Observations

- Getting first byte from disk read is slow
  - high latency
- Peak bandwidth high, but rarely achieved
- Need to mitigate disk performance impact
  - Do extra calculations to speed up disk access
    - Schedule requests to shorten seeks
  - Move some disk data into main memory – file system caching

Disk Scheduling

- Which disk request is serviced first?
  - FCFS
  - Shortest seek time first
  - Elevator (SCAN)
  - C-SCAN (Circular SCAN)
- Look familiar?
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:**
  - Can elevator scheduling use FCFS?

SSTF (Shortest Seek Time First)

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

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

- **Method**
  - Take the closest request in the direction of travel
  - Travel to the end and change the direction

- **Pros**
  - Bounded time for each request

- **Cons**
  - Request at the other end will take a while

- **LOOK algorithm**
  - Do not go to the end
  - Service the last request, then change the direction

C-SCAN (Circular SCAN)

- **Method**
  - Like SCAN
  - But, wrap around

- **Pros**
  - Uniform service time

- **Cons**
  - Do nothing on the return

- **C-LOOK**
  - Do not go to the end
  - Service the last request, then wrap around
Why Files?

- **Physical reality**
  - Block oriented
  - Physical sector #s
  - No protection among users of the system
  - Data might be corrupted if machine crashes

- **File system model**
  - Byte oriented
  - Named files
  - Users protected from each other
  - Robust to machine failures

File System Requirements

- **Users must be able to:**
  - create, modify, 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
  - transfer information between files.
  - refer to files by symbolic names.
  - retrieve backup copies of files lost through accident or malicious destruction.
  - see a logical view of their files without concern for how they are stored.
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

Directory Contents

- Each entry is for one file:
  - file name (symbolic name)
  - file type indicates format of file
  - location device and location
  - size
  - protection
  - creation, access, and modification date
  - owner identification
Directory Operations

- maps symbolic names into logical file names
  - search
  - create file
  - list directory
  - backup, archival, file migration

Tree Structured Directories
Tree Structured Directories

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