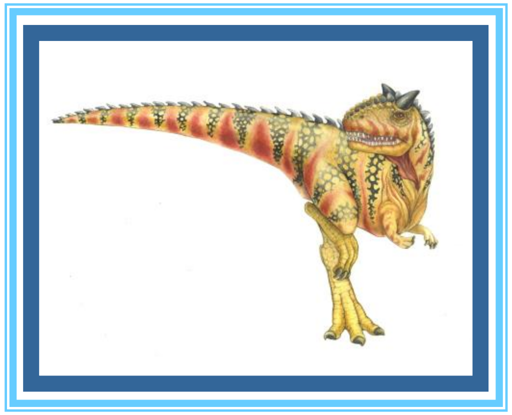
Chapter 11: Mass-Storage Systems



Chapter 11: Mass-Storage Systems

- Overview of Mass Storage Structure
- HDD Scheduling
- NVM Scheduling
- Error Detection and Correction
- Storage Device Management
- Swap-Space Management
- Storage Attachment
- RAID Structure

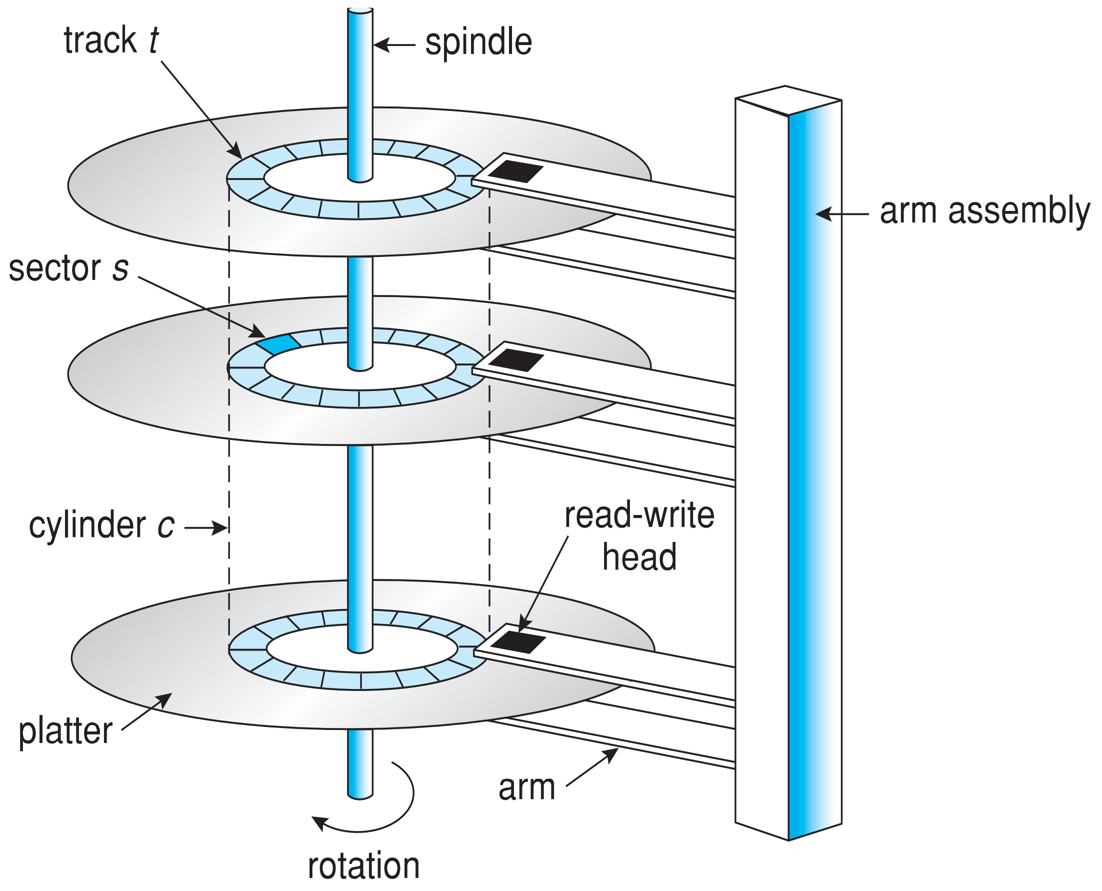
Objectives

- Describe the physical structure of secondary storage devices and the effect of a device’s structure on its uses
- Explain the performance characteristics of mass-storage devices
- Evaluate I/O scheduling algorithms
- Discuss operating-system services provided for mass storage, including RAID

Overview of Mass Storage Structure

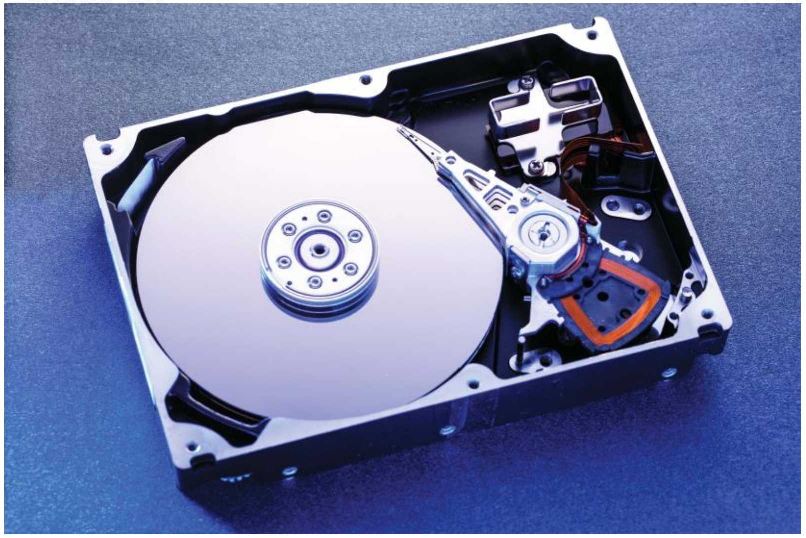
- Bulk of secondary storage for modern computers is **hard disk drives** (**HDDs**) and **nonvolatile memory** (**NVM**) devices
- **HDDs** spin platters of magnetically-coated material under moving read-write heads
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which 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 -- That's bad
- Disks can be removable

Moving-head Disk Mechanism



Hard Disk Drives

- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency



So, for an HDD with 7200 RPM, the average latency is 0.5 * 60/7200 = 0.00417 s = 4.17 ms

Hard Disk Performance

- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead
- For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

The First Commercial Disk Drive

1956
IBM 305 RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24" platters
Access time = < 1 second

Nonvolatile Memory Devices

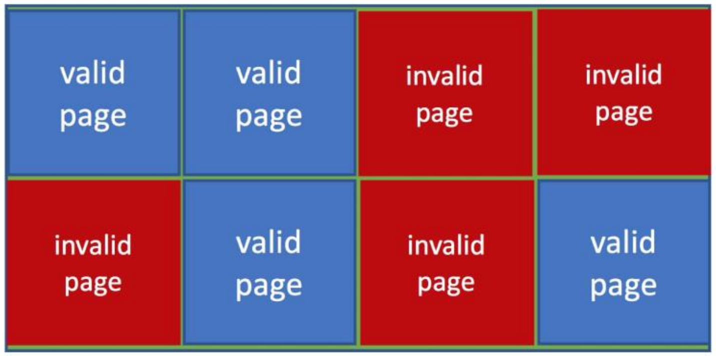
- If disk-drive like, then called **solid-state disks** (**SSDs**)
- Other forms include **USB drives** (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span – need careful management
- Less capacity
- But much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Nonvolatile Memory Devices

- Have characteristics that present challenges
- Read and written in “page” increments (think sector) but can’t overwrite in place
  - Must first be erased, and erases happen in larger ”block” increments
  - Can only be erased a limited number of times before worn out – ~ 100,000
  - Life span measured in **drive writes per day** (**DWPD**)
    - A 1TB NAND drive with rating of 5DWPD is expected to have 5TB per day written within warranty period without failing

NAND Flash Controller Algorithms

- With no overwrite, pages end up with mix of valid and invalid data
- To track which logical blocks are valid, controller maintains **flash translation layer** (**FTL**) table
- Also implements **garbage collection** to free invalid page space
- Allocates **overprovisioning** to provide working space for GC
- Each cell has lifespan, so **wear leveling** needed to write equally to all cells

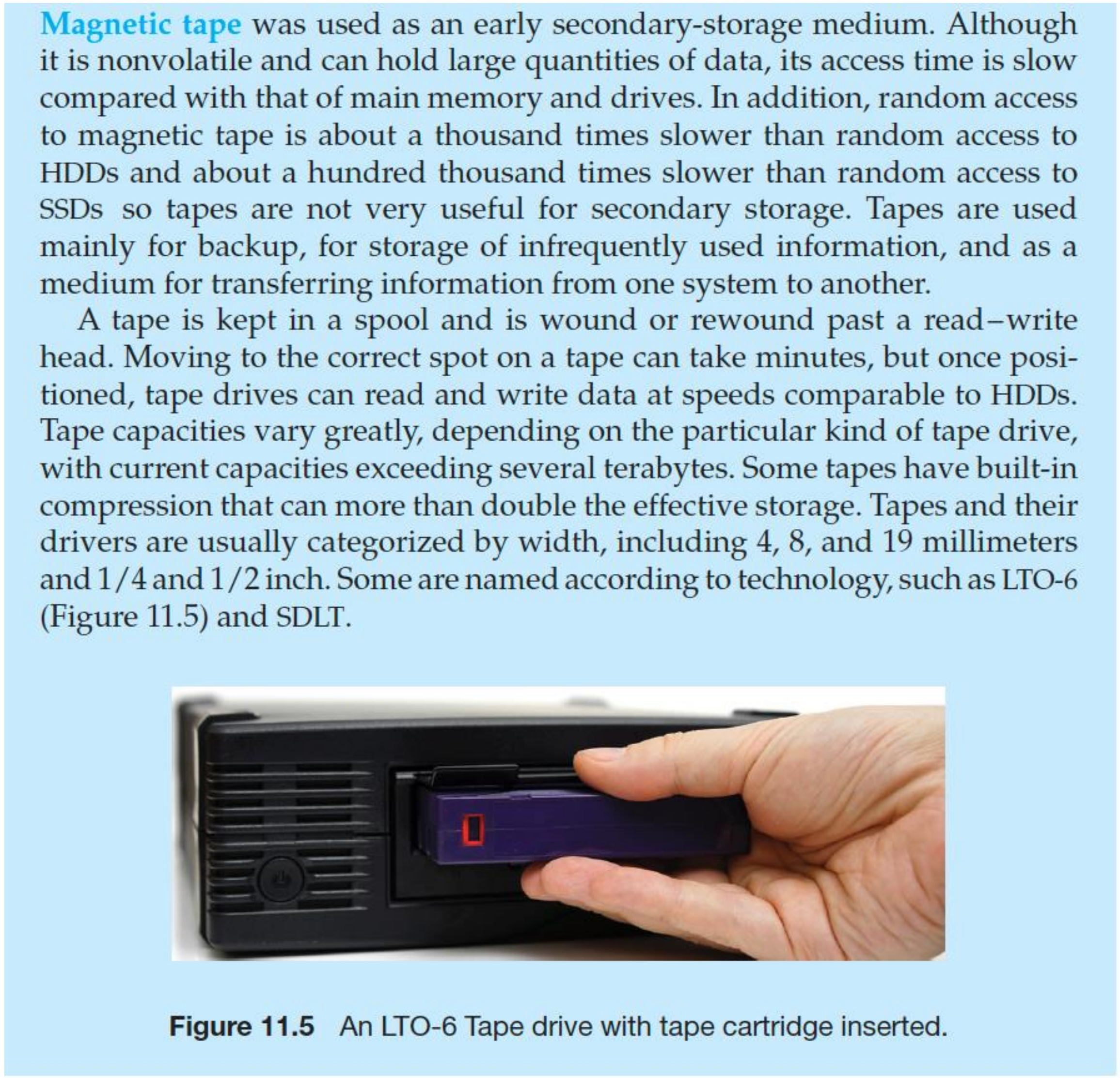


NAND block with valid and invalid pages

Volatile Memory

- DRAM frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, be used like very fast secondary storage
- **RAM drives** (with many names, including RAM disks) present as raw block devices, commonly file system formatted
- Computers have buffering, caching via RAM, so why RAM drives?
  - Caches / buffers allocated / managed by programmer, operating system, hardware
  - RAM drives under user control
  - Found in all major operating systems
    - Linux `/dev/ram`, macOS `diskutil` to create them, Linux `/tmp` of file system type `tmpfs`
- Used as high speed temporary storage
  - Programs could share bulk data, quickly, by reading/writing to RAM drive

Magnetic Tape



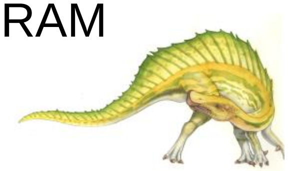
Magnetic tape was used as an early secondary-storage medium. Although it is nonvolatile and can hold large quantities of data, its access time is slow compared with that of main memory and drives. In addition, random access to magnetic tape is about a thousand times slower than random access to HDDs and about a hundred thousand times slower than random access to SSDs so tapes are not very useful for secondary storage. Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

A tape is kept in a spool and is wound or rewound past a read–write head. Moving to the correct spot on a tape can take minutes, but once positioned, tape drives can read and write data at speeds comparable to HDDs. Tape capacities vary greatly, depending on the particular kind of tape drive, with current capacities exceeding several terabytes. Some tapes have built-in compression that can more than double the effective storage. Tapes and their drivers are usually categorized by width, including 4, 8, and 19 millimeters and 1/4 and 1/2 inch. Some are named according to technology, such as LTO-6 (Figure 11.5) and SDLT.

Figure 11.5 An LTO-6 Tape drive with tape cartridge inserted.

Disk Attachment

- Host-attached storage accessed through I/O ports talking to **I/O busses**
- Several busses available, including **advanced technology attachment** (**ATA**), **serial ATA** (**SATA**), **eSATA**, **serial attached SCSI** (**SAS**), **universal serial bus** (**USB**), and **fibre channel** (**FC**).
- Most common is SATA
- Because NVM is much faster than HDD, new fast interface for NVM called **NVM express** (**NVMe**), connecting directly to PCI bus
- Data transfers on a bus carried out by special electronic processors called **controllers** (or **host-bus adapters**, **HBAs**)
  - Host controller on the computer end of the bus, device controller on device end
  - Computer places command on host controller, using memory-mapped I/O ports
    - Host controller sends command messages to device controller
    - Data transferred via DMA between device and computer DRAM



Address Mapping

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - 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
  - Logical to physical address should be easy, but
    - For bad sectors (re-mapped to elsewhere on the drive)
    - Non-constant # of sectors per track via constant linear velocity
    - Disk controllers now internally manage the mapping between logical block address (LBA) to physical sectors

HDD Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS, system processes, and user processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue (common in a multiprogramming system)
  - Optimization algorithms only make sense when a queue exists
- In the past, operating system responsible for queue management, and translating an LBA to track/head/sector
  - Now, LBA-to-physical address mapping is built into the storage devices, controllers
  - OS just provides LBAs, handles sorting of requests (sequential I/O)
    - Some of the algorithms they use are described next

Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

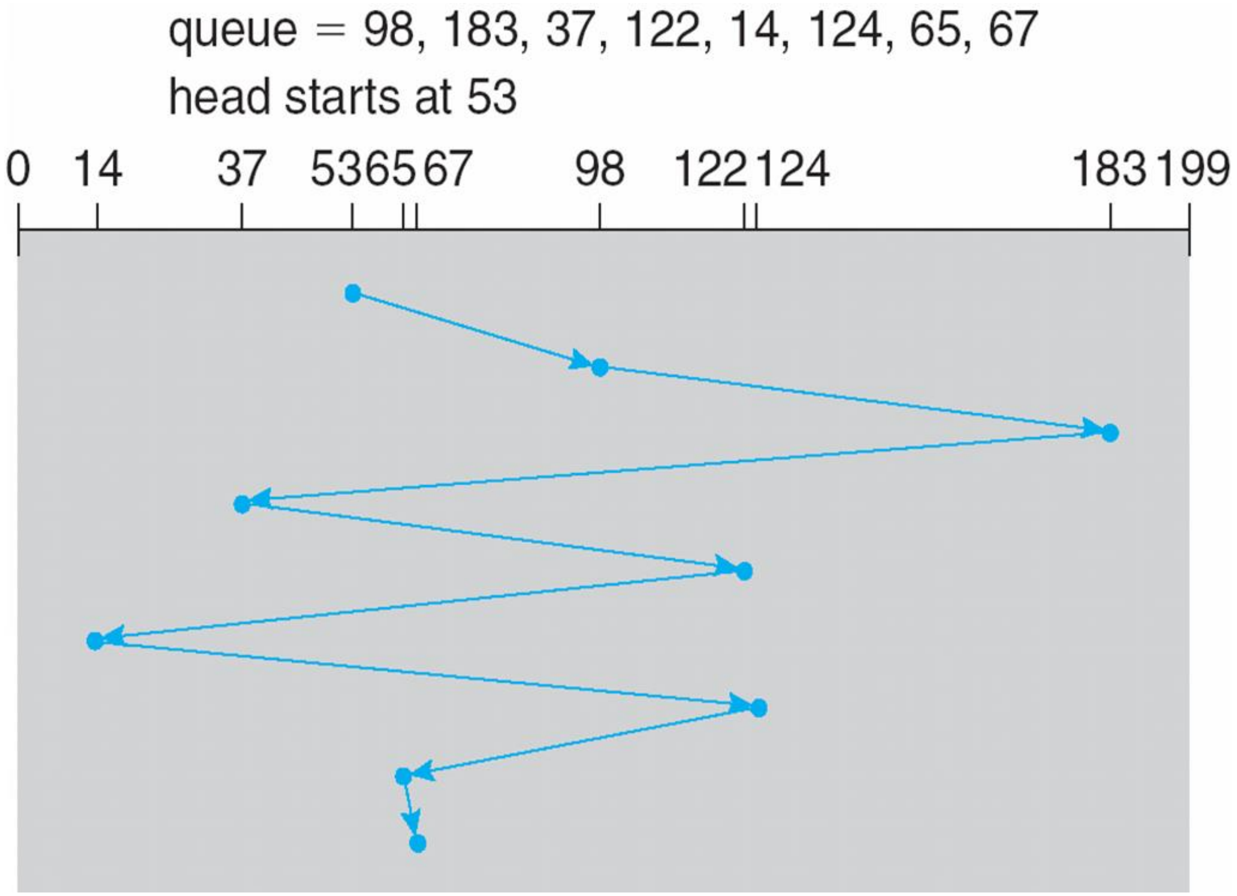
98, 183, 37, 122, 14, 124, 65, 67

Disk head is initially at cylinder 53

FCFS

Illustration shows total head movement of 640 cylinders

45+85+146+85+108+110+59+2 = 640



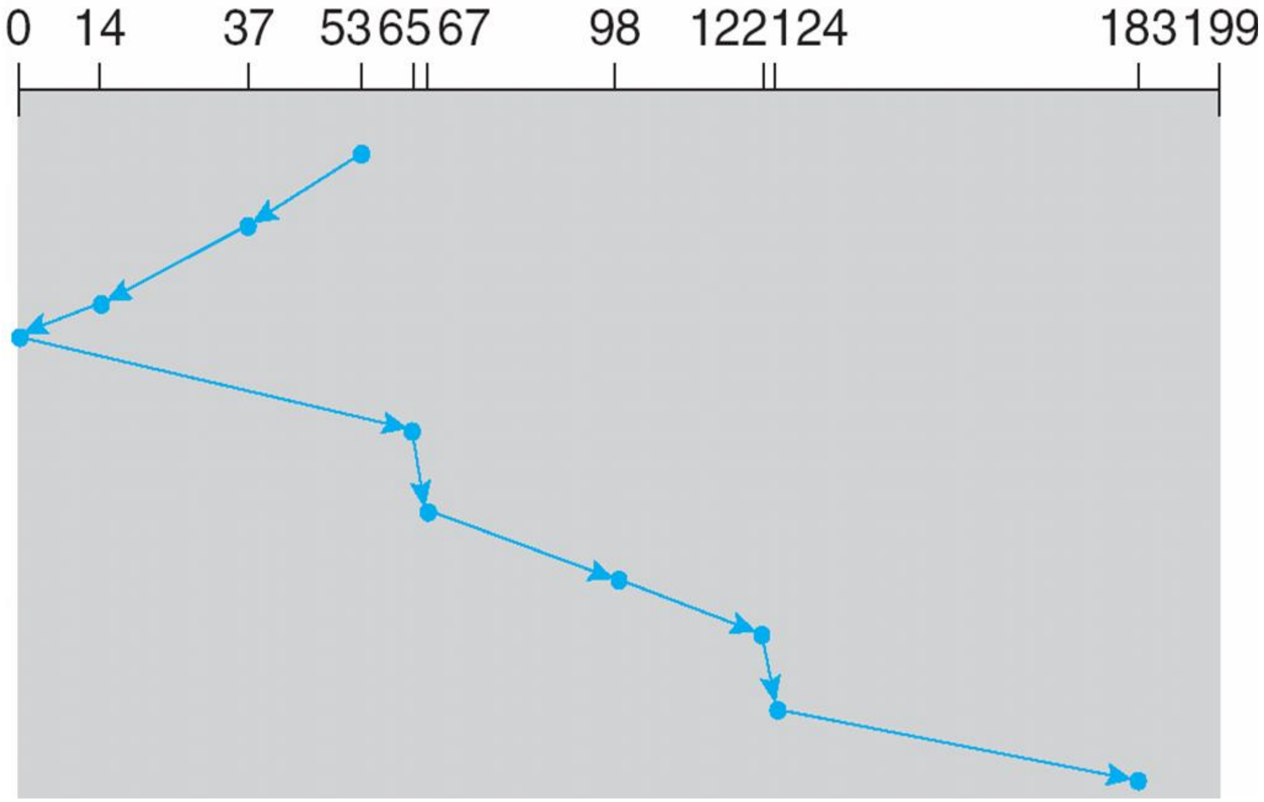
SCAN

- 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.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 208 cylinders (next slide)
  - 53 to 14 (39), then 14 to 183 (169)
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

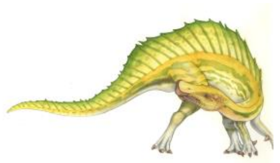
queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53 moving toward track 0 but may only need to get to 14

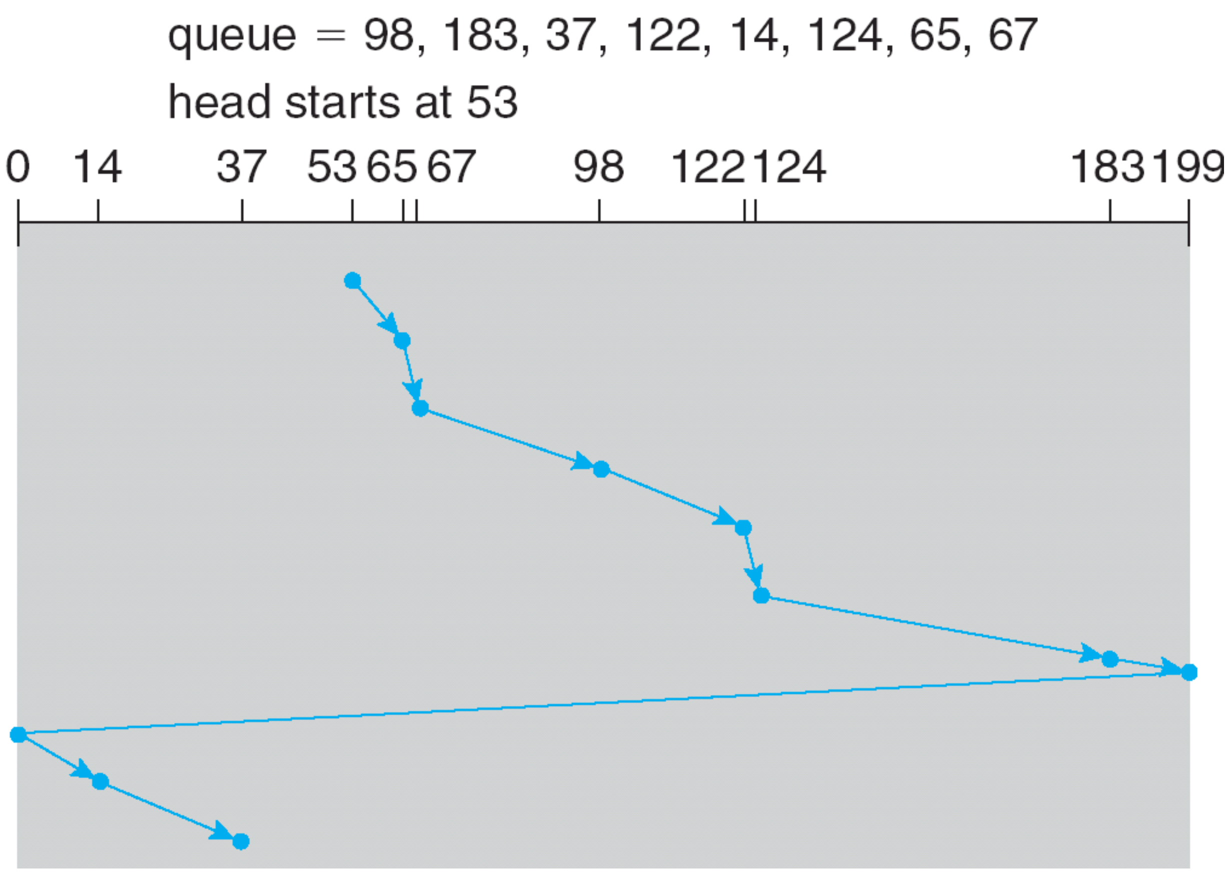


C-SCAN

- Provides a more uniform wait time than 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
- Total number of cylinders?



C-SCAN (Cont.)



queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Selecting a Disk-Scheduling Algorithm

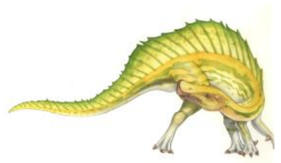
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - No starvation, but long wait still possible
- To avoid starvation Linux implements **deadline** scheduler
  - Maintains separate read and write queues, gives read priority (shorter deadline, 500ms for read vs 5s for write)
    - Because processes more likely to block on read than write
  - Implements four queues: 2 x read and 2 x write
    - 1 read and 1 write queue sorted in LBA order, essentially implementing C-SCAN
    - 1 read and 1 write queue sorted in FCFS order with deadlines
    - All I/O requests sent in batch sorted in that queue's order
    - After each batch, checks if any requests in FCFS older than configured age (default 500ms/5s)
      - If so, LBA queue containing that request is selected for next batch of I/O
- In RHEL 7, **NOOP** and **completely fair queueing** scheduler (**CFQ**) also available, defaults vary by storage device

NVM Scheduling

- No disk heads or rotational latency but still room for optimization
- In RHEL 7 **NOOP** (no scheduling) is used but adjacent LBA requests are combined
  - NVM best at random I/O, HDD at sequential
  - Sequential throughput can be similar
  - On random access I/O, measured in **Input/Output operations per second** (**IOPS**), it's much higher with NVM (hundreds of thousands vs hundreds for HDD)
  - But NVM's writing is less consistent depending on how full the device is
    - **write amplification** (a write causing garbage collection and many read/writes can decrease the performance advantage of NVM

Error Detection and Correction

- Fundamental aspect of many parts of computing (memory, networking, storage)
- **Error detection** determines if a problem has occurred (for example a bit flipping)
  - If detected, can halt the operation before error propagation
  - Detection frequently done via parity bit (0 for even 1's)
- Parity is one form of **checksum** – uses modular arithmetic to compute, store, compare values of fixed-length words
  - Another error-detection method common in networking is **cyclic redundancy check** (**CRC**) which uses hash function to detect multiple-bit errors
- **Error-correction code** (**ECC**) not only detects, but can correct some errors
  - Soft errors correctable, hard errors detected but not corrected



Storage Device Management

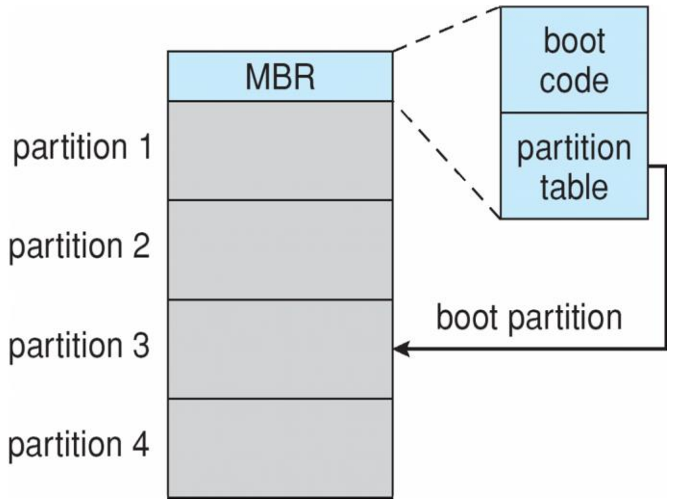
- **Low-level formatting**, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write
  - Each sector can hold header information, plus data, plus error correction code (**ECC**)
  - Usually 512 bytes of data but can be selectable
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
  - **Partition** the disk into one or more groups of cylinders, each treated as a logical disk
  - **Logical formatting** or "making a file system"
    - Create maps of free and allocated space and a root directory
  - To increase efficiency most file systems group blocks into **clusters**
    - Disk I/O done in blocks
    - File I/O done in clusters

Storage Device Management (cont.)

- **Root partition** contains the OS, other partitions can hold other OSes, other file systems, or be raw (for swap space or database)
  - **Mounted** at boot time
  - Other partitions can mount automatically or manually
- At mount time, file system consistency is checked
  - Is all metadata correct?
    - If not, fix it, try again
    - If yes, add to mount table, allow access
- Boot block can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
  - Or a boot management program for multi-OS booting

Device Storage Management (Cont.)

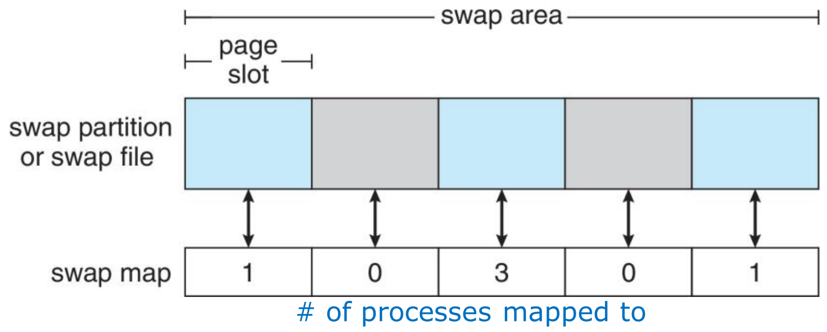
- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
    - The bootstrap is stored in ROM, firmware
    - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks



Booting from secondary storage in Windows

Swap-Space Management

- Used for moving entire processes (swapping), or pages (paging), from DRAM to secondary storage when DRAM not large enough for all processes
- Operating system provides **swap space management**
  - Secondary storage slower than DRAM, so important to optimize performance
  - Usually multiple swap spaces possible – decreasing I/O load on any given device
  - Best to have dedicated devices
  - Can be in raw partition or a file within a file system (for convenience of adding)
  - Data structures for swapping on Linux systems:



End of Chapter 11