OPERATING SYSTEMS

Storage Management

Venkatesh Prasad

Department of Computer Science

OPERATING SYSTEMS

Implementing File-Systems – Disk Space Allocation Methods

Venkatesh Prasad

Department of Computer Science

OPERATING SYSTEMS
## Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:

1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th edition 2018
3. Some presentation transcripts from A. Frank – P. Weisberg
4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

Allocation Methods - Contiguous

■ 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

- Supports both sequential and direct access

- Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line** (**downtime**) or **on-line**
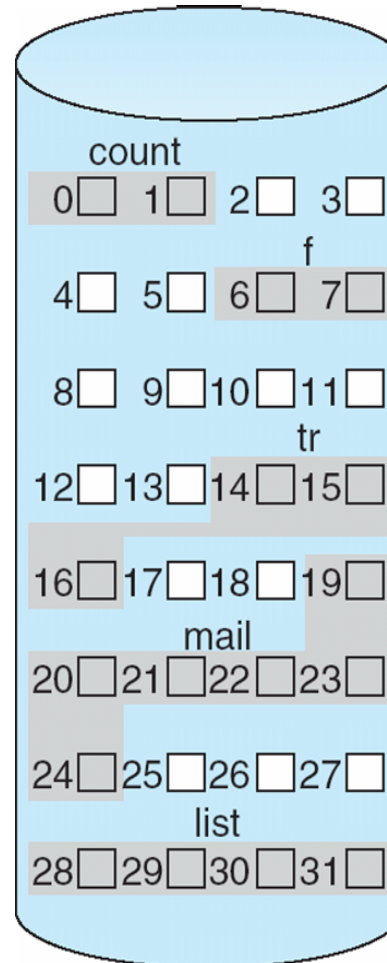
Contiguous Allocation

■ Mapping from logical to physical

directory

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

LA/512

Q

R
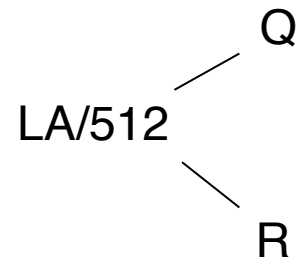
- LA – Length of the area allocated for this file
- Q = displacement into index table
- Block to be accessed = Q + starting address
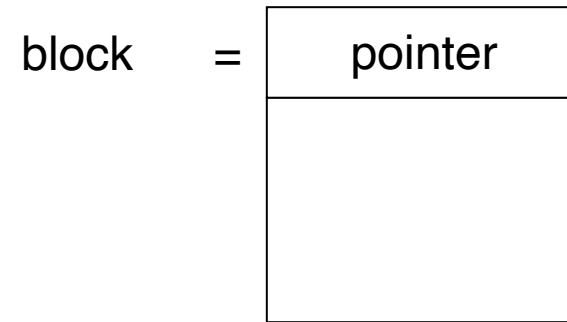- Displacement into block = R

Extent-Based Systems

■ 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

Allocation Methods - Linked

■ **Linked allocation** – each file a linked list of blocks

● File ends at null 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
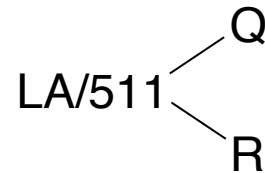
Allocation Methods – Linked (Cont.)

■ FAT (File Allocation Table) variation

- Beginning of volume has table, indexed by block number

- Much like a linked list, but faster on disk and cacheable
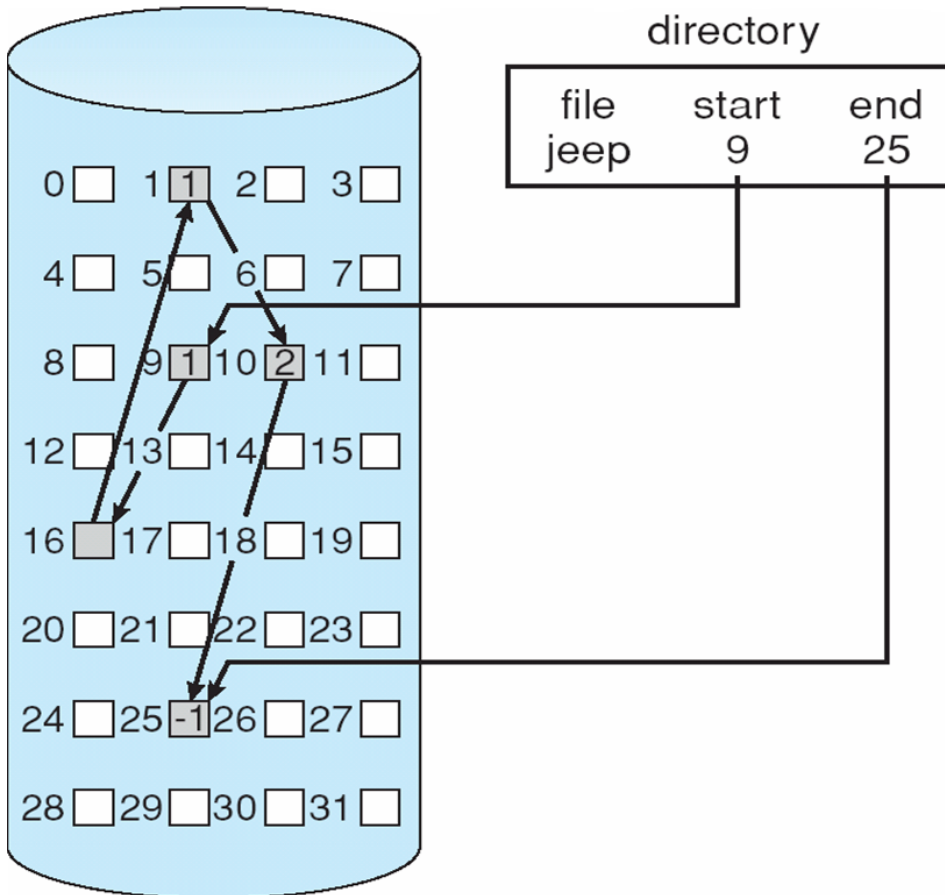
- New block allocation simple

Linked Allocation

■ Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk

block   = | pointer |
                | |
                | |

■ Mapping
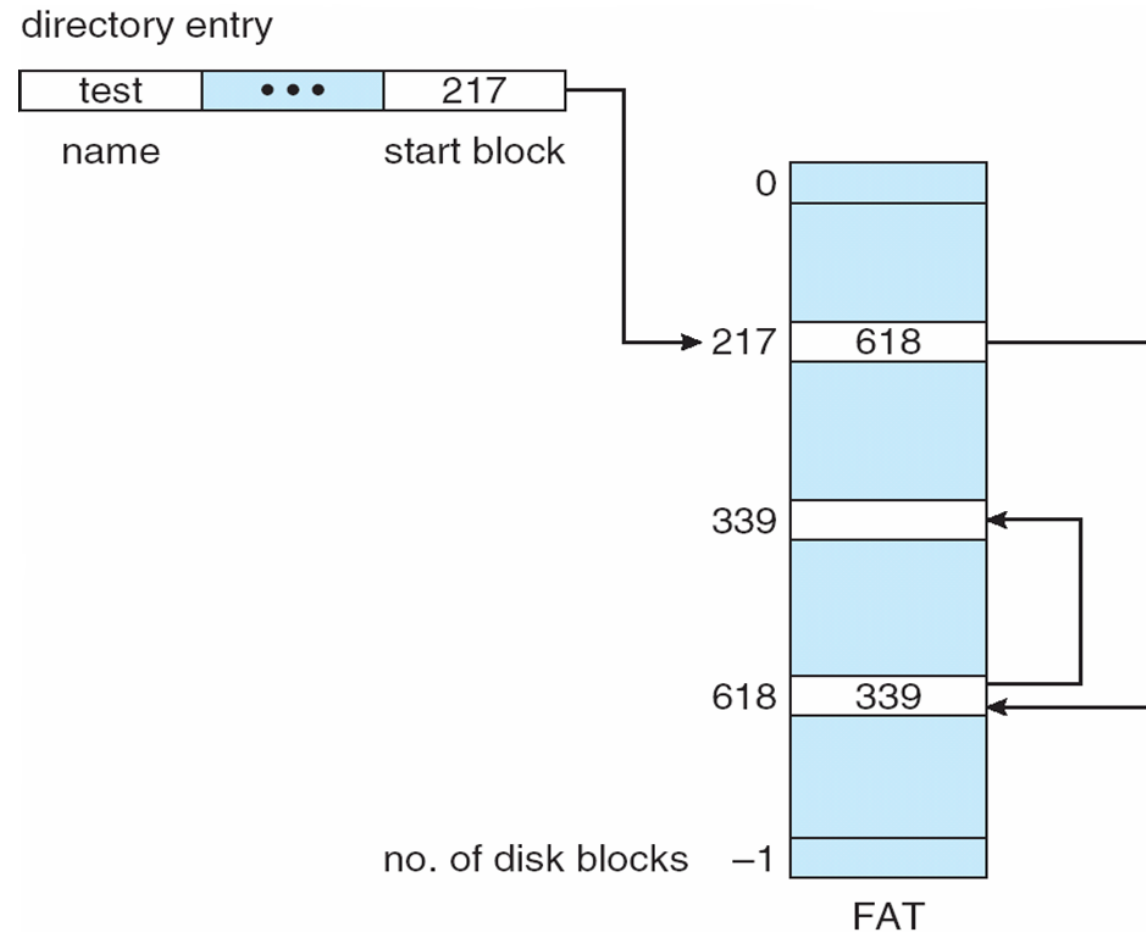
                        Q
                      ╱
          LA/511
                      ╲
                        R

Block to be accessed is the Qth block in the linked chain of blocks representing the file.

Linked Allocation

directory

file	start	end
jeep	9	25

- Each block contains a pointer to the next block.
- If each block is 512 bytes and a disk address (pointer) requires 4 bytes then the user sees blocks of 508 bytes (i.e. some disk space is wasted)
- Collect blocks into multiples called **clusters** and allocate clusters rather than blocks

File-Allocation Table (variation on linked allocation)

•Unused block is indicated by a table value of 0

•To allocate a new block to a file, find the first 0-valued table entry and replace the previous EOF value with the address of the new block.

•Then replace 0 with EOF value

directory entry

test	• • •	217

name   start block

0	
217	618
339	
618	339
−1	

no. of disk blocks

FAT

Allocation Methods - Indexed

■ **Indexed allocation**

● Each file has its own **index block**(s) of pointers to its data blocks

■ Logical view

index table

Example of Indexed Allocation
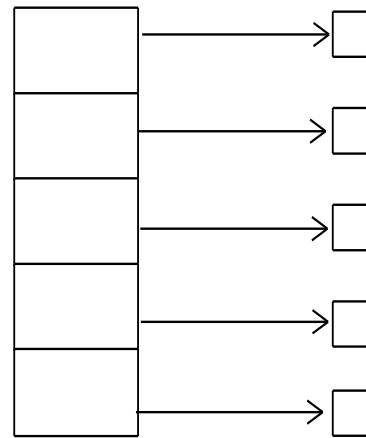
- In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file.
- Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block.
- The directory entry contains the address of the index block as shown in the figure.

Allocation Methods – Indexed (Cont.)
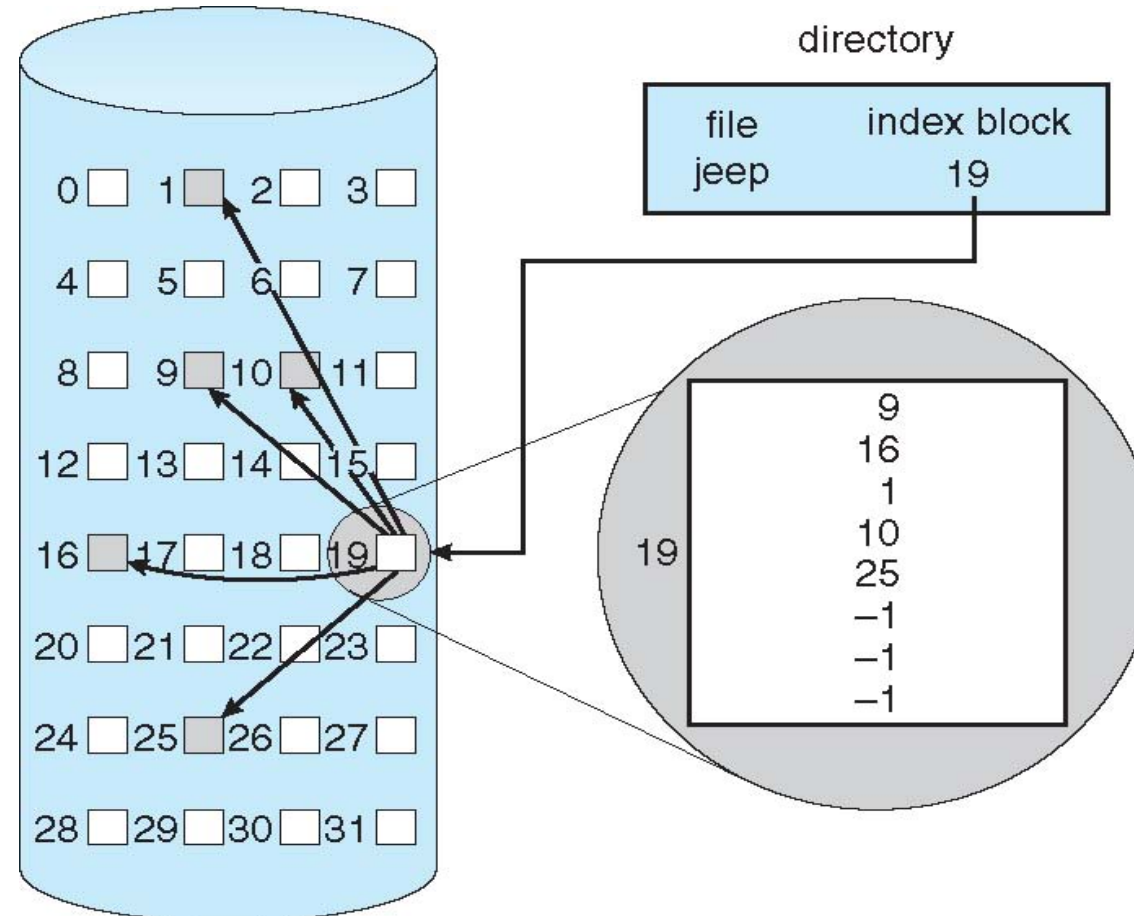
Advantages:
➤ This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
➤ It overcomes the problem of external fragmentation.

Disadvantages:
➤ The pointer overhead for indexed allocation is greater than linked allocation.
➤ 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. (Note: In linked allocation we lose the space of only 1 pointer per block.)

Indexed Allocation – Mapping (Cont.)

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

➢ Other Schemes such as Linked scheme, Multilevel index and Combined Scheme are used.

outer-index

index table

file

Combined Scheme: UNIX UFS
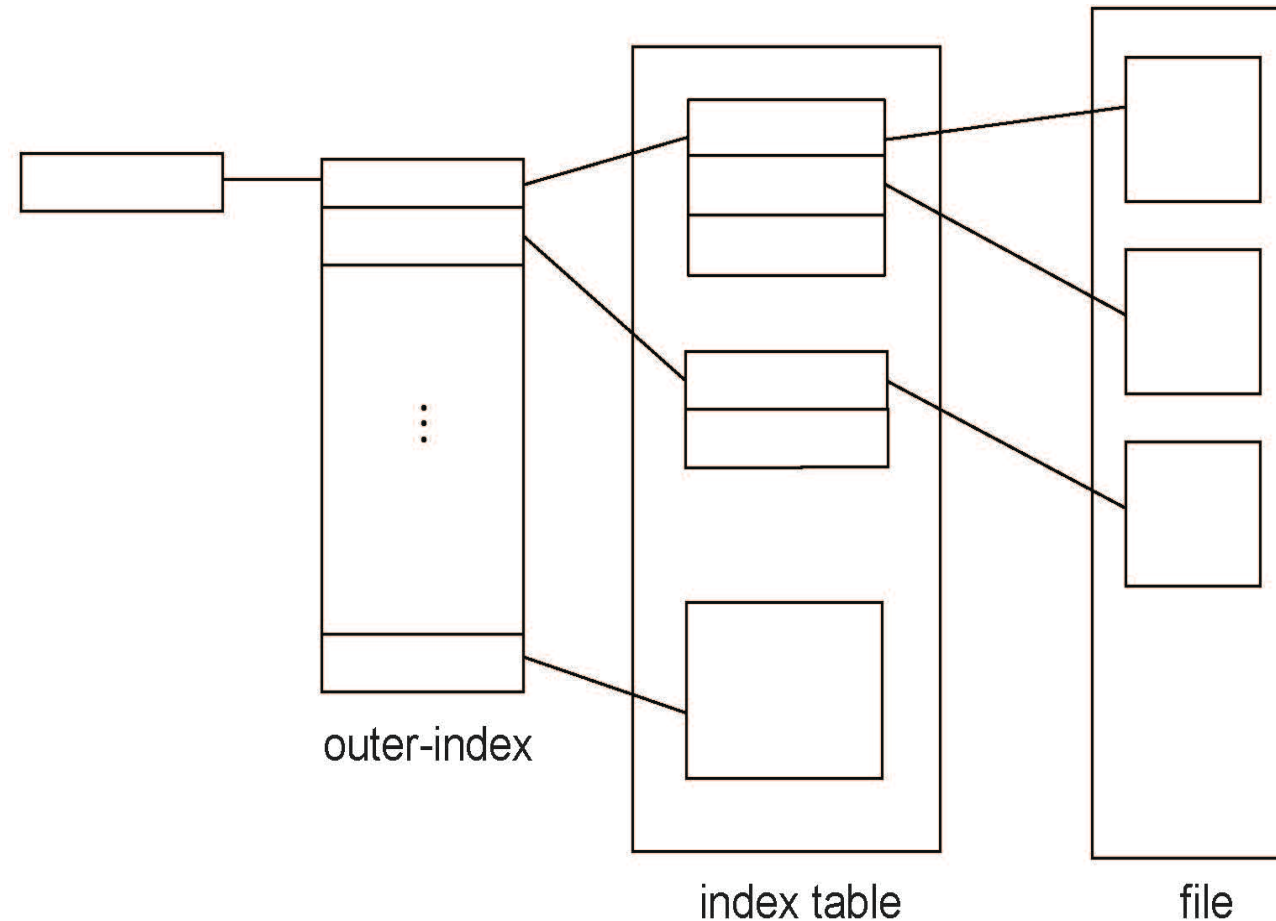
4K bytes per block, 32-bit addresses
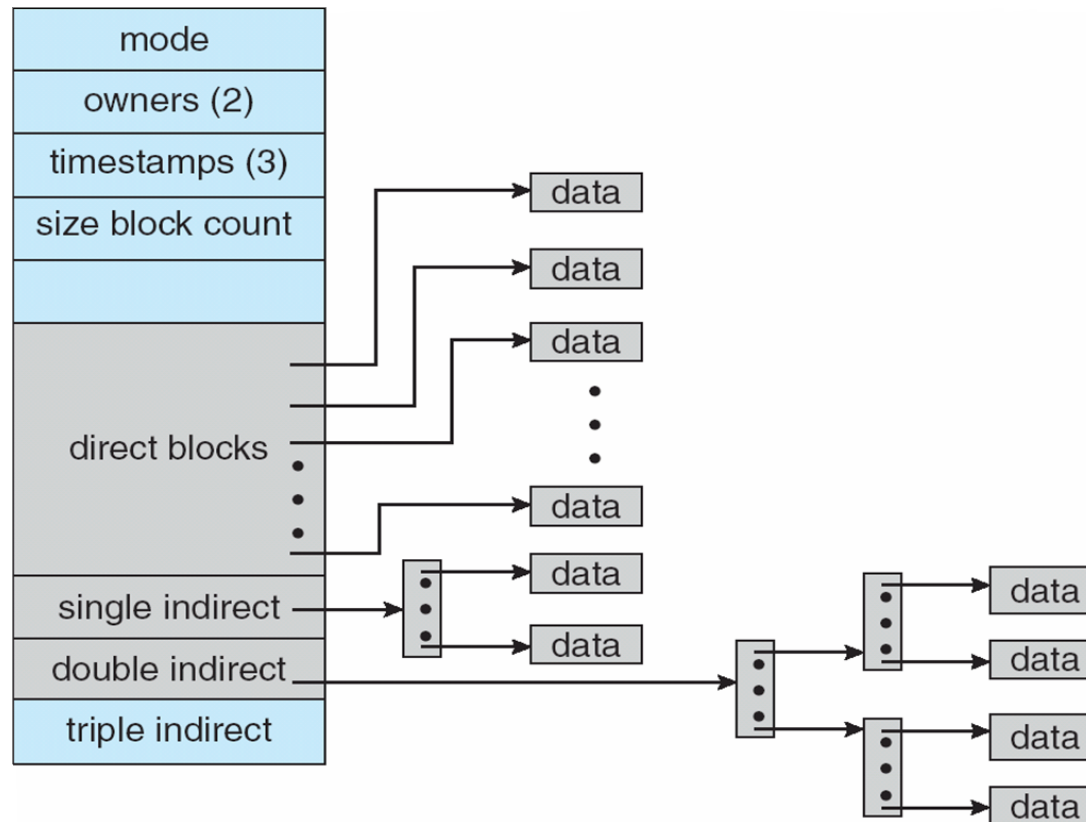
More index blocks than can be addressed with 32-bit file pointer

Performance

- Best method depends on file access type

  - Contiguous great for sequential and random

- Linked good for sequential, not random

- Declare access type at creation -> select either contiguous or linked

- Indexed more complex

  - Single block access could require 2 index block reads then data block read

  - Clustering can help improve throughput, reduce CPU overhead

THANK YOU

Venkatesh Prasad

Department of Computer Science Engineering

venkateshprasad@pes.edu