



CS423 Operating Systems File Systems



Why Files?

- Physical reality
 - Block oriented
 - Physical sector #s
 - No protection among users of the system
 - Data might be corrupted if machine crashes
- Filesystem model
 - Byte oriented
 - Named files
 - Users protected from each other
 - Robust to machine failures



Question

- What functions should file systems provide?

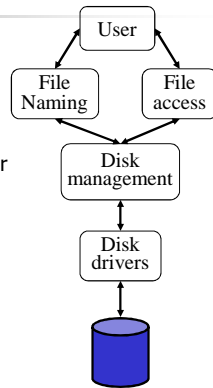


File System Requirements

- Users must be able to:
 - create and delete files at will.
 - read, write, and modify file contents with a minimum of fuss about blocking, buffering, etc.
 - share each other's files with proper authorization
 - refer to files by symbolic names.
 - see a logical view of files without concern for how they are stored.
 - retrieve backup copies of files lost through accident or malicious destruction.

File System Components

- Disk management
 - Arrange collection of disk blocks into files
- Naming
 - User gives file name, not track or sector number, to locate data
- Security
 - Keep information secure
- Reliability/durability
 - When system crashes, lose stuff in memory, but want files to be durable

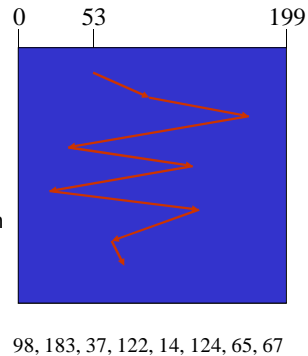


Disk Scheduling

- Which disk request is serviced first?
 - FCFS
 - Shortest seek time first
 - Elevator (SCAN)
 - C-SCAN (Circular SCAN)

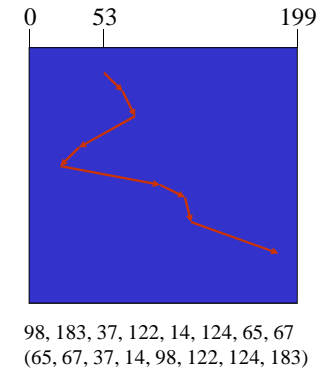
FIFO (FCFS) order

- Method
 - First come first serve
- Pros
 - Fairness among requests
 - In the order applications expect
- Cons
 - Arrival may be on random spots on the disk (long seeks)
 - Wild swing can happen
- Analogy:
 - FCFS elevator scheduling?



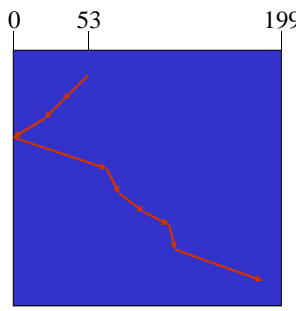
SSTF (Shortest Seek Time First)

- Method
 - Pick the one closest on disk
- Pros
 - Try to minimize seek time
- Cons
 - Starvation
- Question
 - Is SSTF optimal?
 - Can we avoid starvation?



Elevator (SCAN)

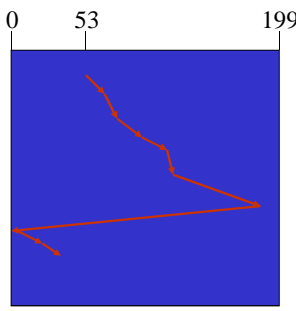
- Method
 - Take the closest request in the direction of travel
- Pros
 - Bounded time for each request
- Cons
 - Request at the other end will take a while



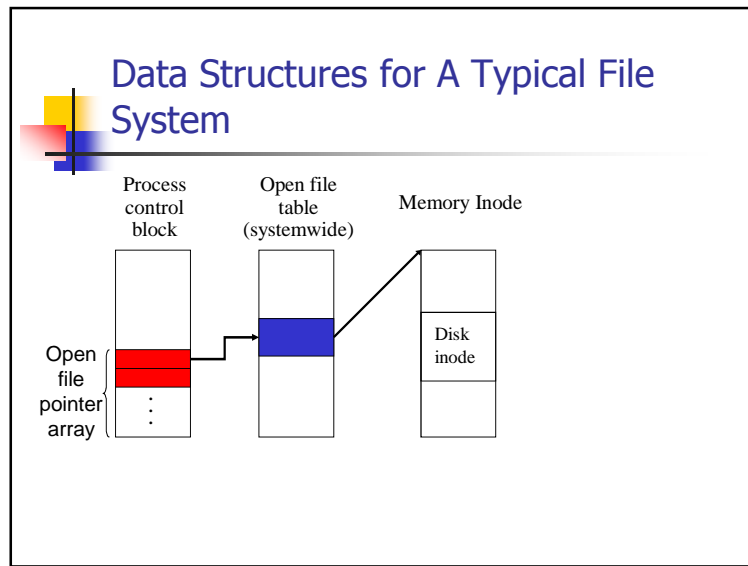
98, 183, 37, 122, 14, 124, 65, 67
(37, 14, 65, 67, 98, 122, 124, 183)

C-SCAN (Circular SCAN)

- Method
 - Like SCAN
 - But, wrap around
- Pros
 - Uniform service time
- Cons
 - Do nothing on the return



98, 183, 37, 122, 14, 124, 65, 67
(65, 67, 98, 122, 124, 183, 14, 37)



A Disk Layout for A File System

Boot block	Super block	File metadata (i-node in Unix)	File data blocks
------------	-------------	--------------------------------	------------------

- Superblock defines a file system
 - size of the file system
 - size of the file descriptor area
 - free list pointer, or pointer to bitmap
 - location of the file descriptor of the root directory
 - other meta-data such as permission and various times
- For reliability, replicate the superblock

File Allocation

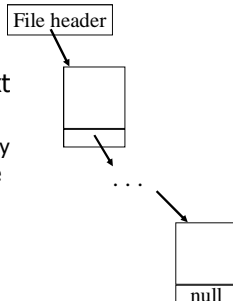
- Contiguous
- Non-contiguous (linked)
- Tradeoffs?

Contiguous Allocation

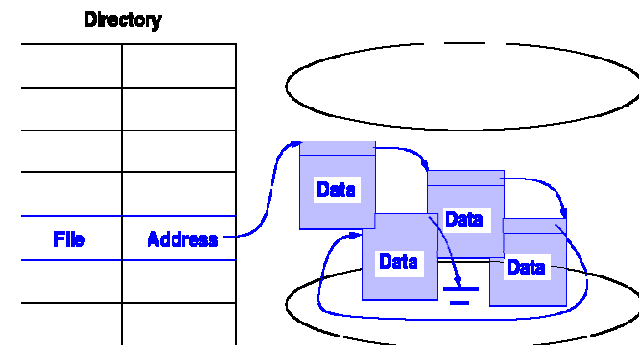
- Request in advance for the size of the file
- Search bit map or linked list to locate a space
- File header
 - first sector in file
 - number of sectors
- Pros
 - Fast sequential access
 - Easy random access
- Cons
 - External fragmentation
 - Hard to grow files

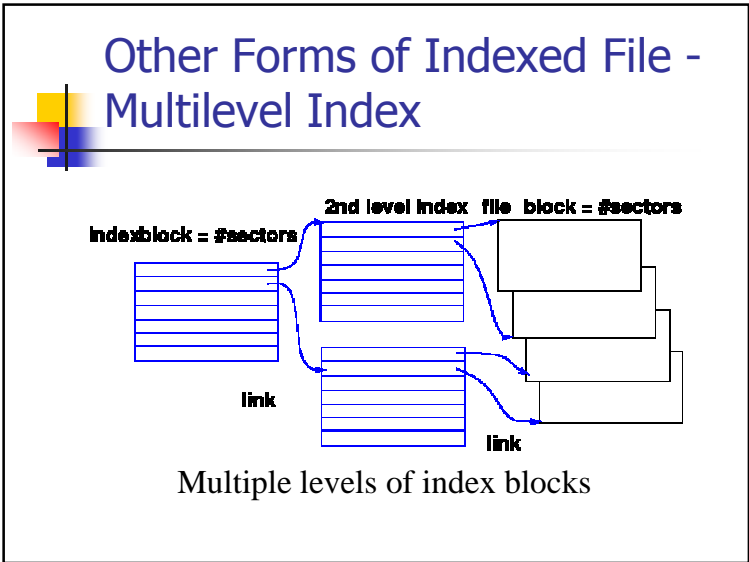
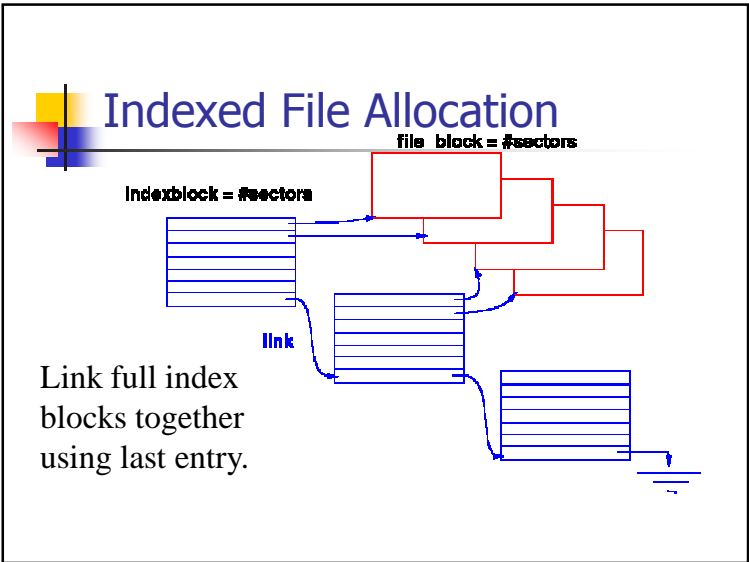
Linked Files

- File header points to 1st block on disk
- Each block points to next
- Pros
 - Can grow files dynamically
 - Free list is similar to a file
- Cons
 - random access: horrible
 - unreliable: losing a block means losing the rest

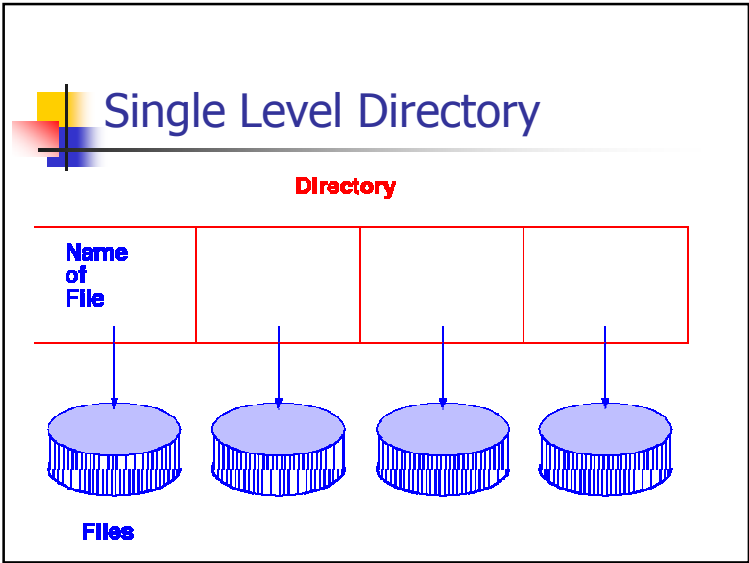


Linked Allocation





- ### Directory Structure Organization
- maps symbolic names into logical file names
 - search
 - create file
 - list directory
 - backup, archival, file migration



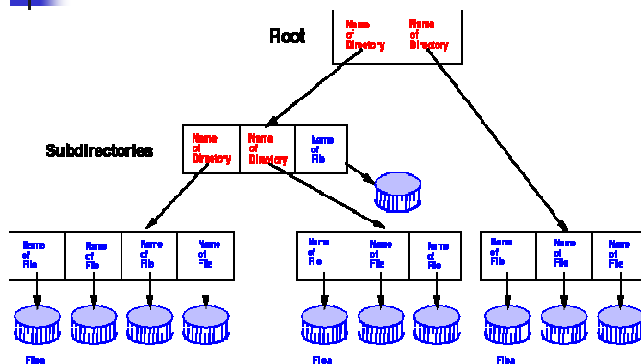
Problems With Single Level Directory

- more than one user
- large file systems
- moving files from one system to another
- name clashes
- modularity

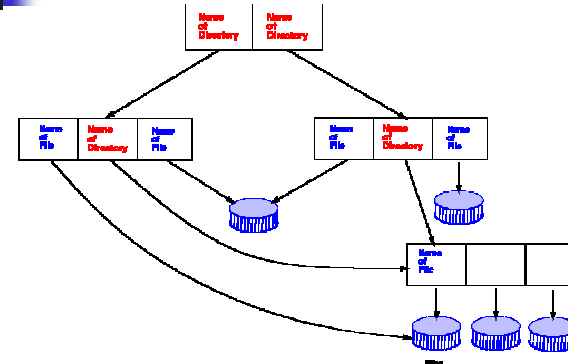
Tree Structured Directories

- arbitrary depth of directories
- leaf nodes are files
- interior nodes are directories
- path name lists nodes to traverse to find node
- use absolute paths from root
- use relative paths from current working directory pointer

Tree Structured Directories



Acyclic Graph Structured Directories



Symbolic links

- **Symbolic** links are different than regular links (often called **hard links**). Created with **ln -s**
- Can be thought of as a directory entry that points to the name of another file.
- Does not change link count for file
 - When original deleted, symbolic link remains
- They exist because:
 - Hard links don't work across file systems
 - Hard links only work for regular files, not directories

