EECS 111:

System Software

Lecture : File System Implementation

Prof. Mohammad Al Faruque

The Henry Samueli School of Engineering
Electrical Engineering & Computer Science
University of California Irvine (UCI)

File System Implementation

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
  - I/O transfers performed in blocks of sectors (usually 512 bytes)
- File control block – storage structure consisting of information about a file
- Device driver controls the physical device

File system organized into layers

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical

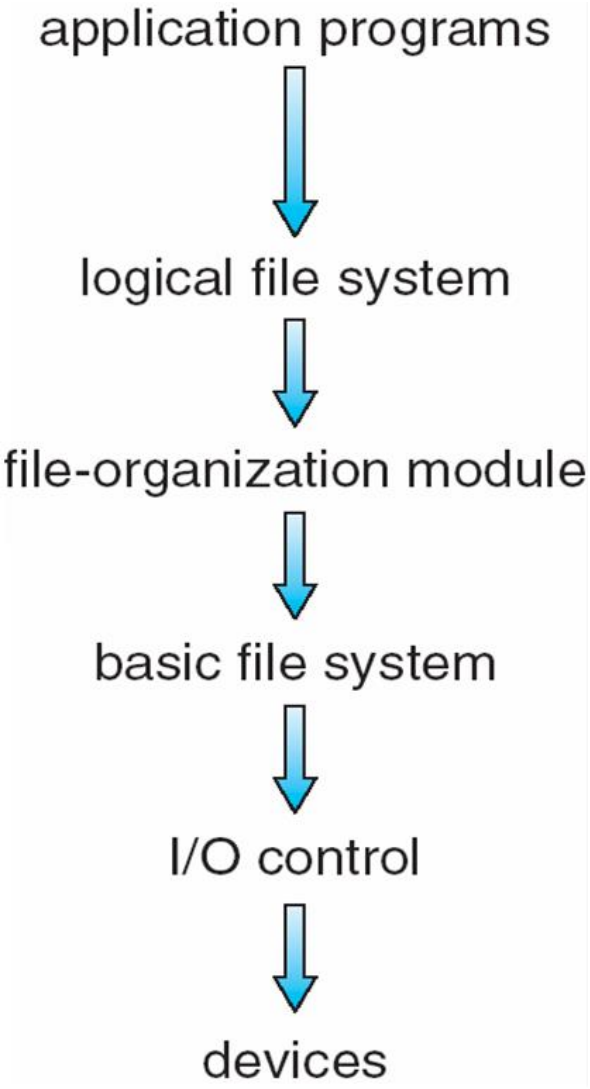
Challenges of a file system

- How the file should look to the user? →previous lecture
- Creating algorithms and data structures to map the logical file system onto the physical secondary-storage device?

- Device driver controls the physical device

File system organized into layers

Layered File System



File System Layers

- **Device drivers** manage I/O devices at the I/O control layer
  - Given commands like "**read drive1, cylinder 72, track 2, sector 10, into memory location 1060**" outputs low-level hardware specific commands to hardware controller
- **Basic file system** given command like "**retrieve block 123**" translates to device driver
  - Also manages memory buffers and caches (allocation, freeing, replacement)
  - Buffers hold data in transit
  - Caches hold frequently used data
- **File organization module** understands files, logical address, and physical blocks
  - Translates logical block # to physical block #
  - Manages free space, disk allocation

File System Layers (Cont.)

- **Logical file system manages metadata information**
  - Translates file name into file number, file handle, location by maintaining file control blocks (**inodes** in Unix)
  - Directory management
  - Protection
- **Layering useful for reducing complexity and redundancy, but adds overhead and can decrease performance**
  - Logical layers can be implemented by any coding method according to OS designer
- **Many file systems, sometimes many within an operating system**
  - Each with its own format (CD-ROM is ISO 9660; Unix has **UFS**, FFS; Windows has FAT, FAT32, NTFS as well as floppy, CD, DVD Blu-ray, Linux has more than 40 types, with **extended file system** ext2 and ext3 leading; plus distributed file systems, etc.)
  - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE

File System Implementation

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management

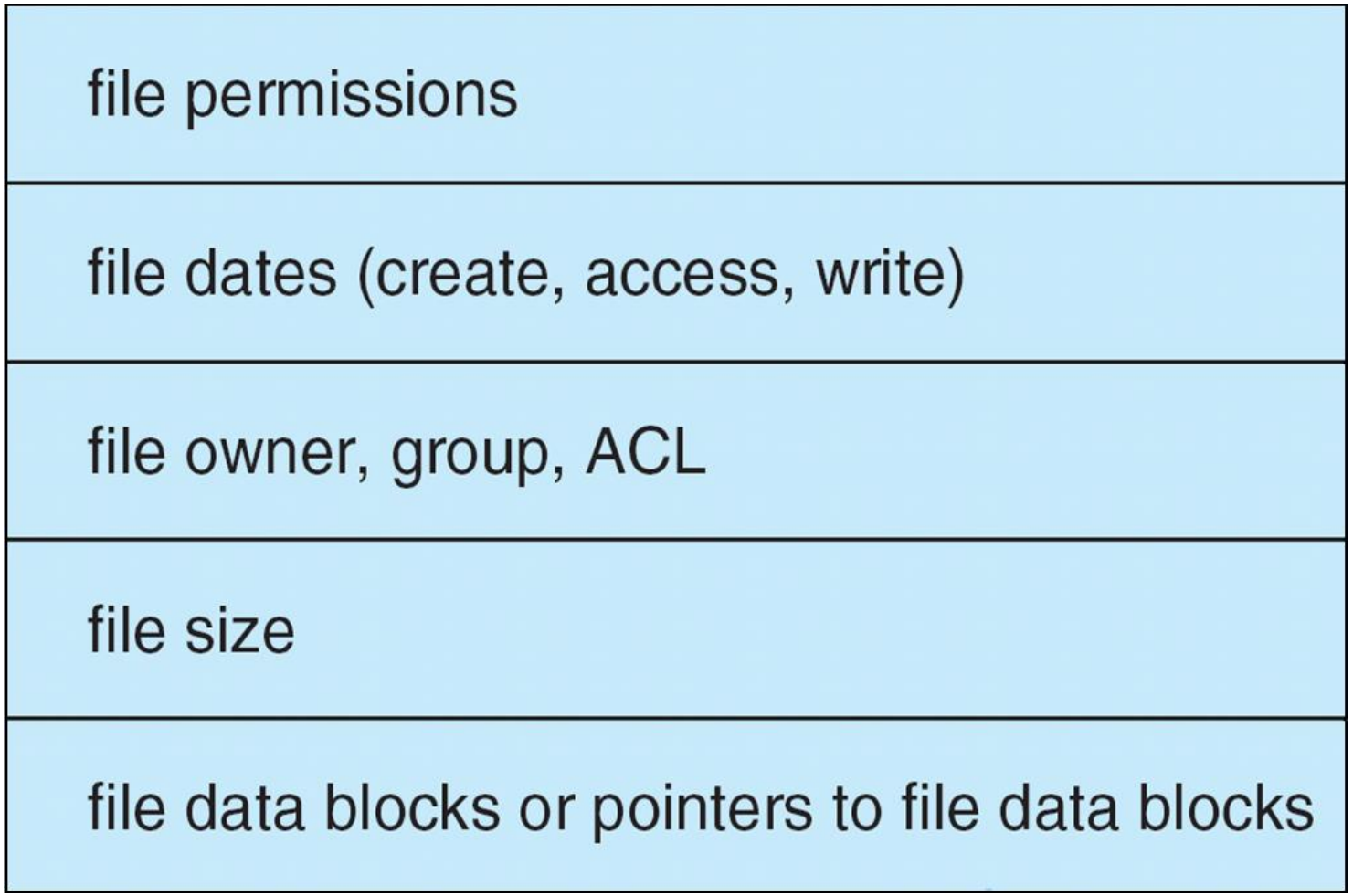
File-System Implementation

- We have system calls at the API level, but how do we implement their functions?
  - On-disk and in-memory structures

On-disk structures

- Boot control block contains info needed by system to boot OS from that volume
  - Needed if volume contains OS, usually first block of volume
- Volume control block (superblock, master file table) contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory structure organizes the files
  - Names and inode numbers, master file table
- Per-file File Control Block (FCB) contains many details about the file
  - Inode number, permissions, size, dates
  - NFTS stores into in master file table using relational DB structures

A Typical File Control Block

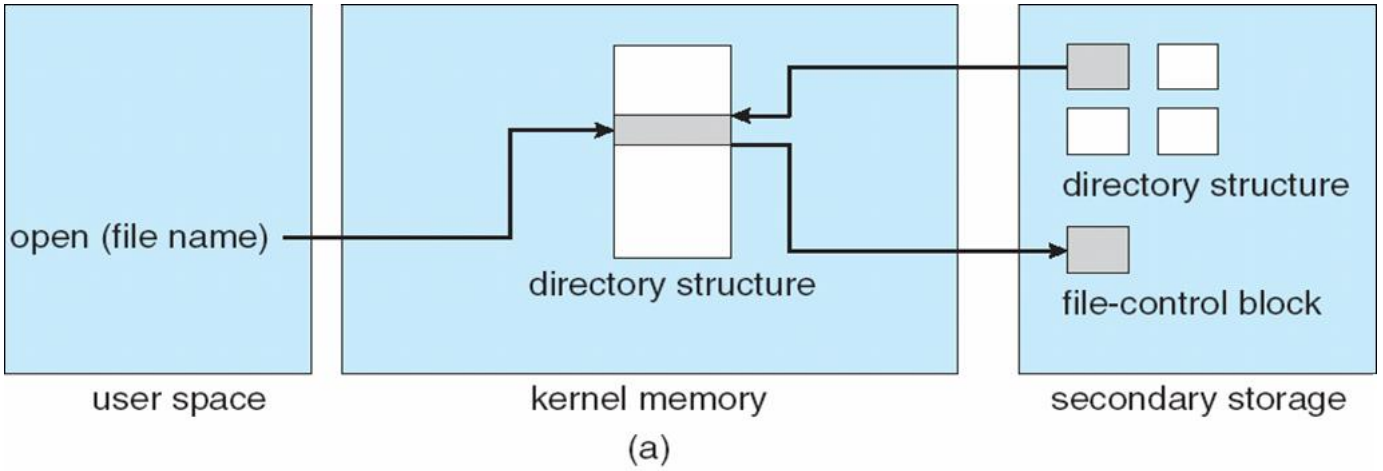


In-Memory File System Structures

In-memory structures

- Mount table storing file system mounts, mount points, file system types
- An in-memory directory-structure cache holds the directory information of recently accessed directories.
- System-wide open file table contains a copy of the FCB of each open file, also other information
- Per-process Open-file Table pointer to the appropriate entry in the system-wide open file table, other information
- Buffers hold file-system blocks when are being read from disk or write to disk.

In-Memory File System Structures



File System Implementation

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management

Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:

- Challenge: The major challenge is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly?

- Three methods exist:
  1. Contiguous
  2. Linked
  3. Indexed

Allocation Methods - Contiguous

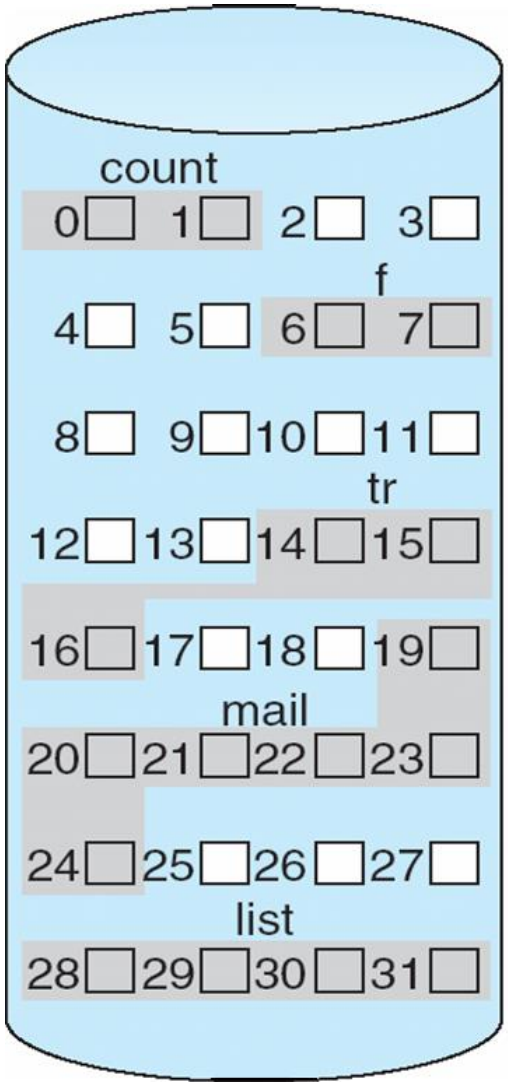
- **Contiguous allocation – each file occupies set of contiguous blocks**
  - **Best performance in most cases**
  - **Simple – only starting location (block #) and length (number of blocks) are required**
  - **Problems include finding space for file, knowing file size, external fragmentation, need for compaction off-line (downtime) or on-line**
  - **Second problem → how much space is needed for a file?**
  - **Similar to problems seen in chapter 8 memory allocation → first fit, best fit, worst fit**

Contiguous Allocation

- **Mapping from logical to physical**

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

Contiguous Allocation of Disk Space



directory

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

Extent-Based Systems

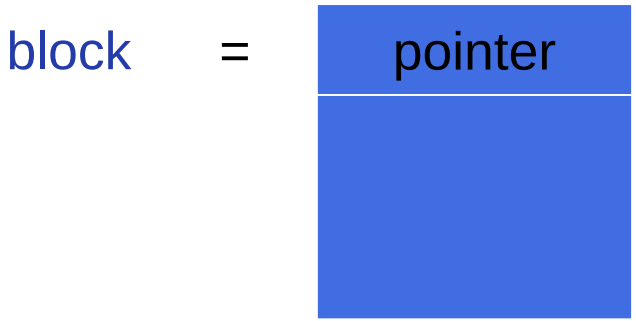
- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An extent is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents

Allocation Methods - Linked

- **Linked allocation – each file a linked list of blocks**
  - **File ends at 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**

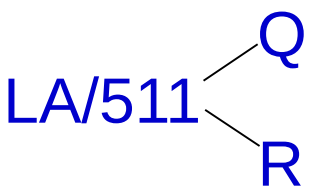
Linked Allocation

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



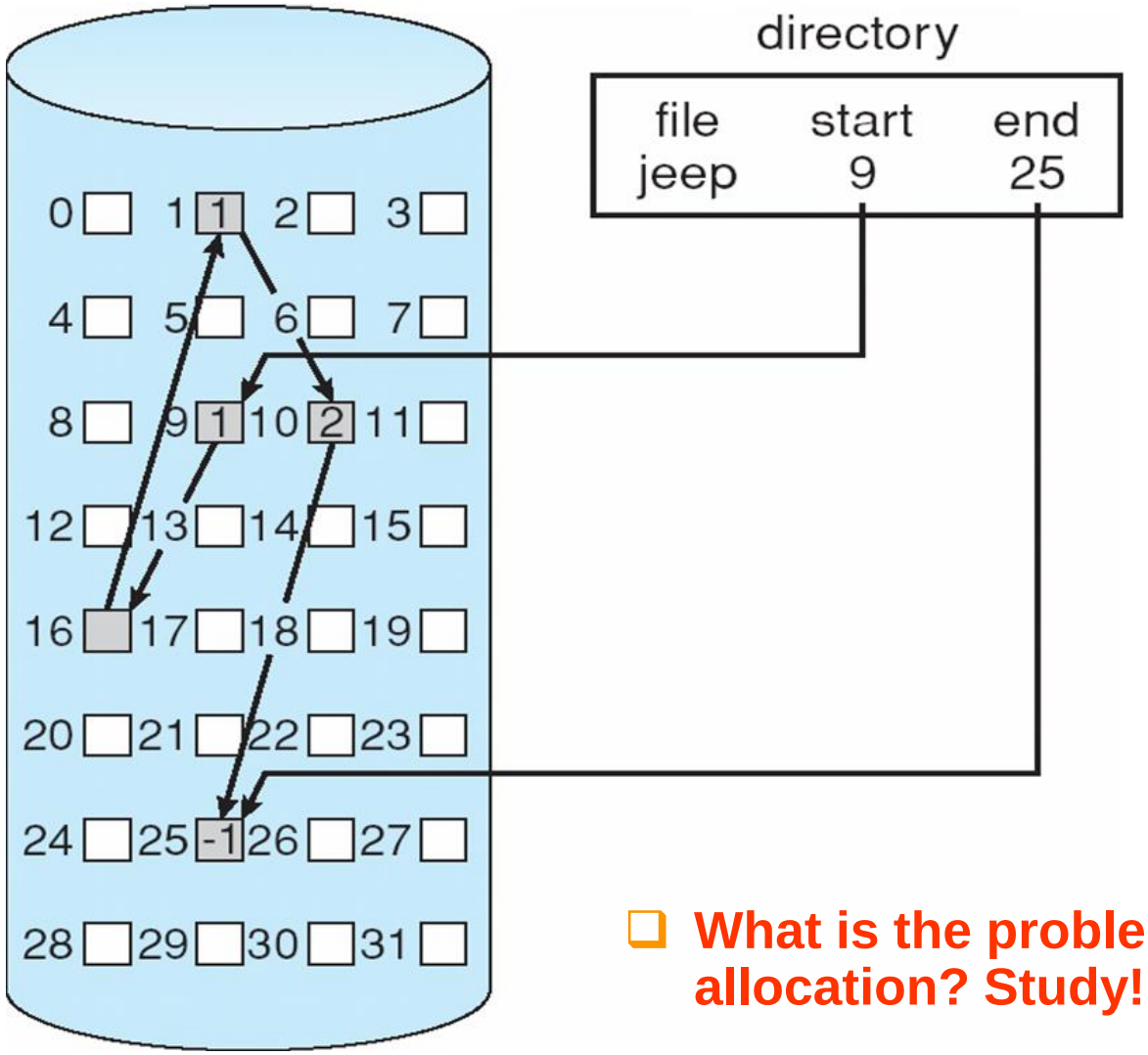
Linked Allocation

- Mapping



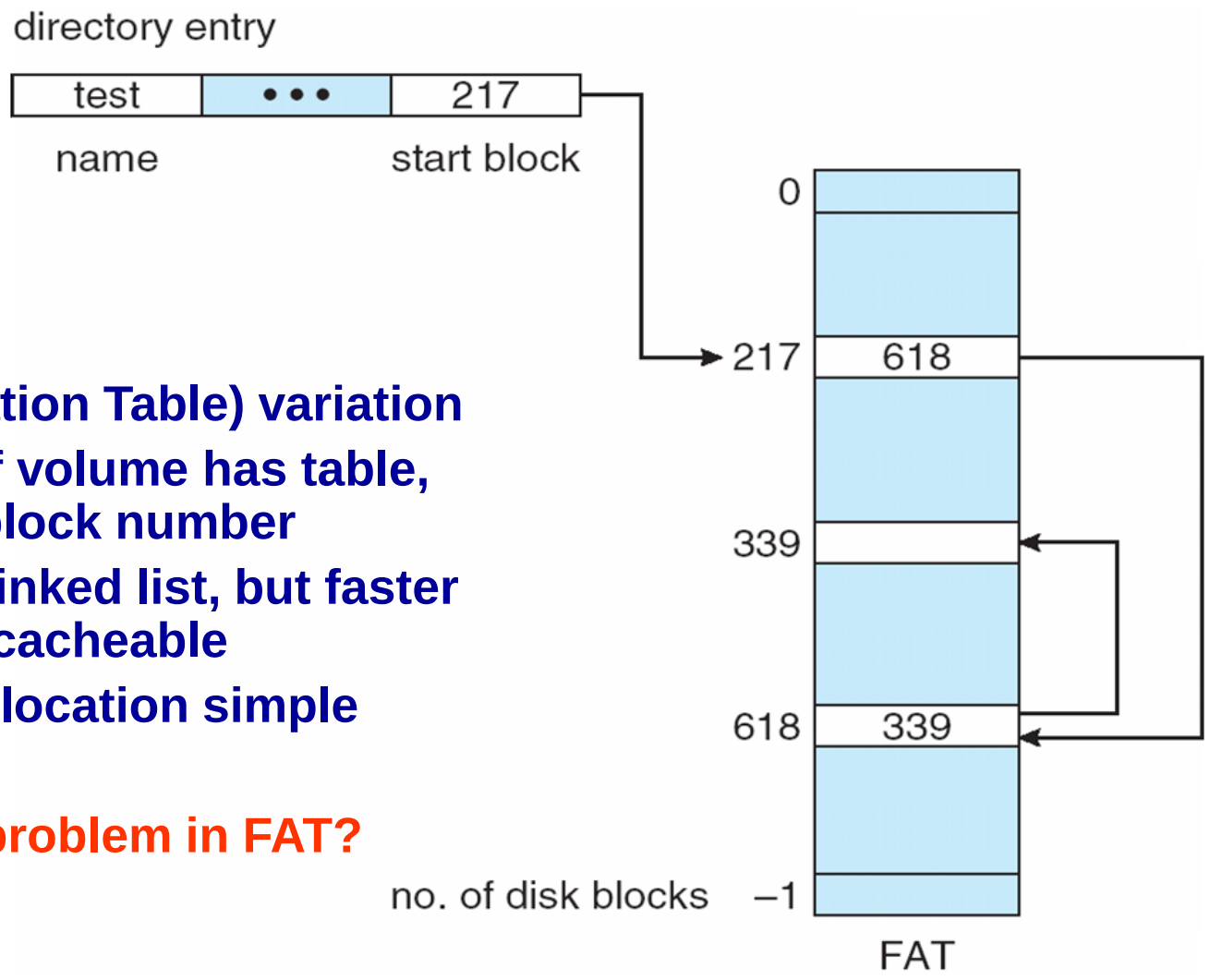
Block to be accessed is the Q^{th} block in the linked chain of blocks representing the file.
Displacement into block = R + 1

Linked Allocation



- **What is the problem in linked allocation? Study!**

File-Allocation Table

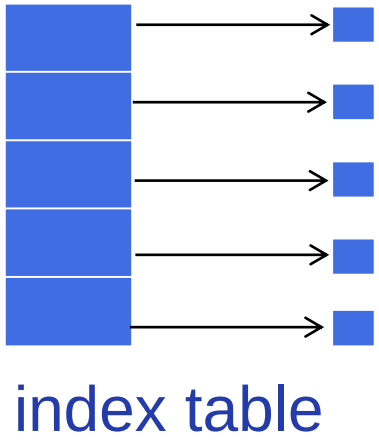


- FAT (File Allocation Table) variation
  - Beginning of volume has table, indexed by block number
  - Much like a linked list, but faster on disk and cacheable
  - New block allocation simple
  - What is the problem in FAT? Study!

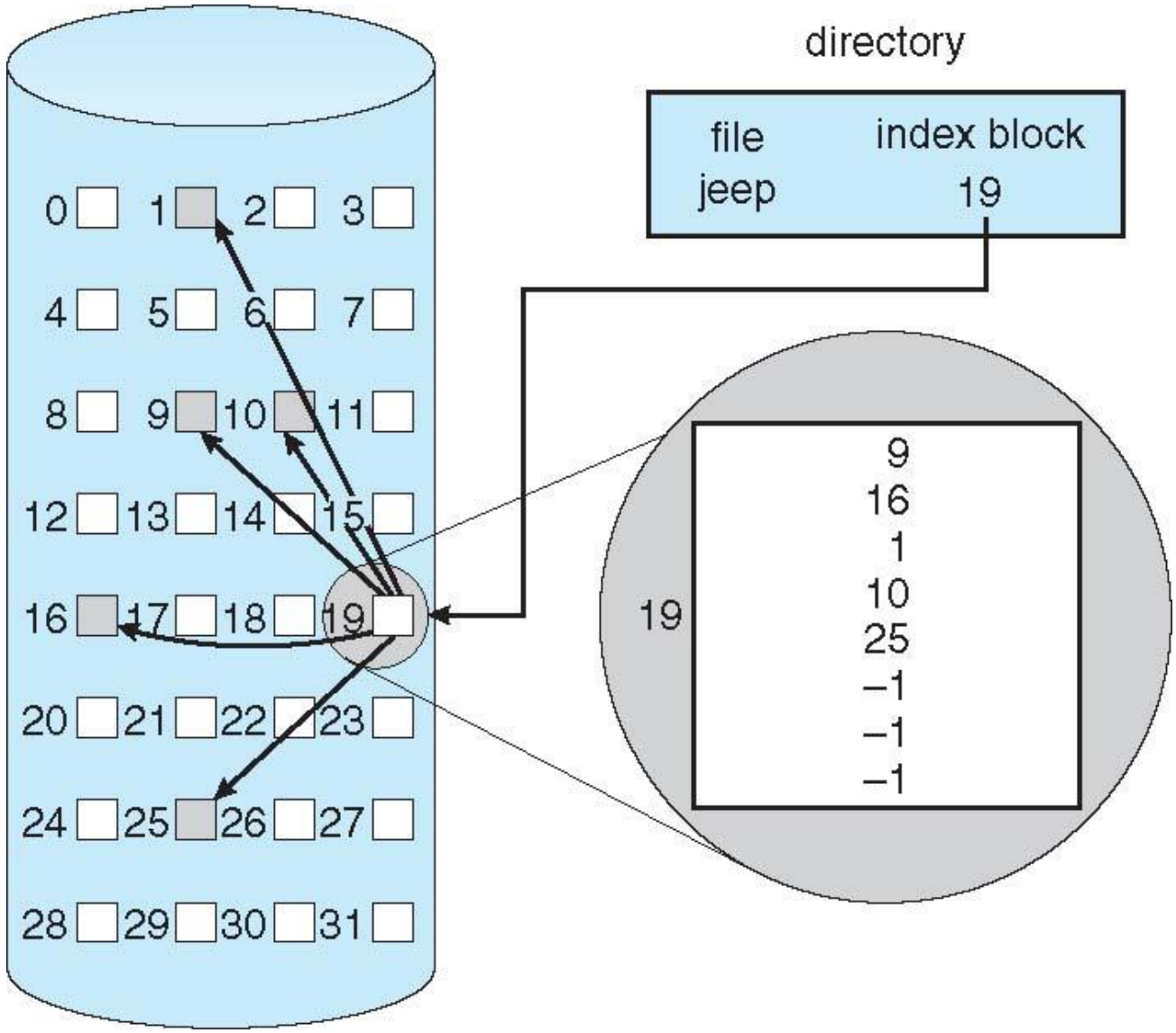
Allocation Methods - Indexed

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

- **Logical view**



Example of Indexed Allocation



References

Part of the contents of this lecture has been adapted from the book Abraham Silberschatz, Peter B. Galvin, Greg Gagne: "Operating System Concept ", Publisher : Wiley; 9 edition (December 17, 2012), ISBN-13: 978-1118063330

Slides also contain lecture materials from John Kubiatowicz (Berkeley), John Ousterhout (Stanford), Nalini (UCI), Rainer (UCI), and others

Some slides adapted from http://www-inst.eecs.berkeley.edu/~cs162/ Copyright © 2010 UCB

Thank you for your attention