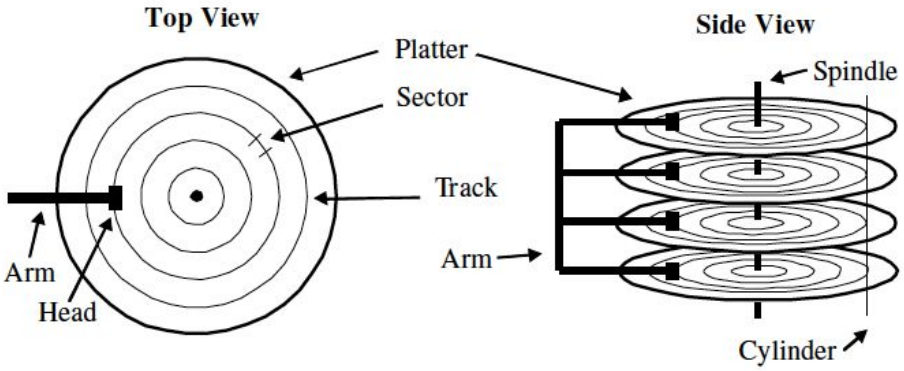
Topic 16: Files

Reading: 12.1, 13.3 - 13.3.2
Next reading: 13.2

- *File*: a named collection of bytes stored on disk. From the OS's standpoint, the file consists of a bunch of fixed-size *blocks* stored on the device. Processes may actually see a different interface (e.g., byte-stream, records), but this doesn't matter to the file system (just pack bytes into disk blocks, unpack them again on reading).
- Disk characteristics:



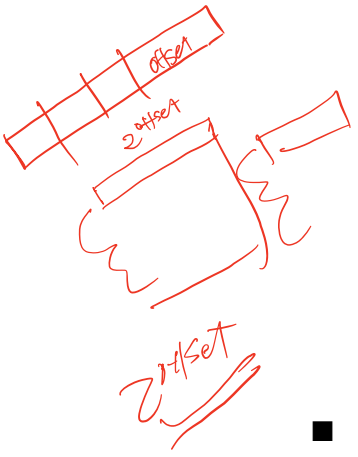
-
  - Unit of storage is a *sector*. Sectors are stored on *platters* in concentric rings called *tracks*. The set of the same track on different platters is called a *cylinder*.
  - Sectors are read and written by a magnetic *head* connected to a movable *arm*. There is one head per platter surface.
- Why does it take milliseconds to access a sector?

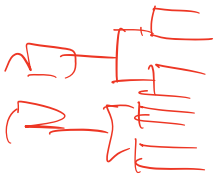
    - *Seek time*: wait for arm to move to correct cylinder
        - 0.3-10 ms
    - *Rotational latency*: wait for desired sector to reach head.
        - On average must wait 1/2 rotation: 2-8 ms.
    - *Transfer time*: wait for bits to be transferred between media and computer.
        - Media rate: > 100MB/sec
        - Bus (SATA, FC, USB) rate: 60-2500MB/sec
        - Burst numbers often reported.
    - Disk performance trends:
        - Focus is on capacity, not performance.
        - Smaller -- faster RPM, shorter seeks.
        - Track buffer -- store entire track (or more) in RAM.
- Disk performance is much, much worse than CPU performance. What's a file system to do?
    - Avoid using the disk.
    - Use the disk intelligently.
- Don't use the disk if you can avoid it. Instead, *cache* file data in main memory. This is called a *block cache*.
    - Store popular blocks in memory, eliminating disk accesses.

- Short-lived file data can be filtered out completely (folding laundry principle).

- Hide disk latency by overlapping computation with I/O.
  - *Read-ahead* blocks that will be read.
  - *Write-behind* blocks that were written.
  - Read-ahead and write-behind increase queue length, which in turn makes scheduling effective.

- Disadvantages of a cache:
  - Requires RAM.
  - Contents are lost in a crash. This is a problem for blocks that are written to the cache but not to the disk. There are several solutions:
    - *Write-through*: write blocks to disk and cache at same time.
    - *Write-through-on-close*: improves performance of applications that write the same blocks many times.
    - *Bounded write-back*: limit amount of time new data can live in cache before they are written to disk. 30 seconds is the UNIX standard.
  - *Cache consistency*: an inconsistency between the cache contents and the disk can lead to stale data errors.





- Modern file systems must address four general problems:
  - *Disk Management*: efficient use of disk space, fast access to files, sharing of space between several users.
  - *Naming*: how do users select files?
  - *Protection*: all users are not equal.
  - *Reliability*: information must last safely for long periods of time.
- Information about each file is represented by a *file record* stored on the disk.
  - This information is referred to as *file metadata*.
  - Information includes the locations of the file's blocks on the disk as well as other *attributes*.

    name not stored here to save space

    - File size
    - Owner/group
    - Protection information
    - Create/modify/access times
    - Reference count (how many names the file has) delete if 0
    - (other attributes depending on filesystem)