Accessing files on disks

- Structure of a disk:
  - Disk head
  - Platter (2 surfaces)
  - Cylinder = all tracks at same distance

Structure of a Disk Surface

- 1 track
- 1 sector
- Gaps help identify beginnings and sectors.

2008: ~100,000 tracks per inch
* Blocks and Directory and files.

Block = logical mapping of sectors into larger (more efficient) units to store files.

Eg: block 4, sector 11

0 1 2 3
1
2
4 5 6 7
8 9 10 11

Directory blocks = blocks reserved for storing directory entries

Directory entries contains

1) file name
2) block numbers of the file
3) other info (eg. owner, permission bits etc.)

Directory = all directory entries stored in the directory blocks (contains data about the files).
Example:

Directory Blocks:

file 1

file 2

Blocks of file

filename 1 567 789...
The Disk Controller

Connecting Disks to a computer system:

```
  +-------------------+  +-----------------+
  |                  |  |                 |
  |              CPU |  |        Main Memory |
  +-------------------+  +-----------------+
      (Data Bus)       (Command Lines)

Disk Controller
```

Disk controller: small (simple) processor that:

1. Controls movement of the disk heads (move head to a specific track).
2. Select a sector on a track.
3. Transfer data (bits) between disk surface and computer main memory.
4. Has an internal buffer to speed up data transfer (can buffer a track or more).
Disk Access Time.

Steps the disk takes to access the data on a specific sector:

1. Move the disk head(s) to the track containing the sector. ⇒
   - Seek time: $0 \sim 10$ msec
     - Some tracks one end to another.
     - Avg. seek time $\approx 5$ msec.

2. Wait for the sector to come under the disk head.
   ⇒ Rotational latency:
     - $0 \sim 10$ msec also.
     - Avg. wait latency $\approx 5$ msec.

3. Transfer data. ($< 1$ msec).
What do you take away from this:

1. Data processing proceeds piece meal:
   1 block at a time.

2. 1 block ~ 4 k = 64 k bytes.

3. Time needed to bring 1 block of data into main memory:
   ~ (estimate) 50 msec.

4. In 50 msec = 50,000 usec
   = 50,000,000 nsec.

A CPU (with good/large cache)
   can execute over 10,000,000 instructions
   More than enough to process 64 k byte of data!!

=) In DB applications: Disk access time will be the dominant factor!
Accelerate Access to Disk:

- Techniques used to speed up file access:

1. Place blocks that are accessed frequently together on the same cylinder.
   
   \[ \Rightarrow \text{because seek time} = 0 \]
   
   \[ \text{seek time} = \frac{1}{2} \text{ the total access time} \]

2. Use multiple smaller disks instead of one large disk.

   \[ \text{Multiple disks} \Rightarrow \text{multiple disk heads} \]

   \[ \text{more independently} \]

   \[ \Rightarrow \text{increase # blocks accessed per sec.} \]
Striping: store disk blocks of a file over many disk drives.

Examples:

\[
\begin{align*}
\text{R1} & \rightarrow \text{R5} \\
\text{R2} & \rightarrow \text{R6} \\
\text{R3} & \rightarrow \text{R7} \\
\text{R4} & \rightarrow \text{R8}
\end{align*}
\]

They can retrieve the data independently.

Controller can control all disks at the same time.

- Control signals are short.

Experience: with 4 disks, avg. speed up

\[
= 3 \times \text{speed of 2 disk.}
\]

(from book).

(Assumption: transfer speed between controller \( \rightarrow \) disk is high enough to support n simultaneous transfers.)
3) Mirror disks. Very expensive ... get added reliability. All disks contain same data.

Read from any disk.

\[ \text{Read access speed} = n \times \text{speed of 1 disk} \]

\[ \text{BUT: write speed} \approx \text{same as 1 disk.} \]

Assumption: transfer speed

Controller \(\Rightarrow\) drive

Sufficiently large.
4) Use a disk-scheduling algorithm to reduce disk access time.
   (the scheduling alg. with re-order the disk requests !!!).

   ⇒ Commonly used disk scheduling alg:
   = The "Elevator" Algorithm.

Elevator Alg:

- Disk Head movement

  current track

  ⇒ Process ALL requests for sectors residing on the current track.

  ⇒ Move disk head to the next track (same direction) with request.

(Operate like an "elevator" pickup up people).
Prefetching blocks

In some applications (e.g., SRT - read all blocks), we can predict the order in which blocks will be requested.

$\Rightarrow$ Load blocks into main memory BEFORE they are requested!

$\Rightarrow$ Best combine with Disk scheduling algorithm (save seek time!!!)