

File-System Implementation

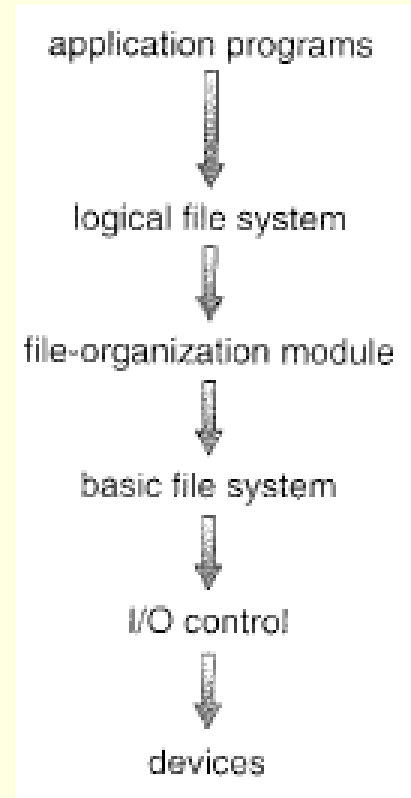
Mehdi Kargahi
School of ECE
University of Tehran
Fall 2016

File-System Structure

- To improve I/O efficiency: Unit of transfer between memory and disk is block
- Each block has one or more sectors
- Sector size varies from 32 bytes to 4KB, but normally 512 bytes

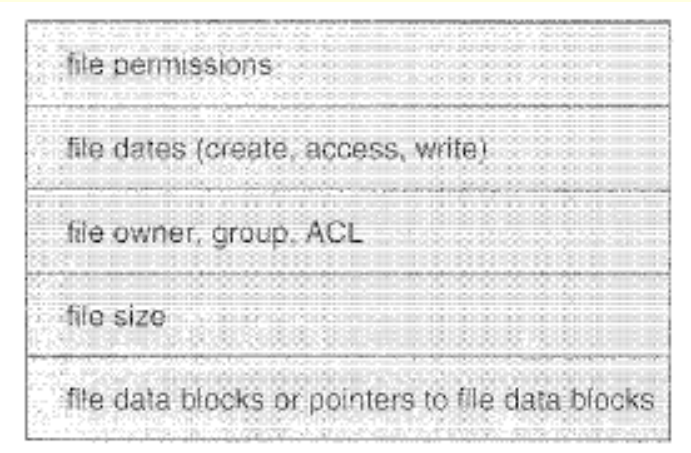
Layered File-System

- I/O control: device drivers and interrupt handlers
- Basic file-system: reads and writes physical blocks on the disk
- File-organization module:
 - Translates logical block addresses to physical block addresses
 - Tracks un-allocated blocks and provides these blocks to the file organization module
- Logical file-system: manages metadata information, directory structure, maintains file-structure via file control-block (FCB) (ownership, permissions, location of the file contents ...)
- Different file-systems: FAT, FAT32, NTFS, UFS, DVD file system, ...



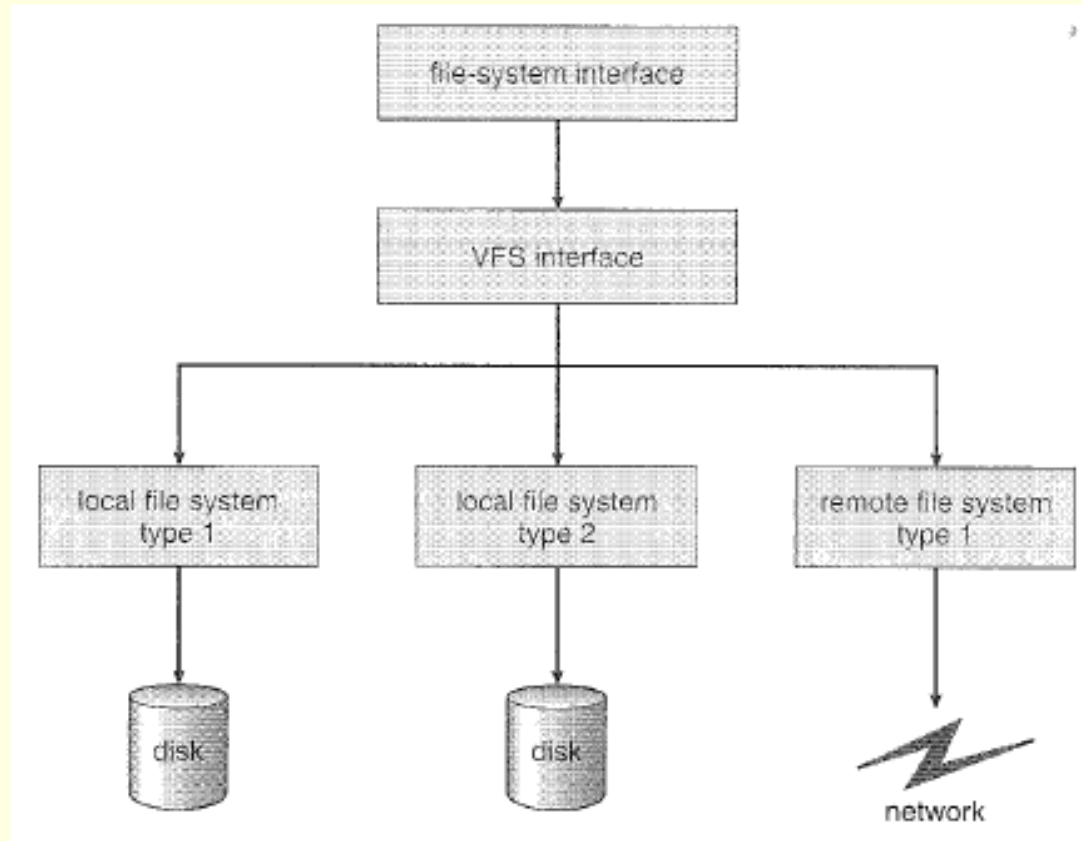
File-System Implementation

- A boot control block (per volume)
 - If the disk does not contain an OS, this block can be empty (UFS: boot block, NTFS: partition boot sector)
- A volume control block (per volume)
 - Contains volume blocks such as the number and size of blocks in the partition, free block count and pointers (UFS: superblock, NTFS: master file table)
- A per-file FCB
 - Contains details about the file (permissions, size and location of data blocks ...)
 - UFS: inode, NTFS: information stored within master file table in a row per file



file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Structure View of a Virtual File-System (VFS)



Directory Implementation

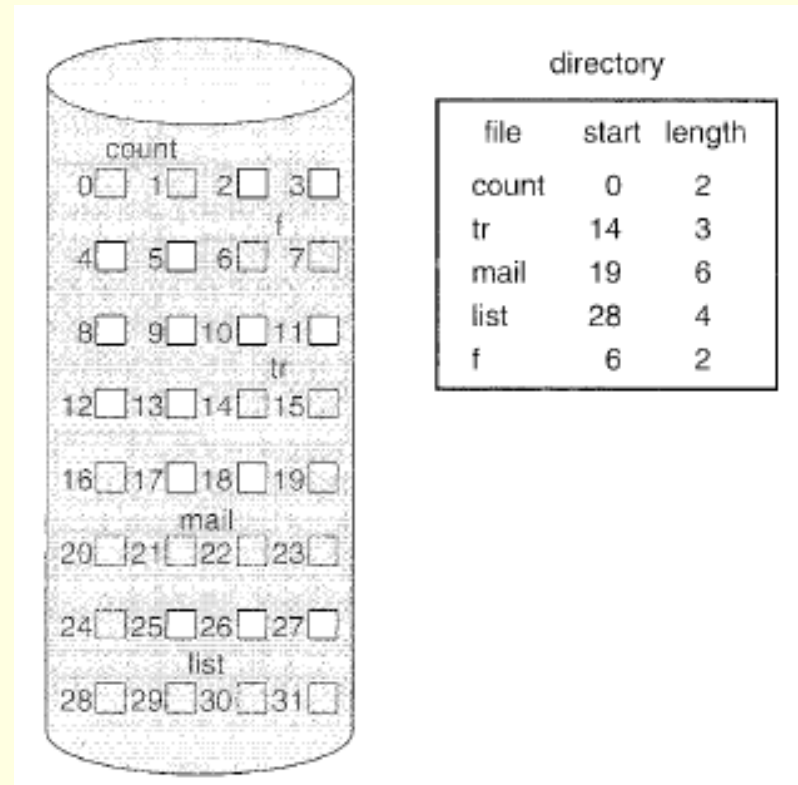
- Methods for accessing the file names
 - Linear list
 - Hash table

Allocation Methods

- Contiguous allocation
- Linked allocation
- Indexed allocation

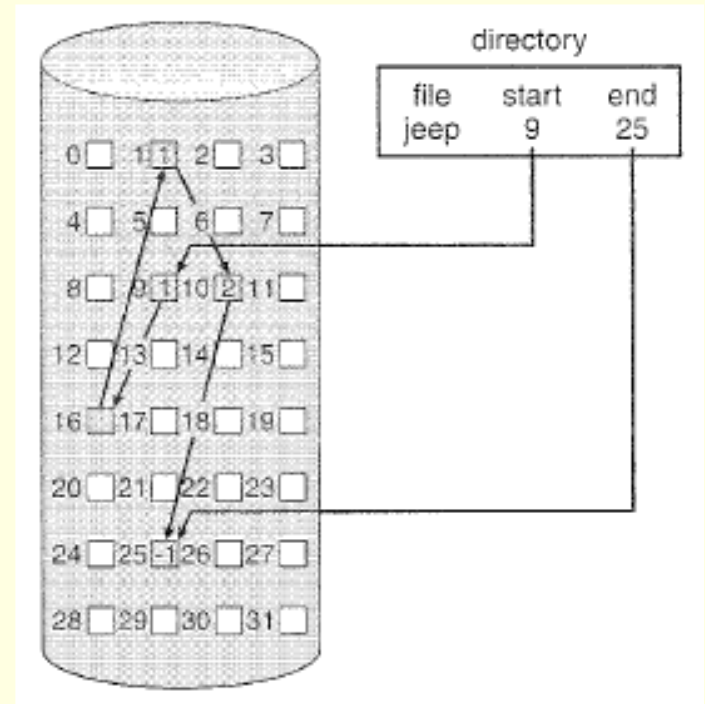
Contiguous Allocation

- Difficulties
 - Finding space for new files
 - The size of the file should be known on creation
- Solution
 - Using extents
 - A file may have a number of contiguous chunks of space linked together



Linked Allocation

- Solves all problems of contiguous allocation
- Disadvantages
 - Sequential access is only effective
 - Space is required for the pointers
 - Clusters of the blocks can be used to reduce overhead (internal fragmentation but higher throughput)
 - Reliability

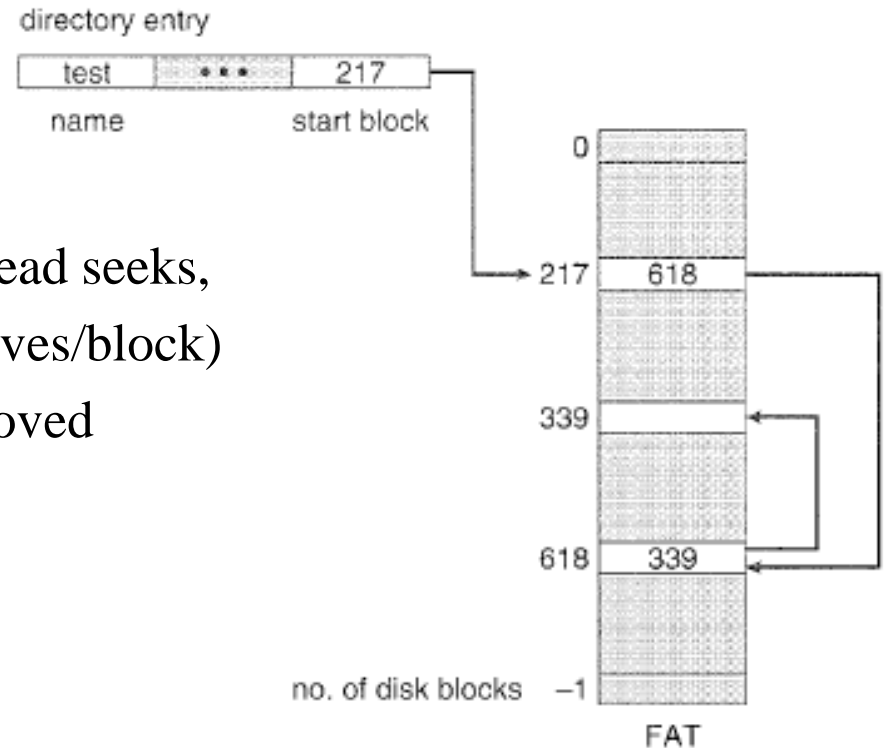


File Allocation Table (FAT)

- An important variation on linked allocation

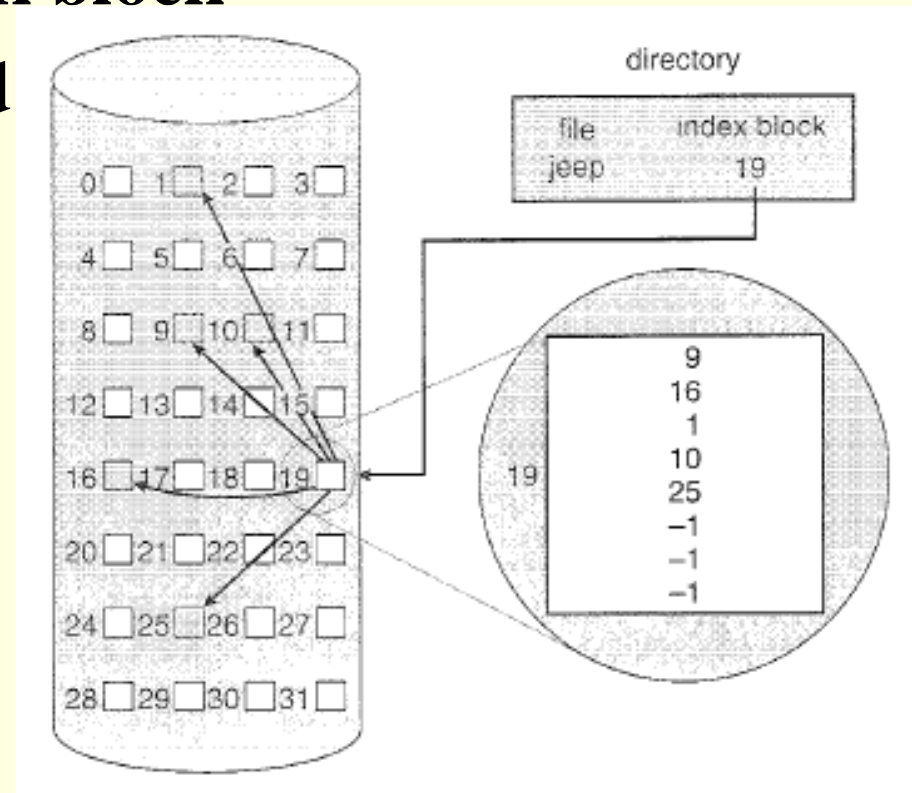
- Properties

- Significant number of disk head seeks, unless the FAT is cached (2 moves/block)
- Random-access time is improved



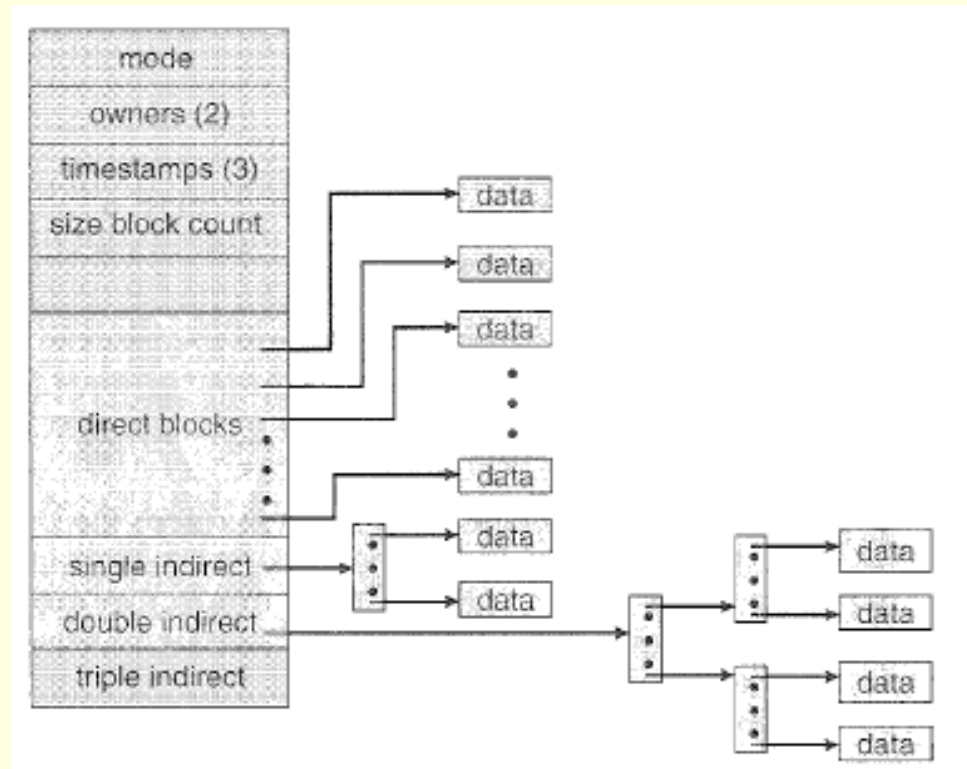
Indexed Allocation

- Bringing all the pointers together into one location, namely, **index block**
- An entire block is used for pointers

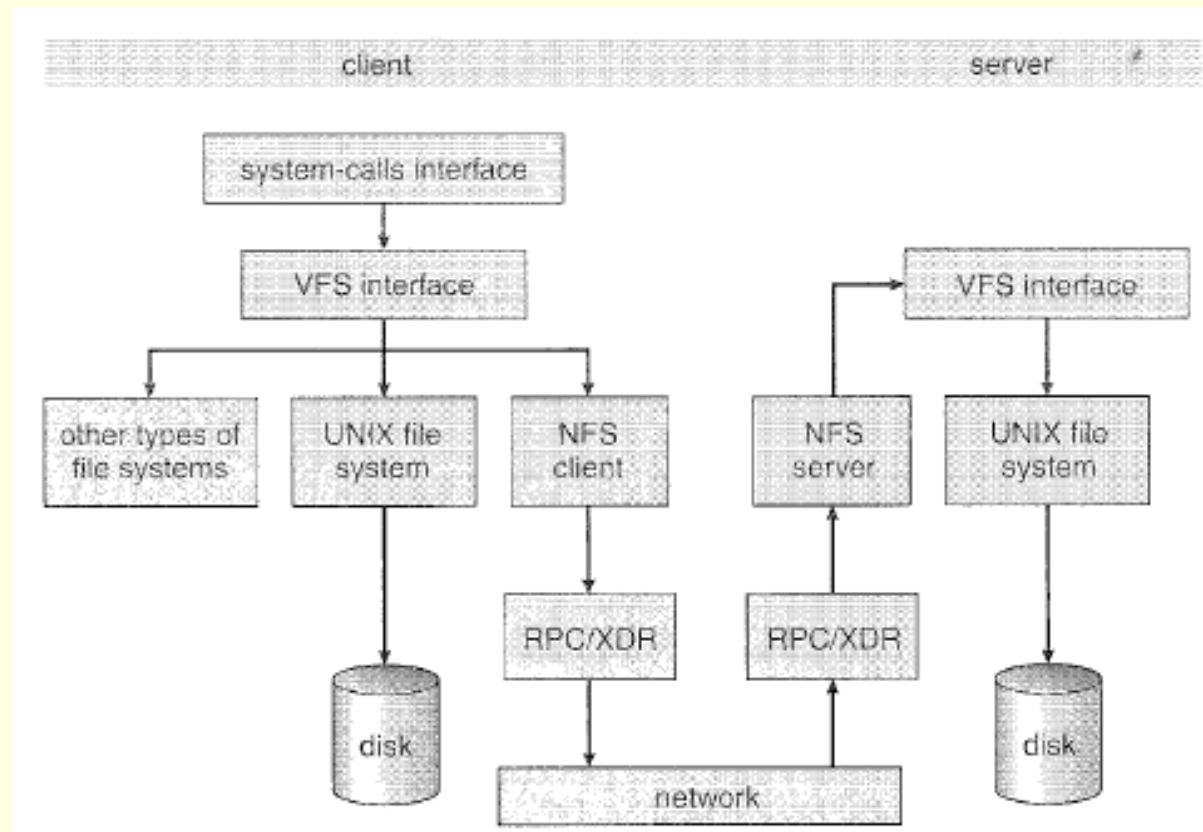


Indexed Allocation

- Linked scheme
- Multilevel index
- Combined scheme
 - The UNIX inode →

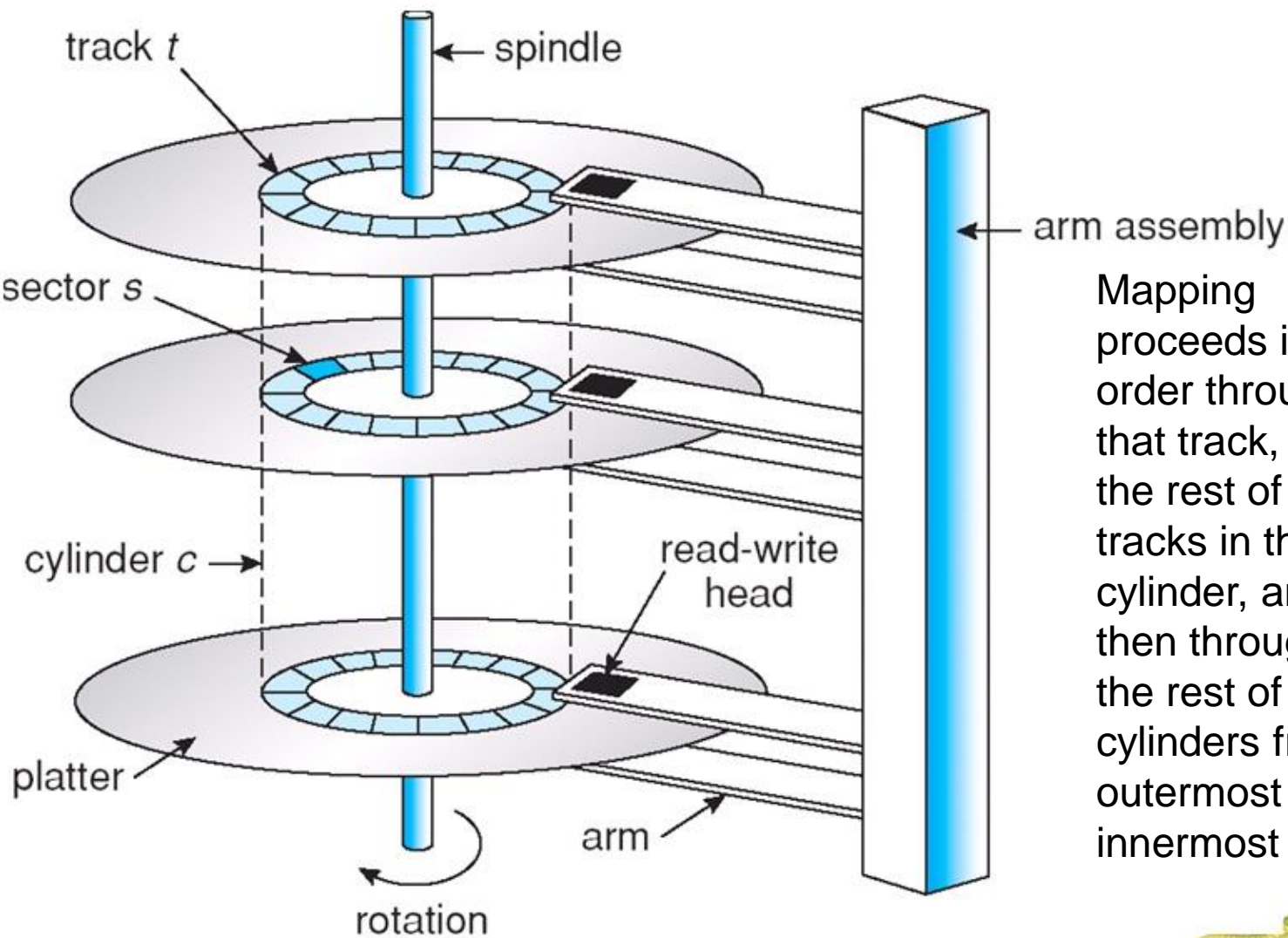


Network File-System



Mass Storage Systems

Mehdi Kargahi
School of ECE
University of Tehran
Summer 2016



Sector 0 is the first sector of the first track on the outermost cylinder.

Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost



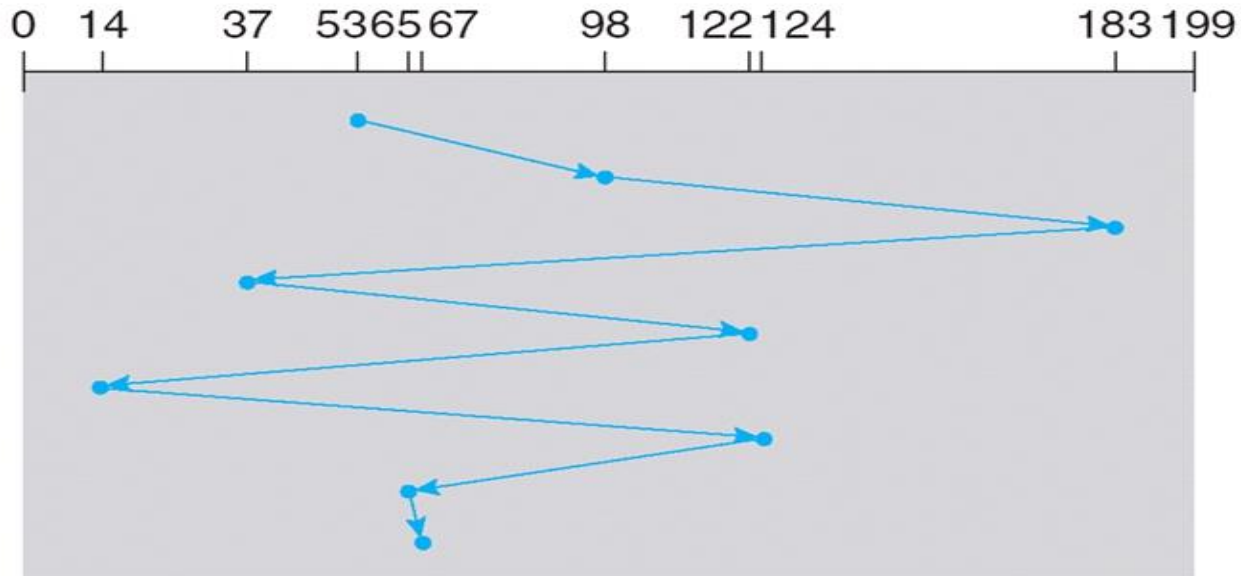
Overview

- Magnetic disks provide bulk of secondary storage of modern computers
 - Drives rotate at 60 to 200+ times per second
 - **Transfer rate** is rate at which data flow between drive and computer
 - **Positioning time (random-access time)** is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
- **Head crash** results from disk head making contact with the disk surface

Disk Scheduling-FCFS



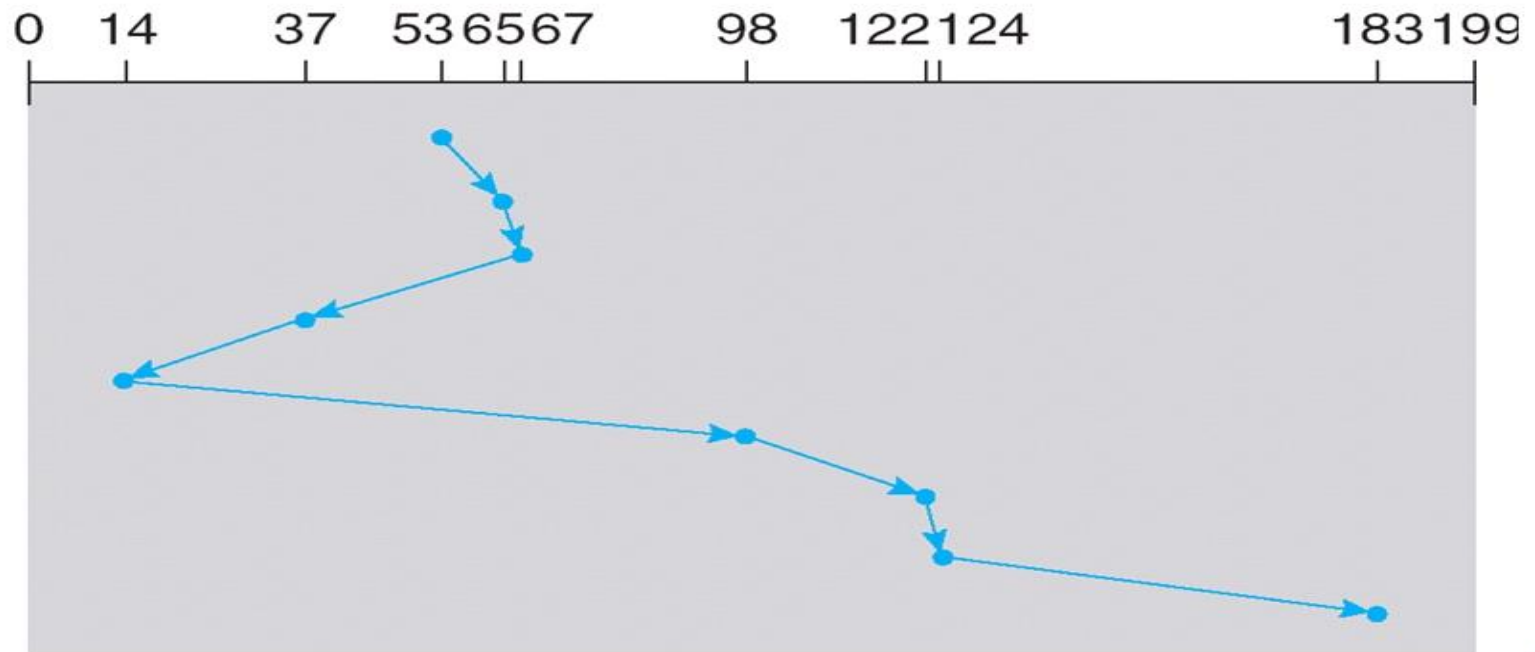
queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



Disk Scheduling-SSTF



queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



Disk Scheduling-SCAN (Elevator)

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.

Disk Scheduling-SCAN (Elevator)



queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



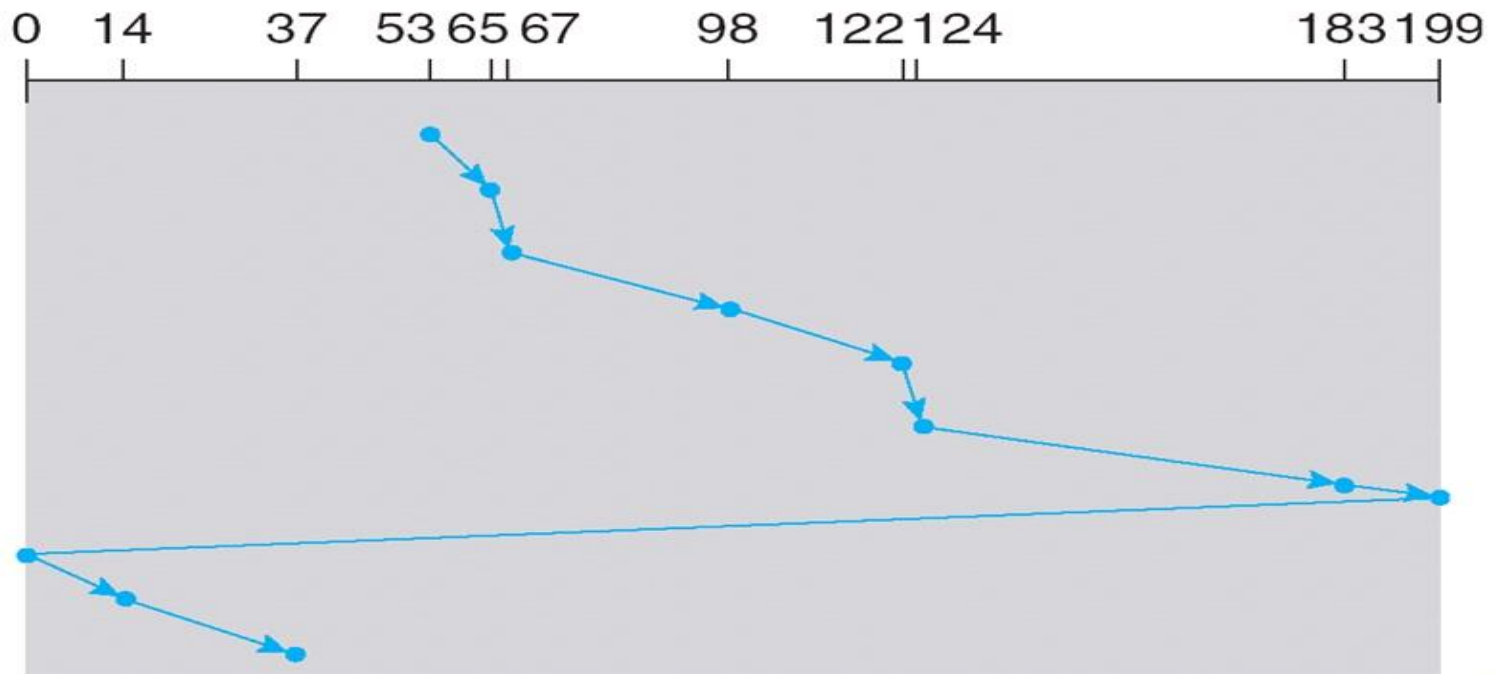
Disk Scheduling-C-SCAN

- The head moves from one end of the disk to the other, servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- Provides a more uniform wait time than SCAN

Disk Scheduling-C-SCAN



queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



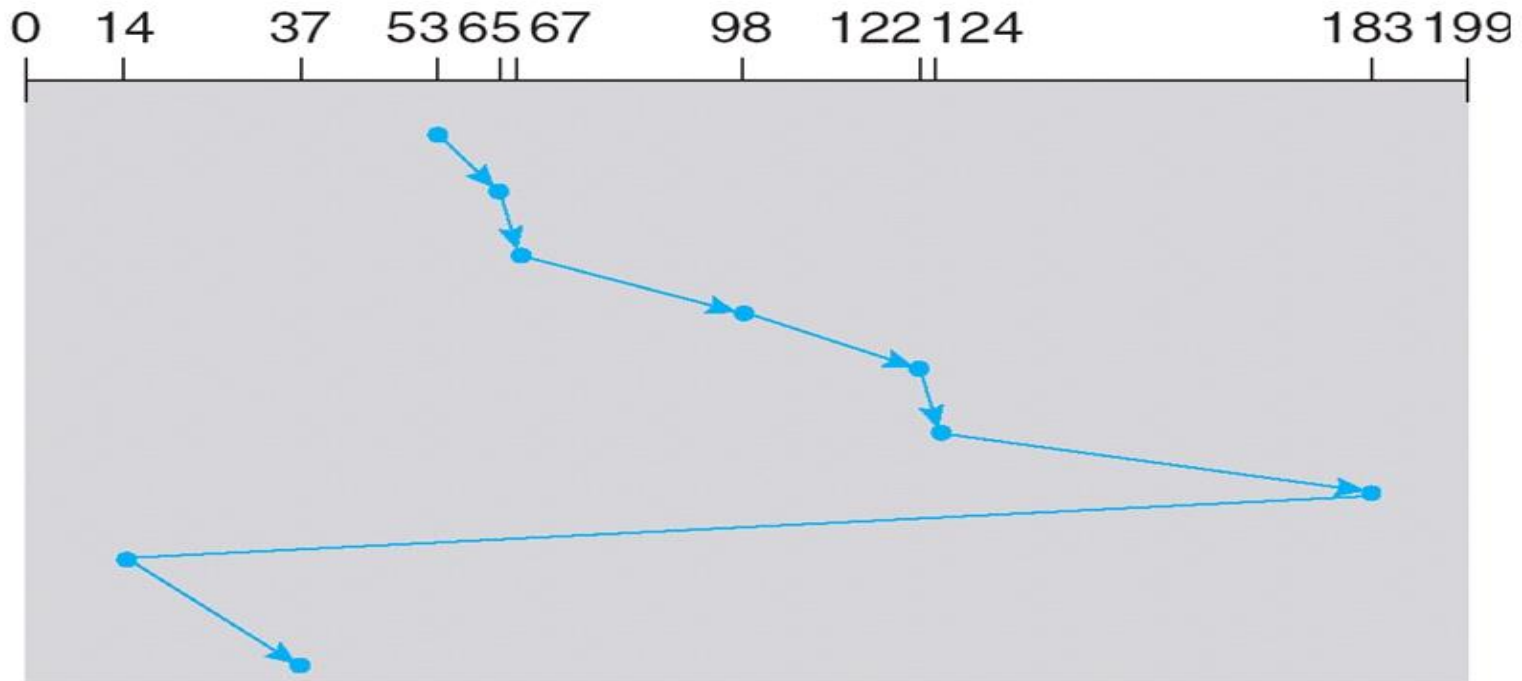
Disk Scheduling-C-LOOK

- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- Version of C-SCAN

Disk Scheduling-C-LOOK



queue 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



RAID Structure

- **RAID** (Redundant Arrays of Independent Disks): multiple disk drives provide **reliability** via **redundancy**
- Disk striping uses a group of disks as one storage unit
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
 - Mirroring or shadowing keeps duplicate of each disk
 - Block interleaved parity uses much less redundancy

RAID Structure



(a) RAID 0: non-redundant striping.



(b) RAID 1: mirrored disks.



(c) RAID 2: memory-style error-correcting codes.



(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.



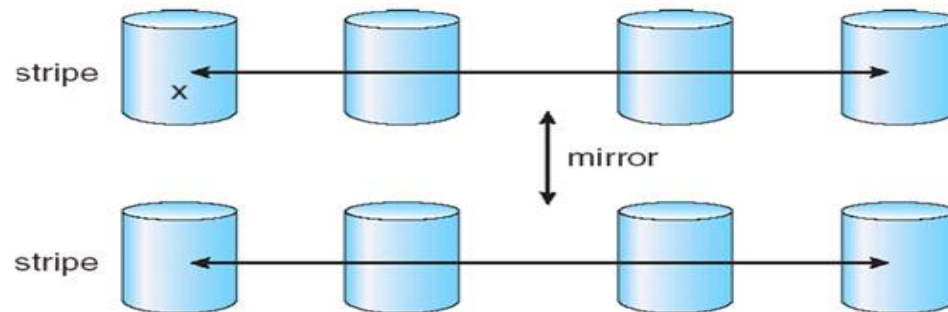
(f) RAID 5: block-interleaved distributed parity.



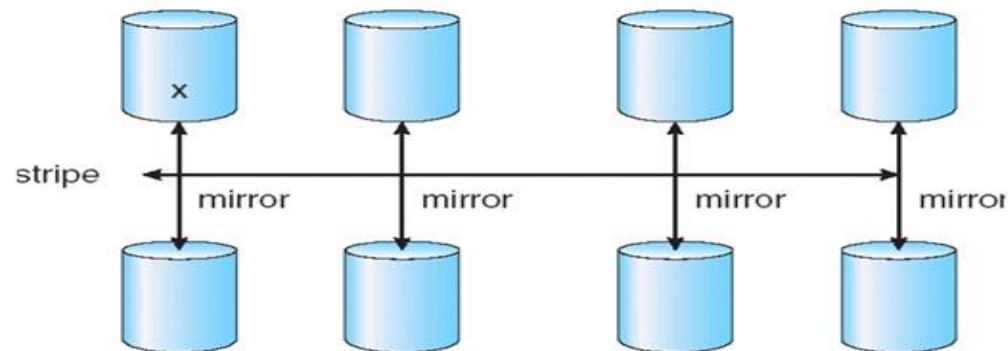
(g) RAID 6: P + Q redundancy.



RAID Structure



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

