# Data Management and File Structure

Introduction

# Topics

- Introduction to File Systems
- Problems in using Disks
- Disk Structure
- Disk I/O Timing Parameters
- Definitions
  - Blocks and Records
  - Buckets
  - Double Buffering
  - Blocking Factor

# Information Systems

- Many computer systems need to store a large amount of data.
- Examples are: Student information system, Hospital information system, etc.
- This information cannot be stored in computer memory because
  - Memory has a limited capacity
  - Information is lost when we turn off the computer

# How to store data?

- Data is stored in files.
- Files are stored on hard disks because disks:
  - Have larger capacity
  - Can store data even when we turn off the computer ( non-volatile)

# Problems in using disks

- Disks are very slow
  Typical time to read an integer from
  RAM = 60 nsec
  Disk = 6 msec
  RAM is 10 million times faster

# How to speed up I/O from a disk?

- For faster input/output we can organize data of the files.
- File structure aim is to develop file formats for faster input/output operations

Example: Sorting files
Using Indexes
Hashing

# Sorting Files

Advantage:

Search in a sorted file is faster (binary search)

Disadvantage:

Keeping file sorted is difficult (insert, update)

# Indexing

- Index is a list, showing the location of records in data files
- For faster indexing, trees are used (example B+tree)

# Hashing

- Hashing refers to methods for finding the location of records in data files
- Hashing is faster than indexing

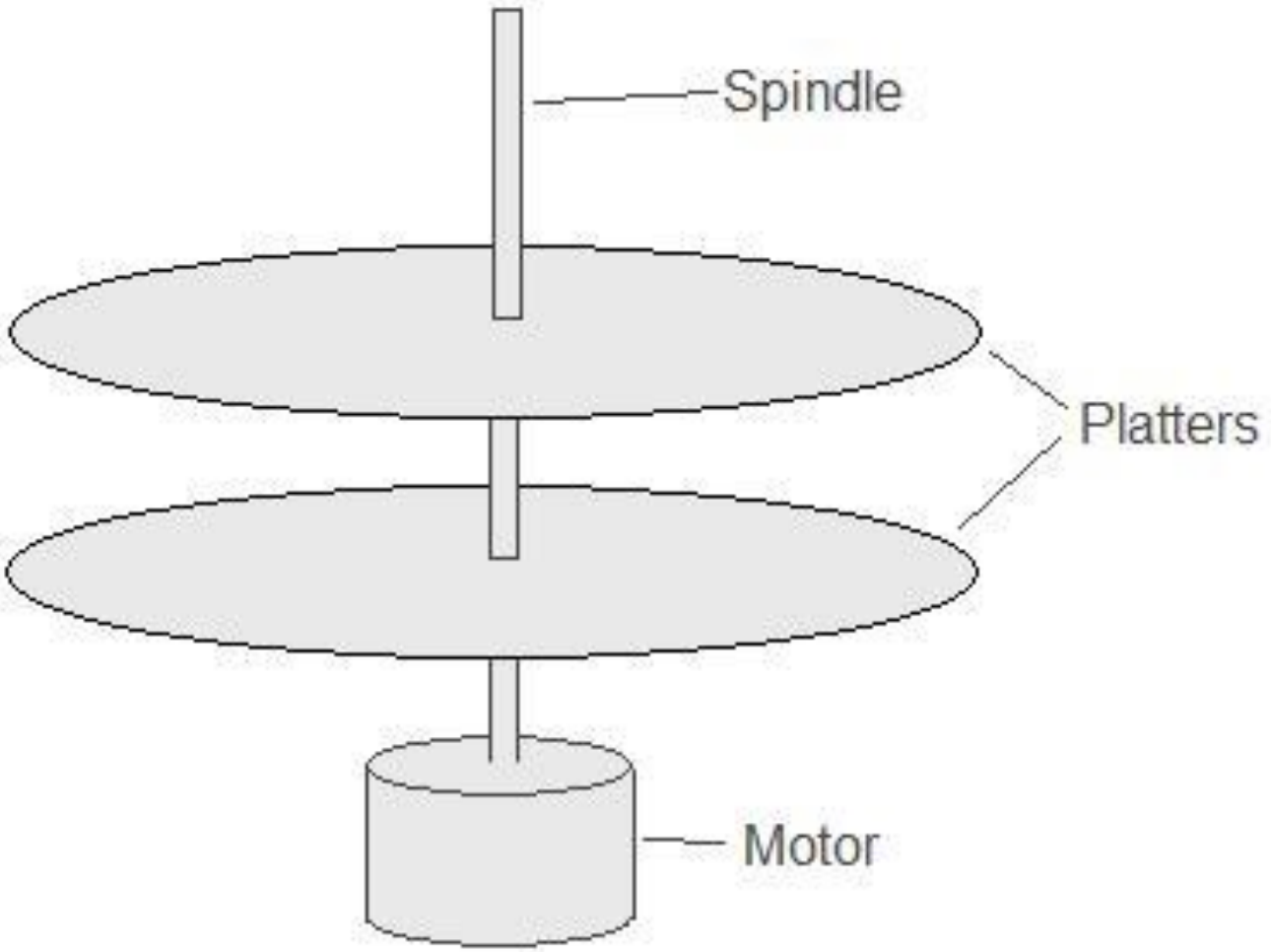
# Disks

- Disks are slow compared to RAM
- Disk I/O can be optimized by organizing data of the files.

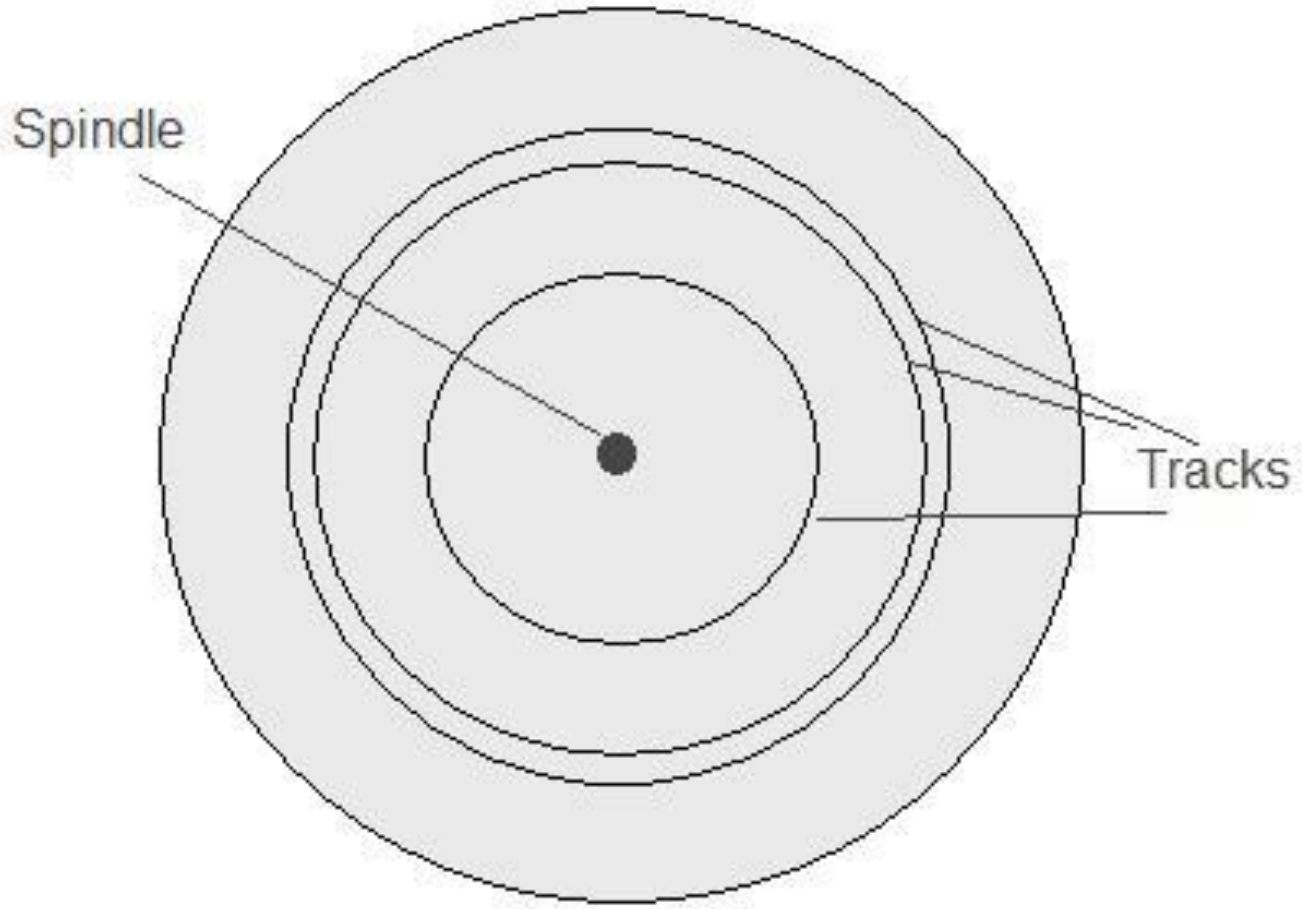
# Disk Structure

- Disks have
  - platters to store data
  - spindle to hold platters
  - Motor to turn spindle and hence platters
  - Head to read/write data
  - Arm to hold and move head

# Disk Structure

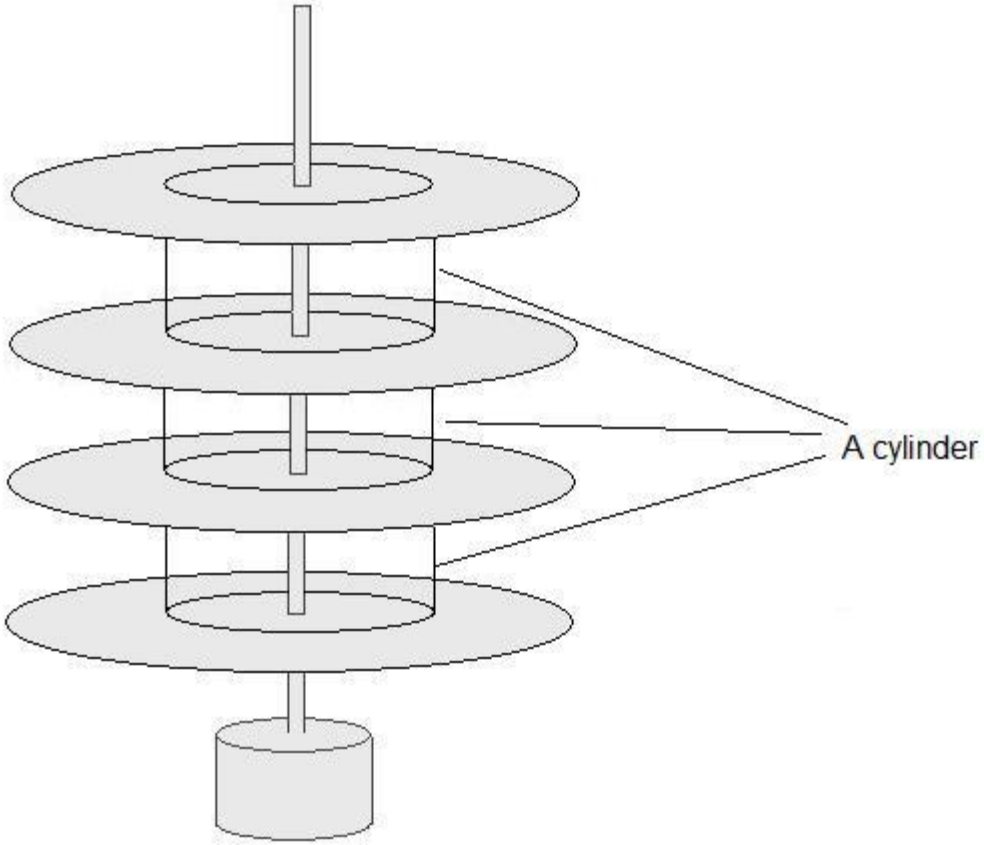


# Tracks

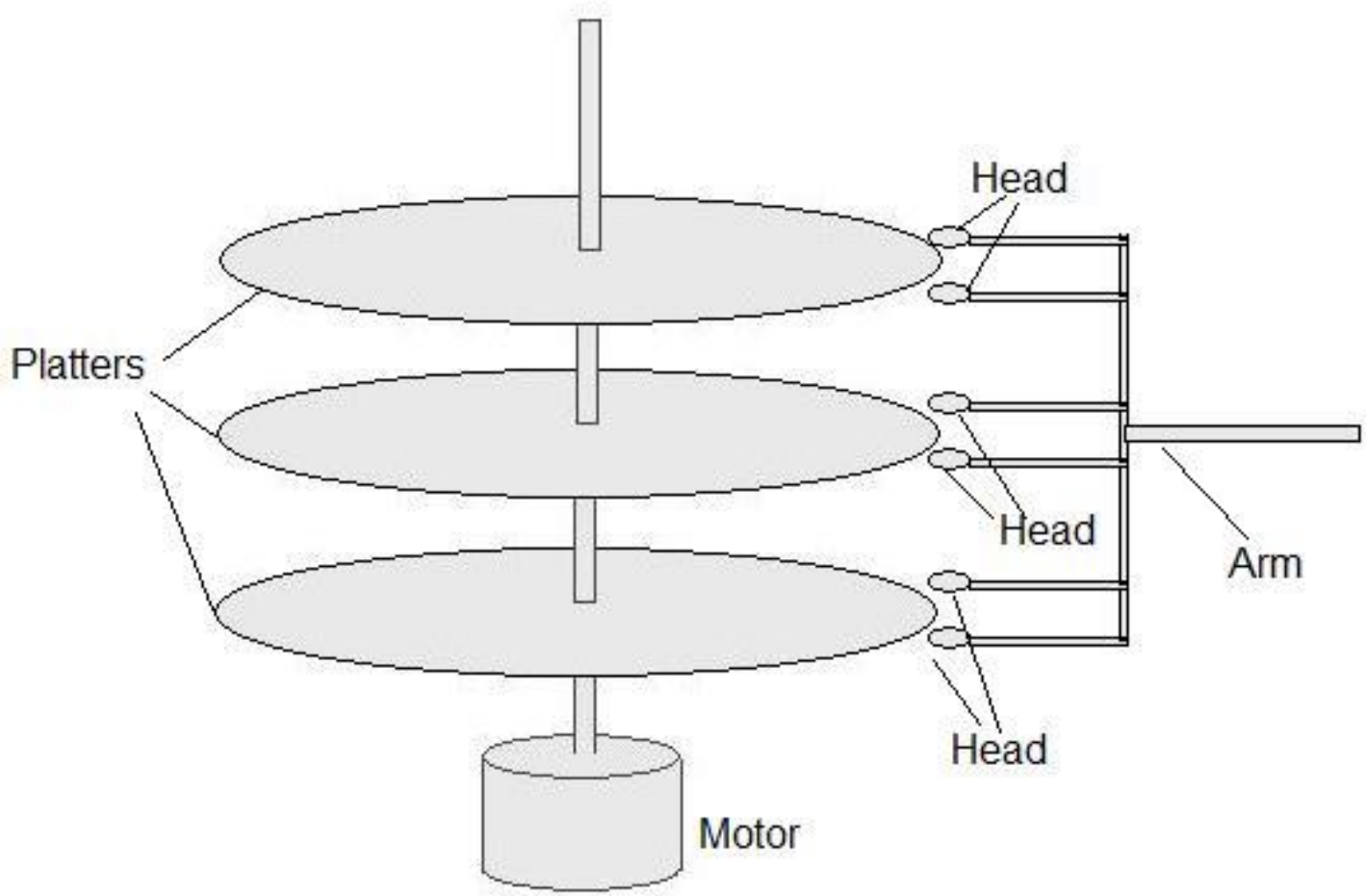


# Cylinder

- Tracks of different platters with the same distance from the center (spindle)

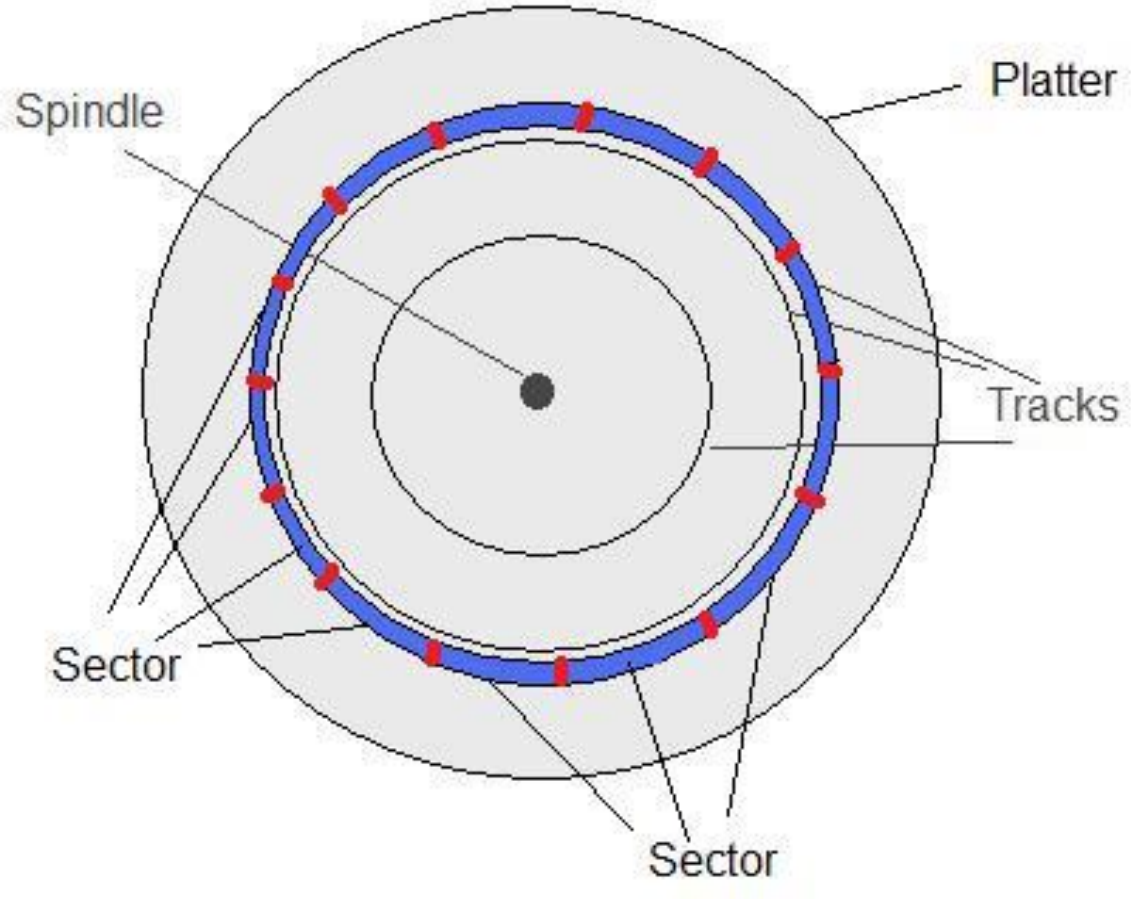


# Heads and Arm



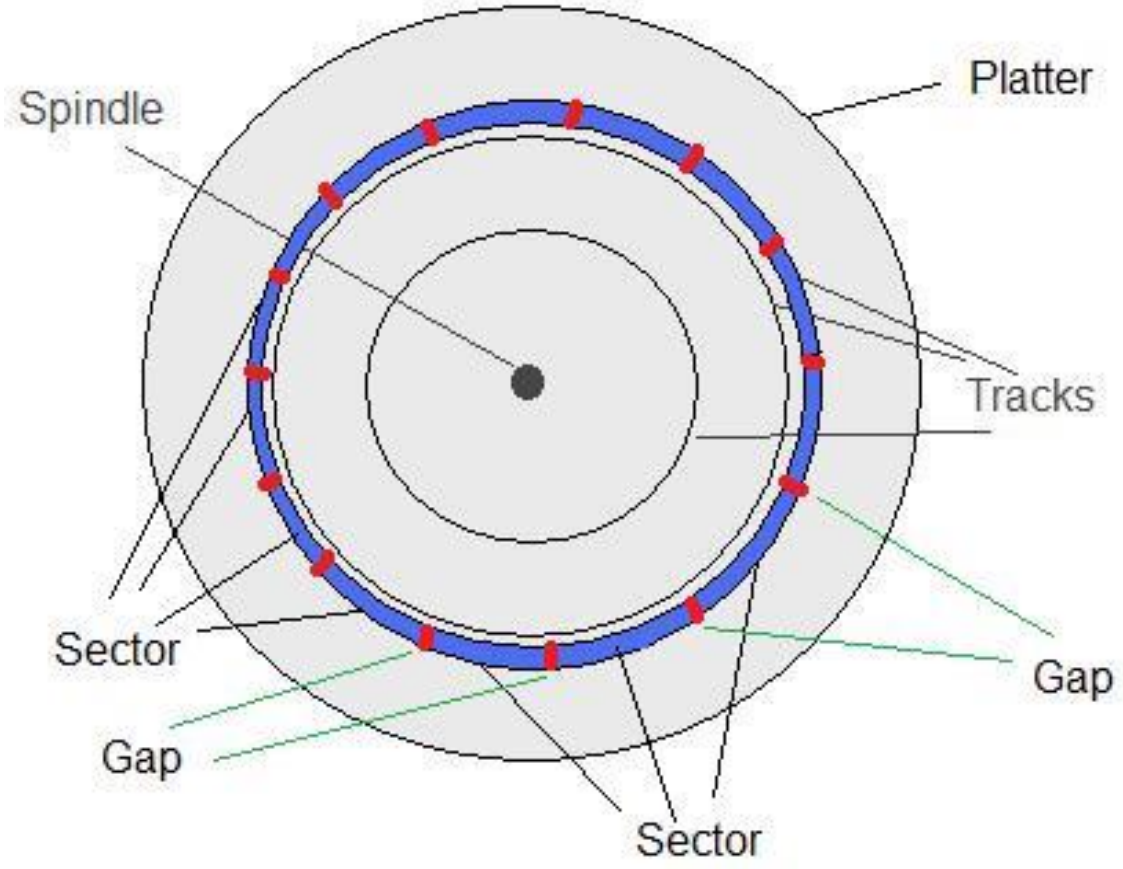
# Sectors

- Each track is divided into smaller parts called sectors



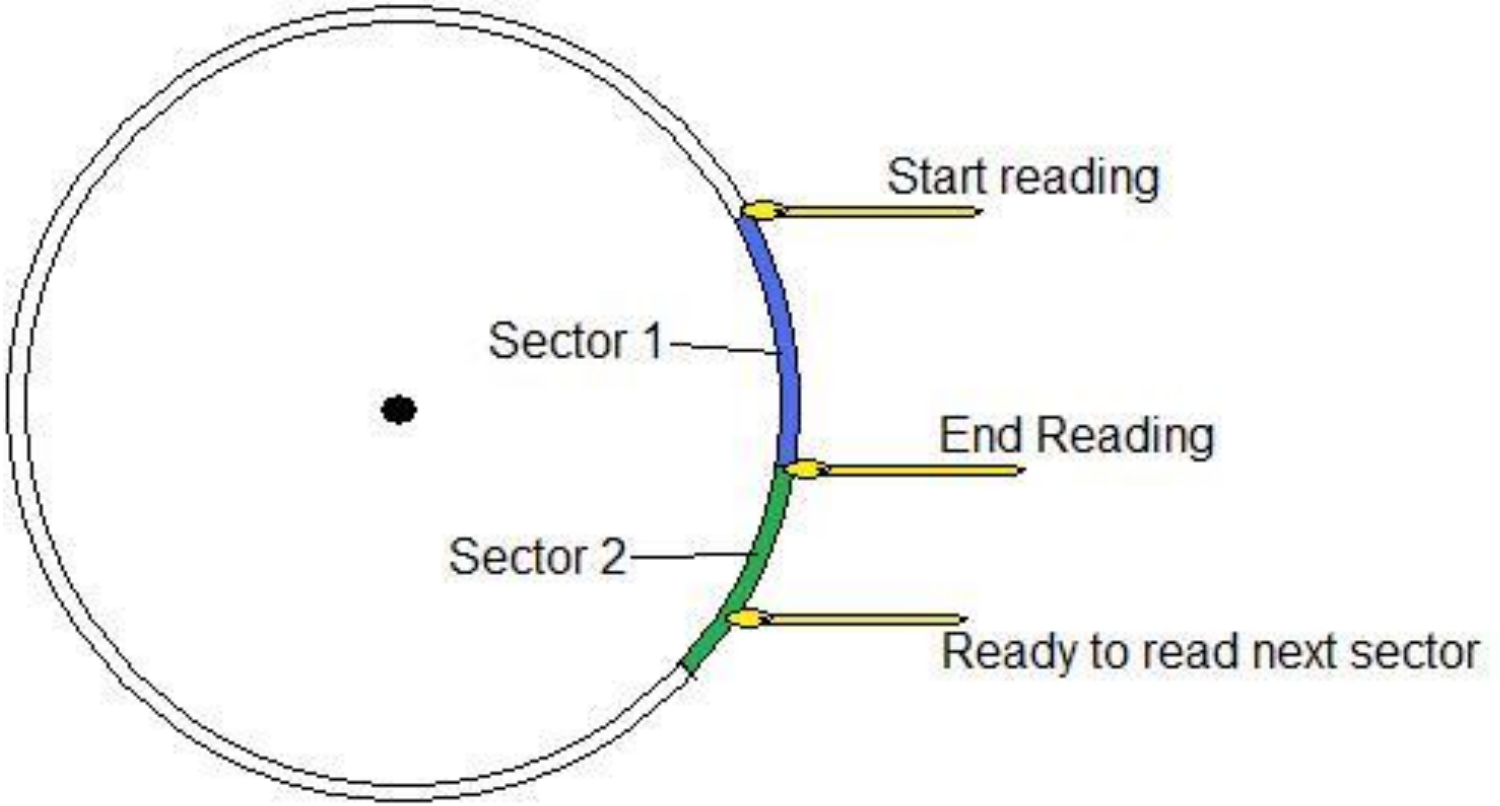
# Inter-Sector Gaps

- Gaps include information like: sector start marker, Sector number, track number, etc



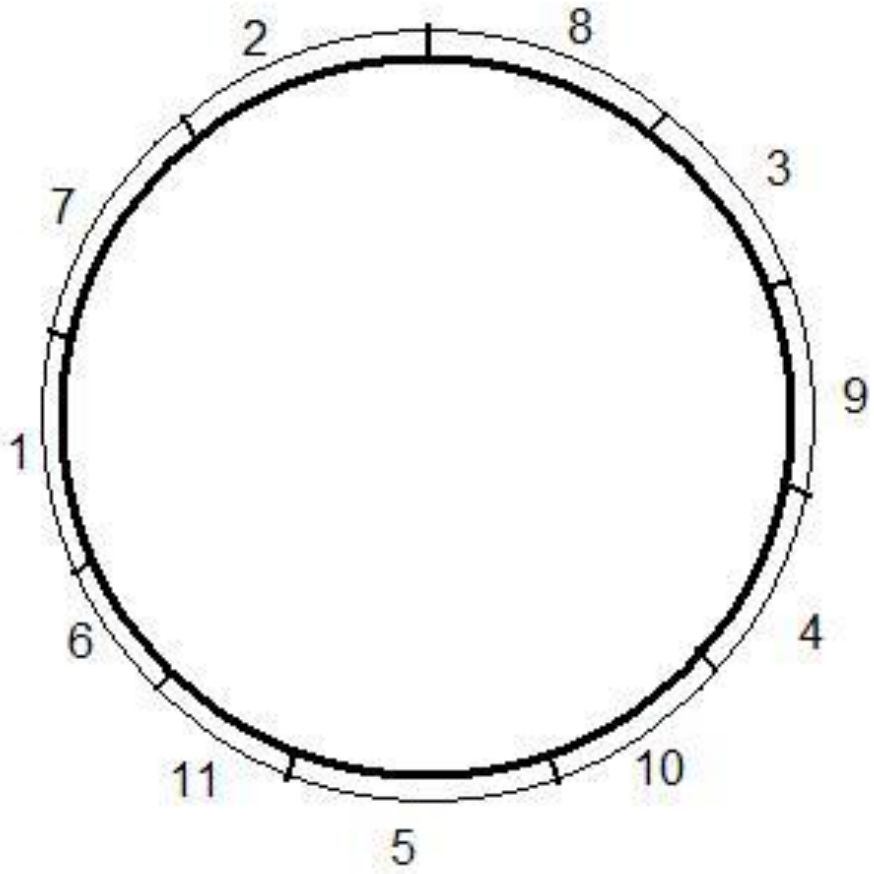
# Interleaving

- Error checking the data in the sector will slow down the I/O operation



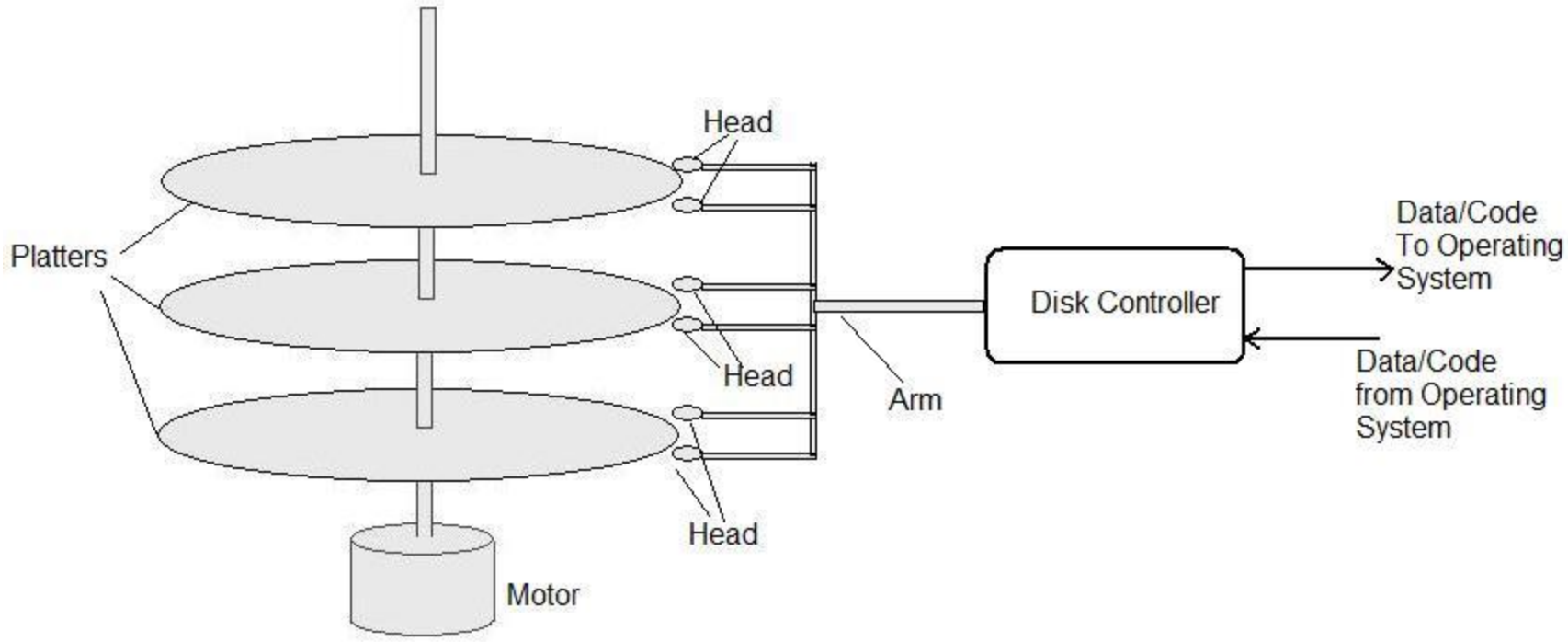
# Interleaving

- Changing the order of numbering the sectors can speed up file I/O



# Disk Controller

- Disk controller gets data/command from OS, controls I/O operation, and sends back the results to the OS



# Disk I/O Steps

- To read or write:
  - Move the head to the track
  - Find the sector
  - Transfer data to/from disk controller
  - Error checking and reporting to the OS

# Disk I/O Timings

- ***Seek Time (S):*** The time needed for the head to move onto the track
- ***Rotational Latency Time (r):*** The time needed for the disk to rotate until the sector comes under the head
- ***Block Transfer Time (btt):*** The time needed to transfer data from head to sector (write) or sector to head (read)

# Disk I/O Timing

Time to read/write a block : s+r+btt

Note: if the time needed for the head to pass over the gap is also considered then we have:

Time to read a block : s+r+ebt

**ebt** : Effective Block Transfer Time

# Optimizing File I/O

- In many data processing applications, the data file is read from the beginning to the end.
- For these applications if the data is stored on
  - the same track
  - Or the neighboring tracks

  The seek time (s) will be smaller, and the file I/O will be faster.

# Blocks and Records

- A **Block** is the unit of I/O from a hard disk
- It is not possible to read a fraction of a block from a hard disk
- A **record** is the unit of information stored in a file. Example: Student Record (St. ID, St. Name, St. major, St. address, …)
- A **File** is a set of related records. Example: Hospital data file

# Example 1

- Compute the time needed to read 10 consecutive blocks from the same track. Assume no interleaving.
- Use:

  s=16 msec

  r=8.3msec

  btt=0.8 msec

  ebt=0.84 msec

# Example 2

- Find the time needed to read 10 random blocks from the disk

  Use parameters from example 1

# Buckets

- If the record and the block have different sizes, then several records are stored in a block.
- A Bucket is a group of records stored in a block
- The file read/write unit is bucket (not record!!)
- The data read from a file is put in a temporary place called a ***Buffer***

# Blocking factor

- Blocking factor (Bfr) is the number of records in a block
- Example:

  Block size (B) = 2400 Bytes

  Record size ( R) = 100 Bytes

  Bfr = B/R = 2400 / 100 = 24

# Double Buffering

- Buffer is a place to store blocks for processing
- When a block is processed, the disk reads the second block and puts it in a second buffer
- The role of the first and the second buffer is changed for the third block
- This use of two buffers is called double buffering

# Questions?