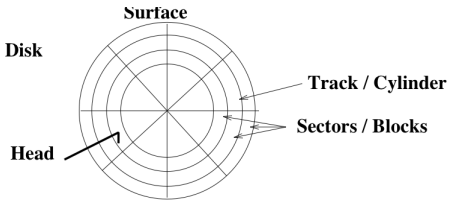
Today: Secondary Storage



- **To read or write a disk block:**
  - **Seek:** (latency) position head over a track/cylinder. The seek time depends on how fast the hardware moves the arm.
  - **Rotational delay:** (latency) time for the sector to rotate underneath the head. Rotational delay depends upon how fast the disk spins.
  - **Transfer time:** (bandwidth) time to move the bytes from the disk to memory
  - **Disk I/O Time** = seek + rotational delay + transfer.

Typical Disk Parameters

	SATA disk	SCSI disk
Disk Capacity	160 GB	146GB
Platters per pack	16	8
Tracks per surface	16,383	6,358
Sectors per track	63	644
Bytes per sector	512	732
Revolutions per minutes	7200	15,000
Average seek time	4 ms	<4ms
Average rotational latency	4.17 ms	2 ms
Buffer to host burst transfer rate	78 MB/sec	85 MB/sec
Buffer size	8 MB	4 MB
size	3.5 inches	3.5 inches

Access Time

- **Key:** to get the quickest disk response time, we must minimize seek time and rotational latency:
  - Make disks smaller
  - Spin disks faster
  - Schedule disk operations to minimize head movement
  - Lay out data on disk so that related data are on nearby tracks.
  - Place commonly-used files where on the disk?
  - We should also pick our sector size carefully:
    - If the sector size is too small, we will have a low transfer rate because we will need to perform more seeks for the same amount of data.
    - If our sector size is too large, we will have lots of internal fragmentation.
- NOTE: Solid state drives (SSD) will eliminate these problems.

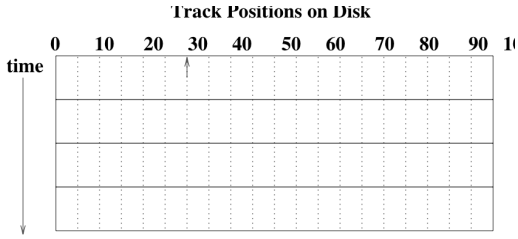
Disk Head Scheduling

- **Idea:** Permute the order of disk requests from the order that they arrive from the users to an order that reduces the length and number of seeks.
  1. First-come, first-served (FCFS)
  2. Shortest seek time first (SSTF)
  3. SCAN algorithm (0 to 100, 100 to 0, 0 to 100, ...). If there is no request between current position and the extreme (0 or N), we don't have to seek there.
  4. C-SCAN circular scan algorithm (0 to 100, 0 to 100, ...)

FCFS Disk Head Scheduling

Example requests: 65, 40, 18, 78

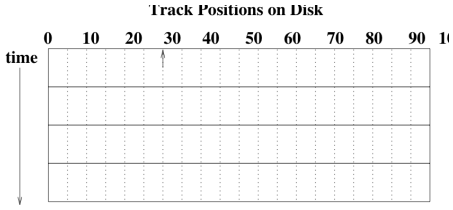
1. FCFS - service the requests in the order that they come in



- Order of seeks: 65, 40, 18, 78
- Distance of seeks: 35 + 25 + 22 + 60 = 142
- When would you expect this algorithm to work well?
  - During light loads ; also for SSD drives

SSTF Disk Head Scheduling

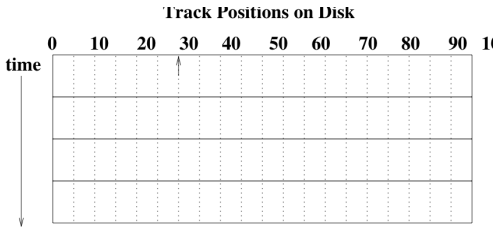
- SSTF: always go to the next closest request



- Order of seeks: 40, 18, 65, 78
- Distance of seeks: 10 + 22 + 47 + 13 = 92
- Can implement this approach by keeping a doubly linked sorted list of requests.
- Is this efficient enough?
- Is it optimal? Greedy (minimizes seek but not overall cost)
- Problems? Potential for starvation

SCAN Disk Head Scheduling

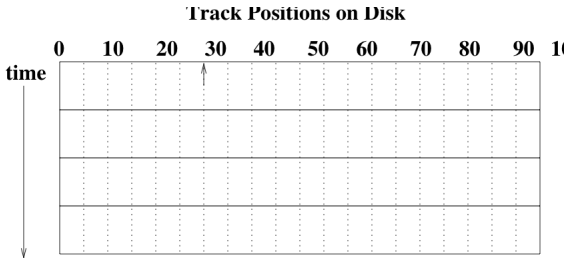
- SCAN: head moves back and forth across the disk (0 to 100, 100 to 0, 0 to 100, ...), servicing requests as it passes them



  - Order of seeks, assuming the head is currently moving to lower numbered blocks: 18, 40, 65, 78
  - Distance of seeks: 12 + 22 + 25 + 13 = 72
  - Requires a sorted list of requests.
  - Simple optimization does not go all the way to the edge of the disk each time, but just as far as the last request.

C-SCAN Disk Head Scheduling

- C-SCAN: circular scan algorithm (0 to 100, 0 to 100, ...)



  - Order of seeks: 40, 65, 78, 18
  - Distance of seeks: 10 + 25 + 13 + 60 = 108
  - More uniform wait times for requests. Why?
    - At the end of a pass, most requests are likely to be at the other end. With SCAN, these requests get serviced at the end of the sweep.

Improving Disk Performance using Disk Interleaving

- *Problem:* Contiguous allocation of files on disk blocks only makes sense if the OS can react to one disk response and issue the next disk command before the disk spins past the next block.

- *Idea:* Interleaving - Don't allocate blocks that are physically contiguous, but those that are temporally contiguous relative to the speed with which a second disk request can be received and the rotational speed of the disk. Might use every second or third block.

Improving Disk Performance using Read Ahead

- **Idea:** read blocks from the disk ahead of user's request and place in buffer on disk controller.

- **Goal:** reduce the number of seeks - read blocks that will probably be used while you have them under the head.

- We considered pre-fetching virtual pages into physical memory, but decided that was difficult to do well since the future is difficult to predict. Is disk read-ahead any better?

Tertiary Storage

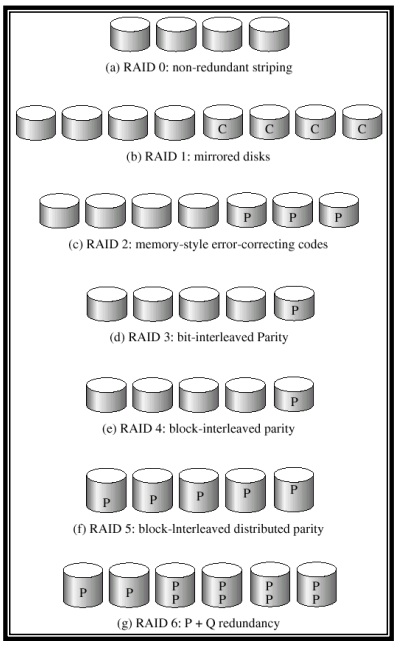
- Lower cost devices than secondary storage (disks)
- Typically Slower, larger, cheaper than disks
- Used primarily for storing archival data or backups.
  - tape drives
  - Jazz and Zip drives
  - Optical disks: Write once read-many (WORM), CD-R, CD-RW
  - Robotic jukeboxes
- Primary, secondary and tertiary devices form a storage hierarchy
- Falling cost of hardware ➔ tapes replaced by (slower) disks

Tapes

- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower.
- Tape is an economical medium for purposes that do not require fast random access, e.g., backup copies of disk data, holding huge volumes of data.
- Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library.
  - stacker – library that holds a few tapes
  - silo – library that holds thousands of tapes
- A disk-resident file can be *archived* to tape for low cost storage; the computer can *stage* it back into disk storage for active use.

RAID Storage

- **RAID** – multiple disk drives provides **reliability** via **redundancy**.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.
- RAID is arranged into six different levels



Summary

- Disks are slow devices relative to CPUs.
- For most OS features, we are very concerned about efficiency.
- For I/O systems, and disk, in particular, it is worthwhile to complicate and slow down the OS if we can gain improvement in I/O times.
- **Review Questions:**
  - What property of disks can we use to make the insertion, deletion, and access to the lists of requests fast?
  - Rank the algorithms according to their expected seek time.
  - Is SCAN or SSTF fairer?
  - Is SCAN or C-SCAN fairer?