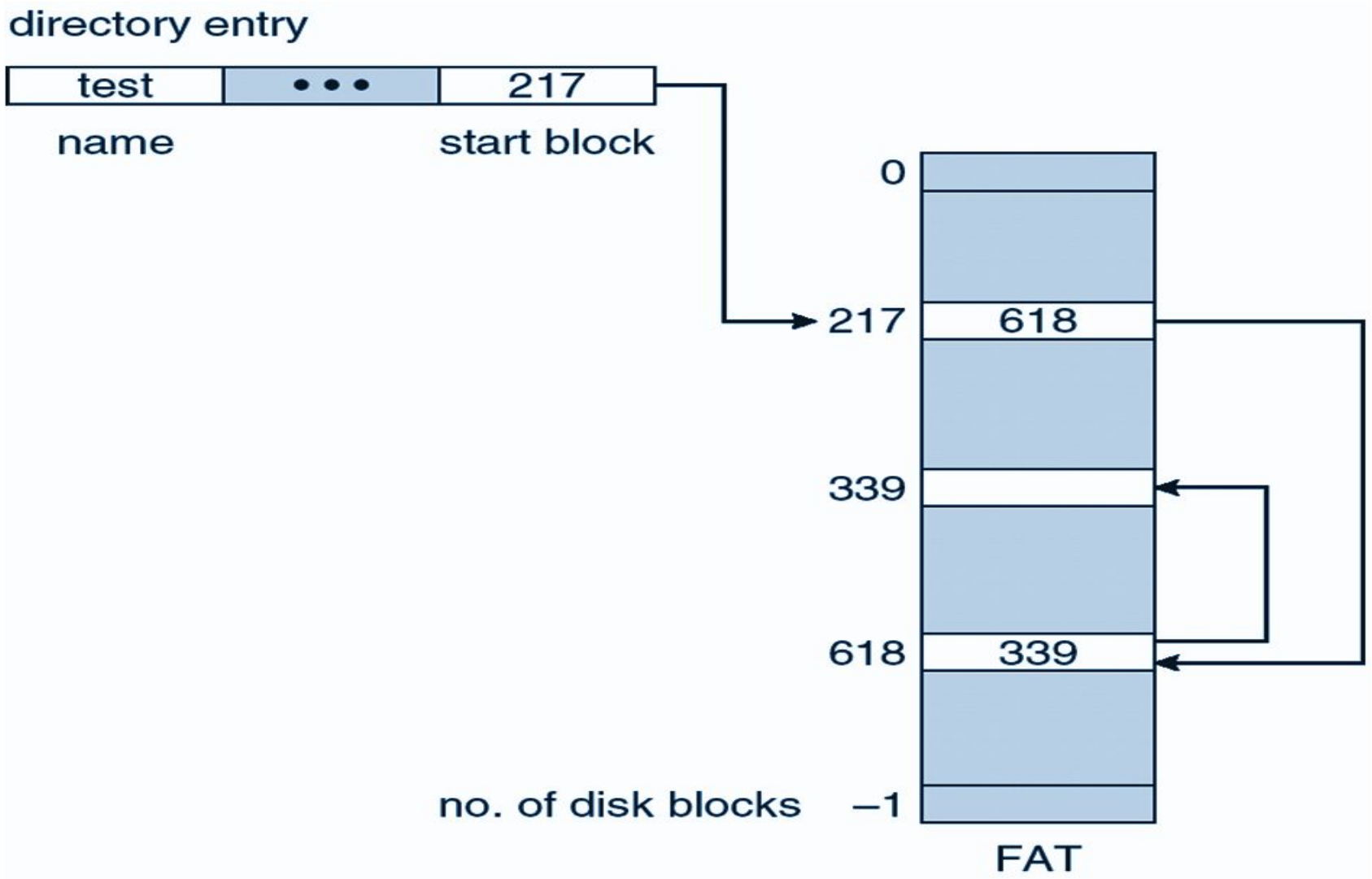
Disadvantages:

1. In this scheme, there is large no of seeks because the file blocks are randomly distributed on disk.
2. Linked allocation is comparatively slower than contiguous allocation.
3. Random or direct access is not supported by this scheme we cannot access the blocks directly.
4. The pointer is extra overhead on the system due to the linked list.

Linked Allocation

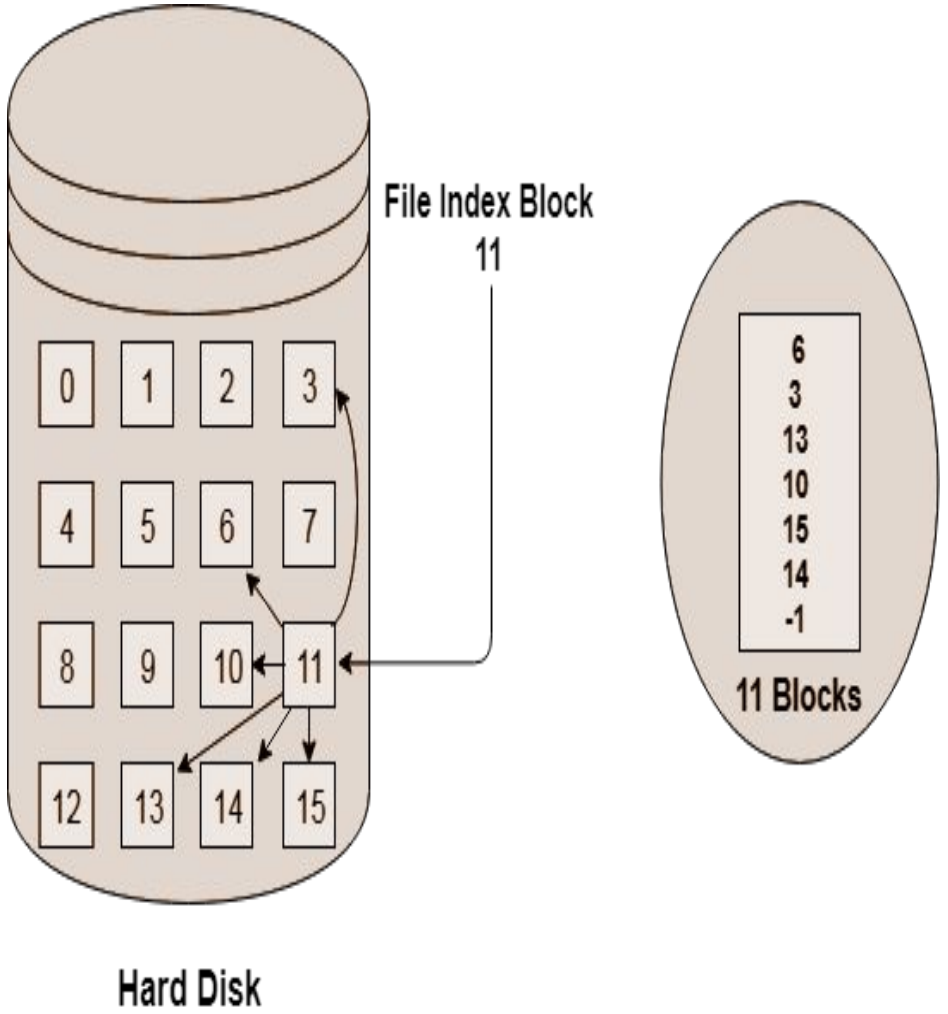


Linked Allocation File Allocation Table



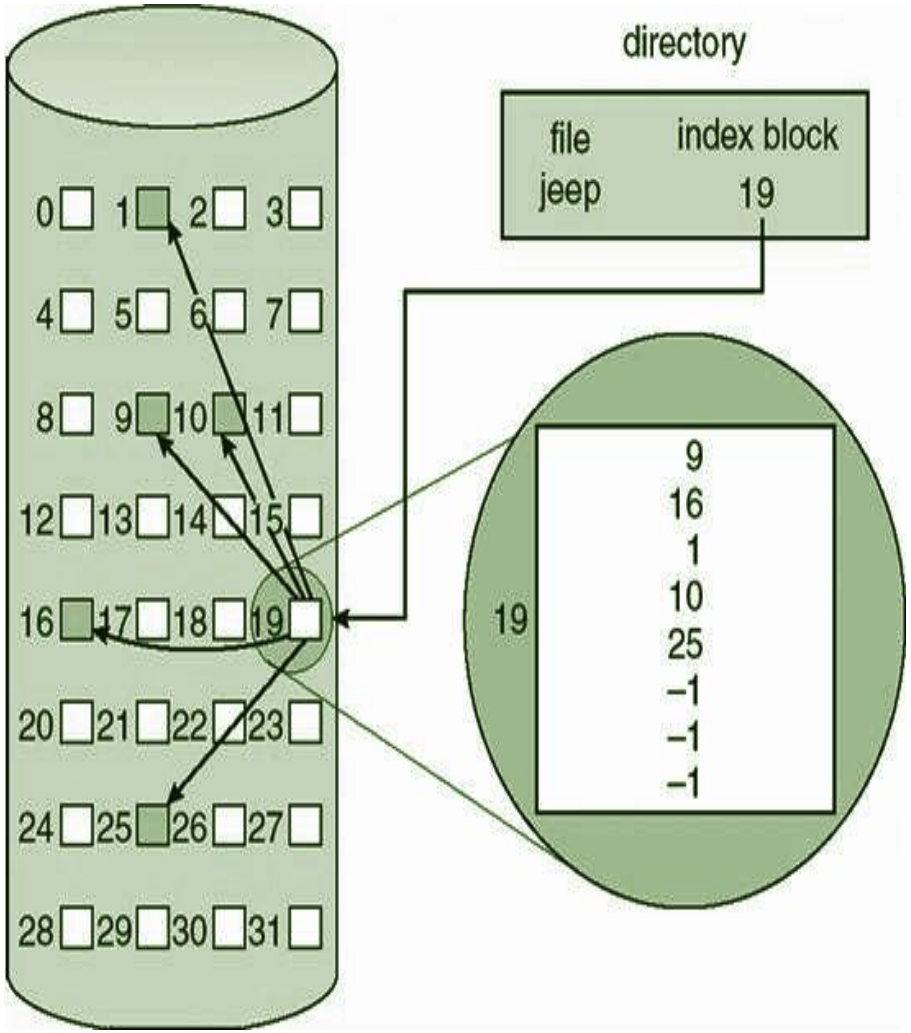
Indexed Allocation

- Instead of maintaining a file allocation table of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block.
- Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file.
- Directory entry will only contain the index block address.



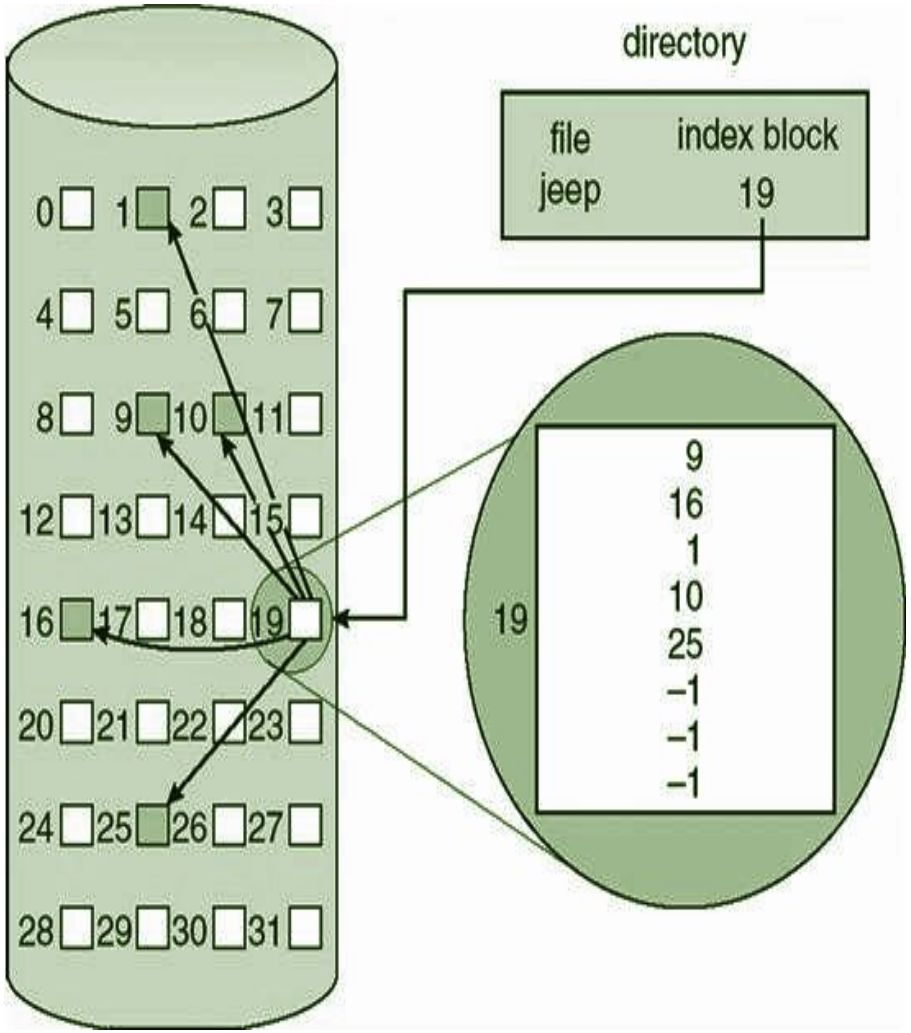
Indexed Allocation

- In Linked Allocation the pointers along with the blocks were scattered across the disk and needed to be retrieved in order by visiting each block for accessing the file.
- In indexed allocation method, all the pointers are gathered together into one location known as Index Block.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.



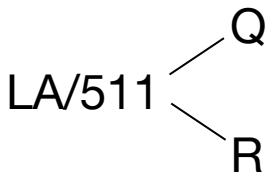
Indexed Allocation

- When a file is created initially, all pointers in the index block are set to null value.
- As new blocks are written, the pointers are modified accordingly.



Indexed Allocation

- Indexed allocation supports direct access and does not suffer from any external fragmentation.
- Indexed allocation suffers from the problem of wasted space.
- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block
- Mapping from logical to physical in a file of maximum size of 256K bytes and block size of 512 bytes. We need only 1 block for index table



Q = displacement into index table
R = displacement into block

Indexed Allocation

Advantages

1. Supports direct access
2. A bad data block causes the loss of only that block.

Disadvantages

1. A bad index block could cause the loss of entire file.
2. Size of a file depends upon the number of pointers, a index block can hold.
3. Having an index block for a small file is totally wastage.
4. More pointer overhead

Indexed Allocation

- Indexed allocation supports direct access and does not suffer from any external fragmentation

- Indexed allocation suffers from the problem of wasted space.

- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

Indexed Allocation

- For files that are very large, single index block may not be able to hold all the pointers.

- Following mechanisms can be used to resolve this:

  - **Linked scheme:** This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.

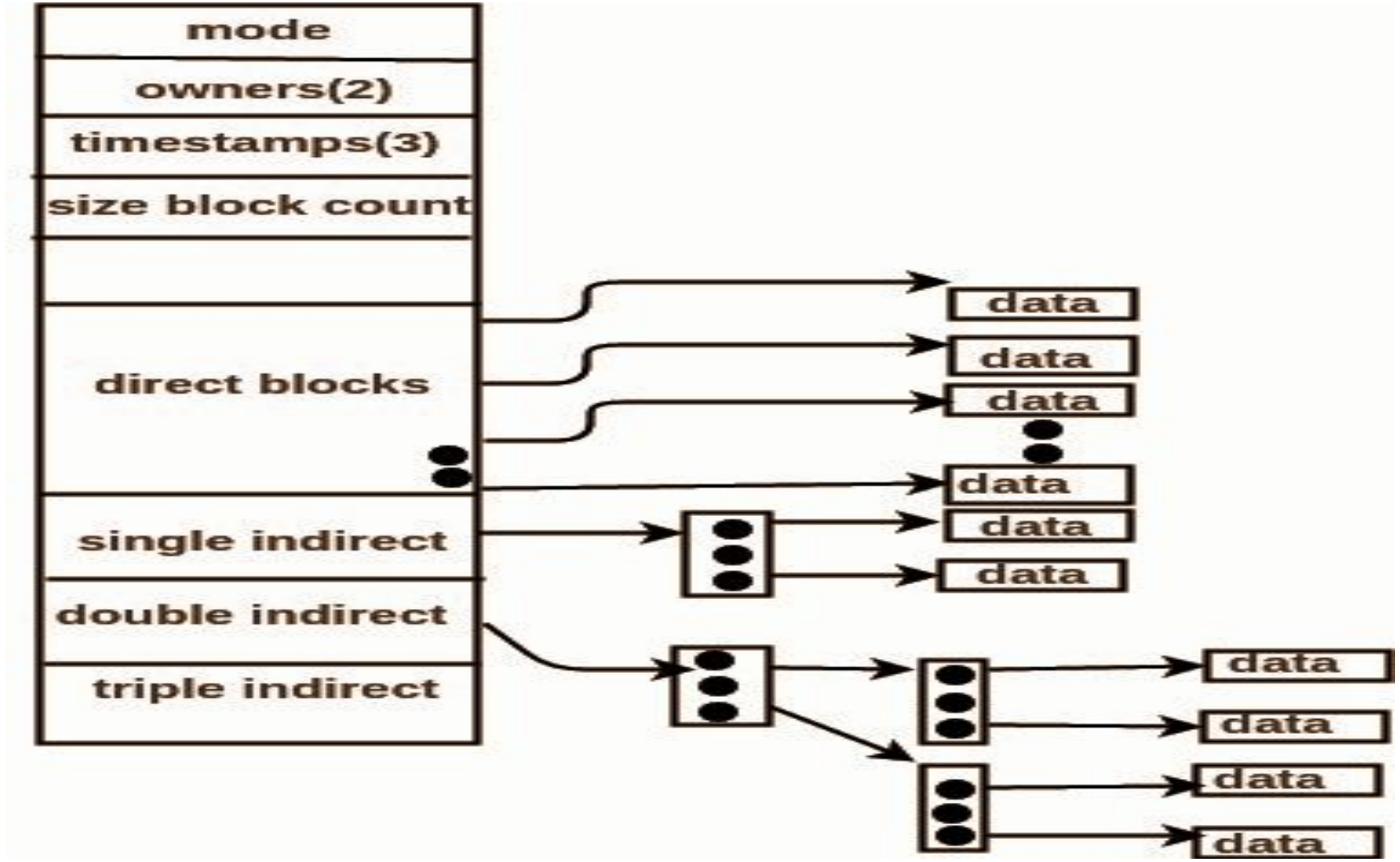
  - **Multilevel index:** In this policy, a first level index block is used to point to the second level index blocks which in turn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.

Indexed Allocation

- For files that are very large, single index block may not be able to hold all the pointers.
- Following mechanisms can be used to resolve this:
  - **Combined Scheme:** In this scheme, a special block called the **inode** (information Node) contains all the information about the file such as the name, size, authority, etc and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file as shown in the image below.
    - The first few of these pointers in Inode point to the direct blocks i.e the pointers contain the addresses of the disk blocks that contain data of the file.
    - The next few pointers point to indirect blocks. Indirect blocks may be single indirect, double indirect or triple indirect. Single Indirect block is the disk block that does not contain the file data but the disk address of the blocks that contain the file data.
    - Similarly, double indirect blocks do not contain the file data but the disk address of the blocks that contain the address of the blocks containing the file data.

Indexed Allocation

4K bytes per block, 32-bit addresses



Topics Uncovered in this Session

- Contiguous Allocation
- Linked allocation
- Indexed Allocation

THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com