Secondary Storage Structures

- Disk structure
- Scheduling disk requests
  - » Scheduling algorithms
  - » Balancing performance and response time
- Managing disk information
  - » Disk formatting
  - » Booting from disk
- Managing swap space for virtual memory
- Dealing with disk reliability
  - » Disk failures & dealing with them
  - » Atomic updates to structures on disk

Overall Disk Structure

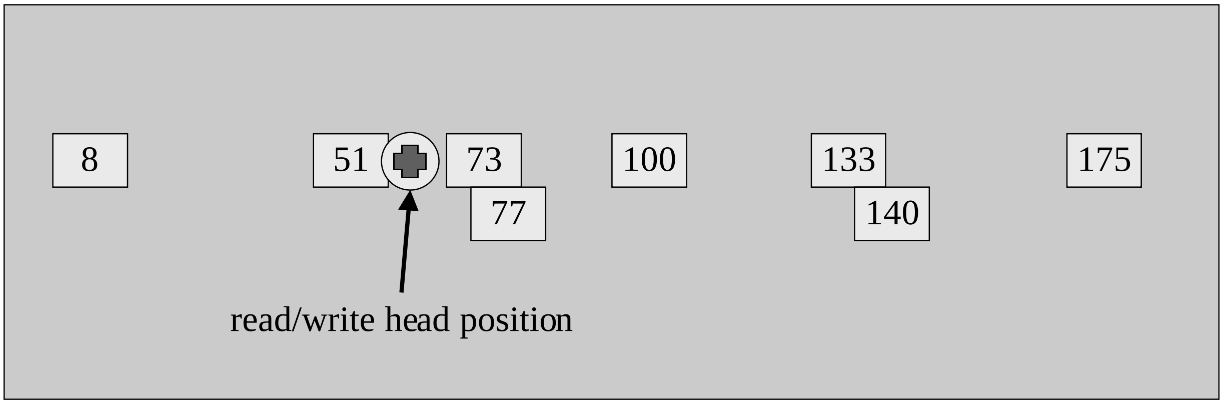
- In the olden days
  - » Disks were organized into cylinders, tracks, and sectors
  - » Operating systems had to know disk geometry!
- In modern disks
  - » Disks are broken into sequentially numbered logical blocks
  - » Logical blocks with nearby numbers are nearby on disk
  - » Logical block numbers are assigned by:
    - – Putting sector 0 on the first sector of the first track in the outermost cylinder
    - – Numbering continues
      - Rest of track
      - Rest of cylinder
      - Rest of the cylinders on the disk, moving inward
  - » File systems refer to blocks by logical address

Scheduling Disk Requests

- Goal: use disk hardware efficiently
  - » Bandwidth as high as possible
  - » Disk transferring as often as possible
- We want to
  - » Minimize disk seek time (moving from track to track)
  - » Minimize rotational latency (waiting for disk to rotate the desired sector under the read/write head)
- Calculate disk bandwidth by
  - » Total bytes transferred / time to service request
  - » Seek time & rotational latency are overhead (no data is transferred), and reduce disk bandwidth
- Minimize seek time & rotational latency by
  - » Using algorithms to find a good sequence for servicing requests
  - » Placing blocks of a given file “near” each other

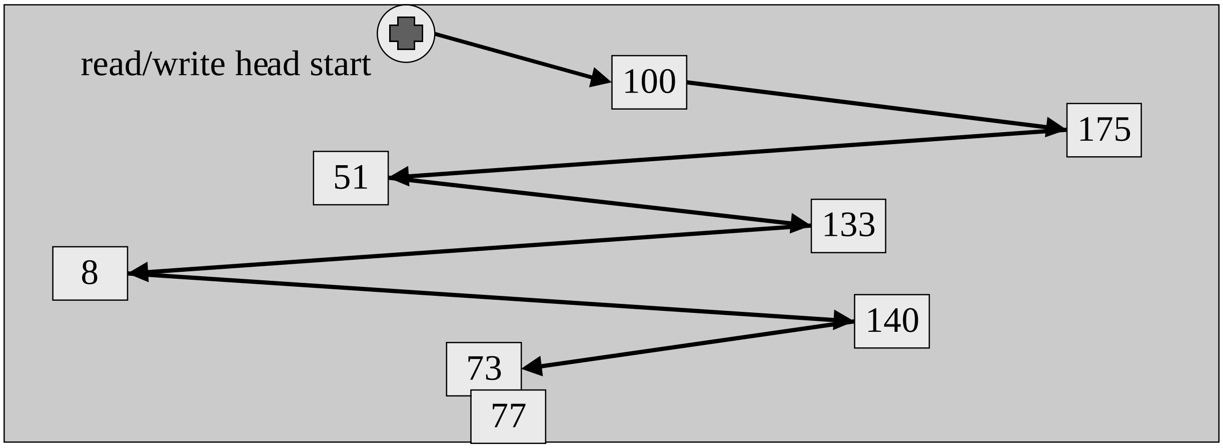
Disk Scheduling Algorithms

- Schedule disk requests to minimize disk seek time
  - » Seek time increases as distance increases (though not linearly)
  - » Minimize seek distance -> minimize seek time
- Disk seek algorithm examples assume a request queue & head position (disk has 200 cylinders)
  - » Queue = 100, 175, 51, 133, 8, 140, 73, 77
  - » Head position = 63



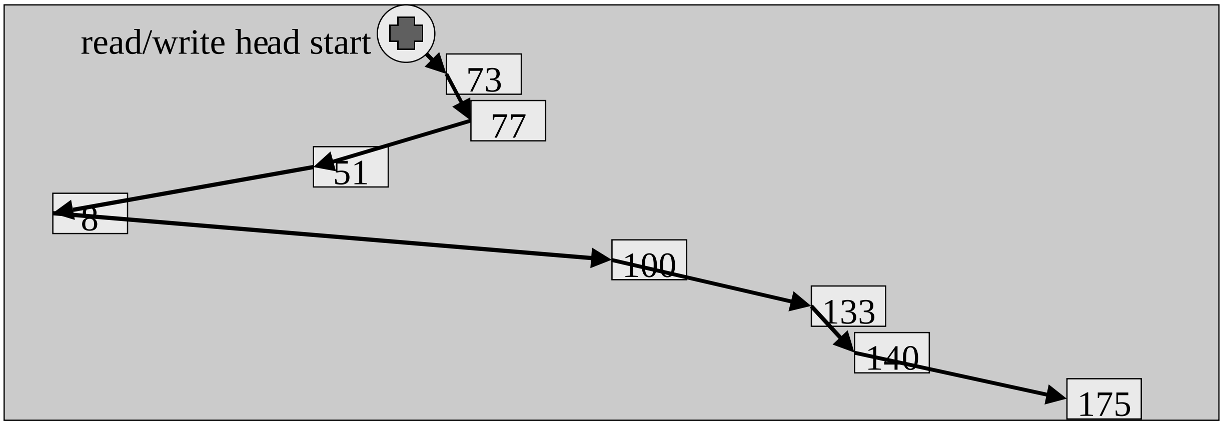
First-Come-First Served (FCFS)

- Requests serviced in the order in which they arrived
  - » Easy to implement!
  - » May involve lots of unnecessary seek distance
- Seek order = 100, 175, 51, 133, 8, 140, 73, 77
- Seek distance = (100-63) + (175-100) + (175-51) + (133-51) + (133-8) + (140-8) + (140-73) + (77-73) = 646 cylinders



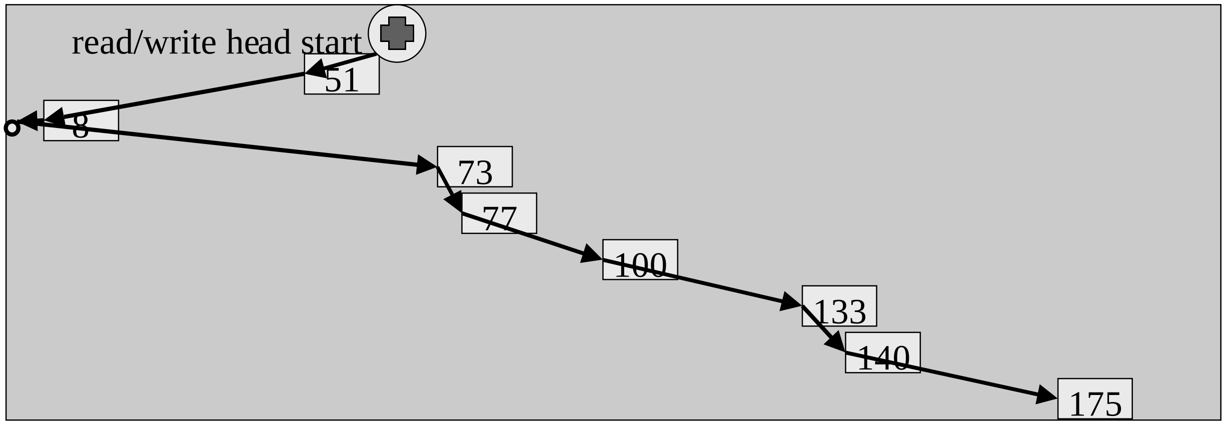
Shortest Seek Time First (SSTF)

- Service the request with the shortest seek time from the current head position
  - » Form of SJF scheduling
  - » May starve some requests
- Seek order = 73, 77, 51, 8, 100, 133, 140, 175
- Seek distance = 10 + 4 + 26 + 43 + 92 + 33 + 7 + 35 = 250 cylinders



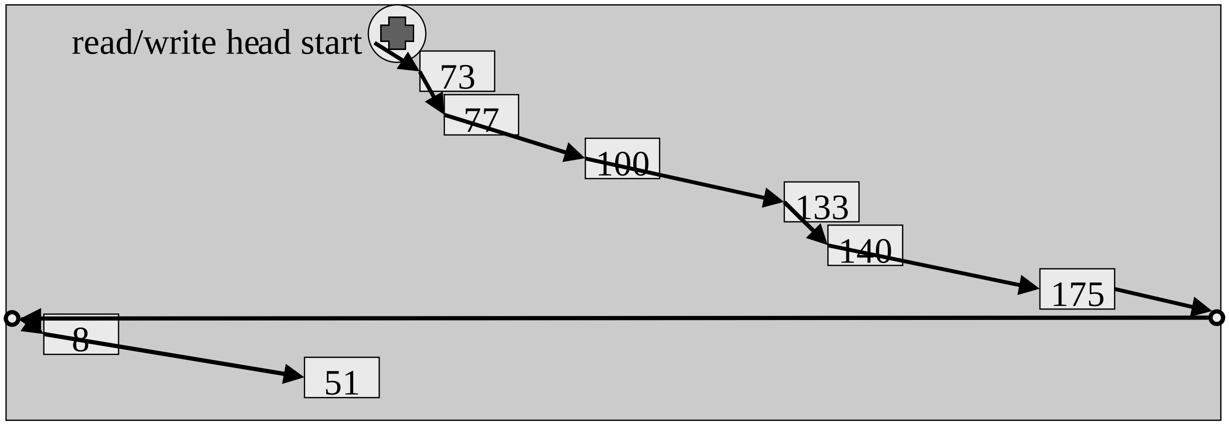
SCAN

- Disk arm starts at one end of the disk and moves towards the other end, servicing requests as it goes
  - » Reverses direction when it gets to end of the disk
  - » Also known as elevator algorithm
- Seek order = 51, 8, 0 , 73, 77, 100, 133, 140, 175
- Seek distance = 12 + 43 + 8 + 73 + 4 + 23 + 33 + 7 + 35 = 238 cyls



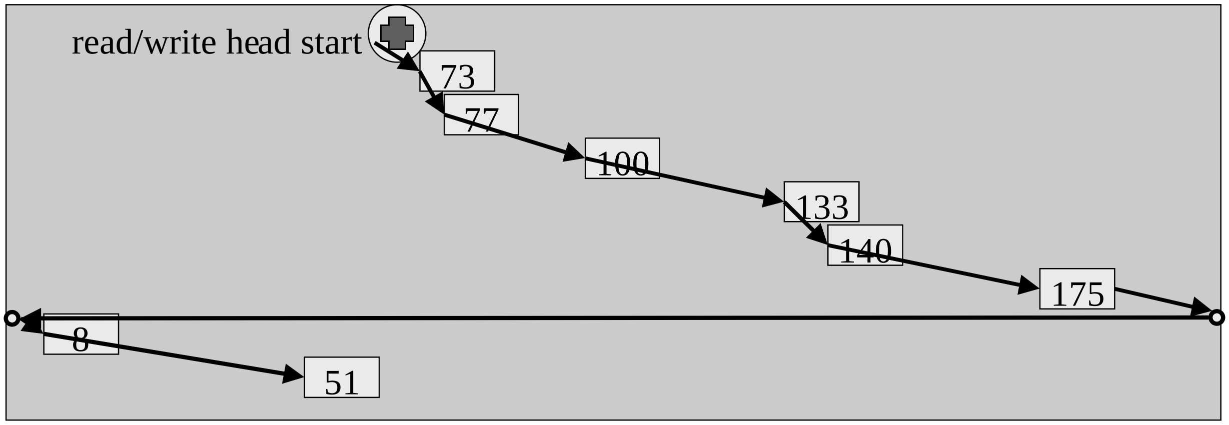
C-SCAN

- Identical to SCAN, except head returns to cylinder 0 when it reaches the end of the disk
  - » Treats cylinder list as a circular list that wraps around the disk
  - » Waiting time is more uniform for cylinders near the edge of the disk
- Seek order = 73, 77, 100, 133, 140, 175, 199, 0, 8, 51
- Distance = 10 + 4 + 23 + 33 + 7 + 35 + 24 + 199 + 8 + 43 = 386 cyls



C-LOOK

- Identical to C-SCAN, except head only travels as far as the last request in each direction
  - » Saves seek time from last sector to end of disk
- Seek order = 73, 77, 100, 133, 140, 175, 8, 51
- Distance = 10 + 4 + 23 + 33 + 7 + 35 + 167 + 43 = 322 cylinders



How to Pick a Disk Scheduling Algorithm

- SSTF is easy to implement and works OK if there aren’t too many disk requests in the queue
- SCAN-type algorithms perform better for systems under heavy load
  - » More fair than SSTF
  - » Use LOOK rather than SCAN algorithms to save time
- Long seeks aren’t too expensive, so choose C-LOOK over LOOK to make response time more even
- Disk request scheduling interacts with algorithms for allocating blocks to files
- Make scheduling algorithm modular: allow it to be changed without changing the file system
- Typically, use SSTF for lightly loaded systems and C-LOOK for heavily loaded systems

Low-Level Disk Management

- Formatting a disk
  - » Physical formatting: dividing a disk into sectors so the controller can read them
  - » Logical formatting: placing the initial versions of the file system structures on the disk (usually done by a specialized program)
- Structures created by physical formatting
  - » Spare blocks to replace other blocks that go “bad”
  - » Identification info (used by controller) for individual sectors
- Structures created by logical formatting
  - » Directory area
  - » Free block information (all blocks free initially)
  - » File system information area (read to initialize FS on boot)
  - » Boot block
  - » Bootstrap loader

Booting a Computer From Disk

- ROM contains simple code to read first *n* blocks from the disk
  - » Code split between ROM & disk varies by computer/OS
  - » Booting from network disk done in the same way
- First blocks of disk contain a program that knows how to read the operating system kernel off the disk
  - » Must know basic info about file system
  - » Need not be able to do everything (no need to write data, create files, cache data, etc.)
- Operating system starts execution
  - » Initializes file system by reading file system info block off disk
  - » Checks and fixes file system (if necessary) before making it available

Managing Swap Space (VM on Disk)

- Virtual memory uses disk to store overflow from main memory
  - Space consumed is called <u>swap space</u>
  - Swap space can be stored in
    - Special disk partition dedicated to swap space
    - Special file in the normal file system
  - Maximum allowable swap space set as OS parameter
- Swap space management is done by:
  - Allocating swap space when process starts
    - Text segment (program code): some OS simply point to the executable in the file system instead
    - Data segment (program data)
  - Keeping track of swap space used by a process in swap maps
  - Writing pages to disk when forced out of physical memory: some OS only allocate space at this point, not when page is created

Disk Reliability Issues

- Disk failure rates are relatively low
  - MTBF (Mean Time Between Failures) is 250,000+ hours
    - Disks don’t actually last 30 years!
    - On average, failure rate per year is 1 in 30
  - Low failure rates + lots of disks = trouble!
- Disk striping places file system across multiple disks, resulting in higher bandwidth (more disks) and lower reliability
- Solution: use redundancy (RAID = Redundant Array of Inexpensive Disks)
  - <u>Mirroring</u>: write two copies of each block, one to each disk
  - <u>Block interleaved parity</u>: keep a checksum of corresponding sectors on a separate disk
    - Requires less overhead for protection
    - Can be slower to write

Dealing With OS Failures

- Problem: how can file system improve performance while guarding against OS crashes?
- Solution: write-ahead log (circular list on disk)
  - » File system writes a list of what it’s about to do in the log
  - » Operations in the log are of the form “allocate block 8431 and map it to block 18 of file 1533”
  - » File system performs the operations in any order
  - » File system optionally writes “operation complete” to the log
- On crash recovery
  - » Read the last few log entries to see what should have been done before the crash
  - » For each operation in the log
    - – If the operation in the log wasn’t done, do it
    - – If the operation was already done, do nothing