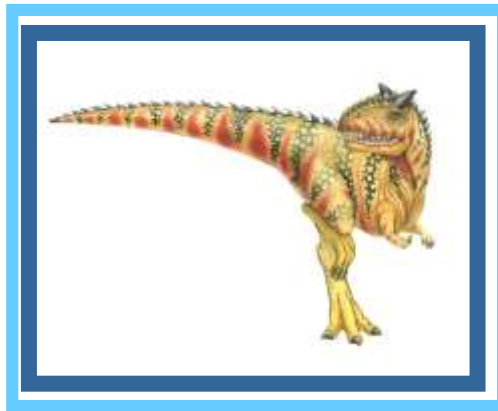


# I/O Management

---





# Chapter 12: Mass-Storage Systems

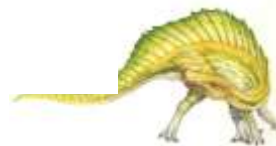
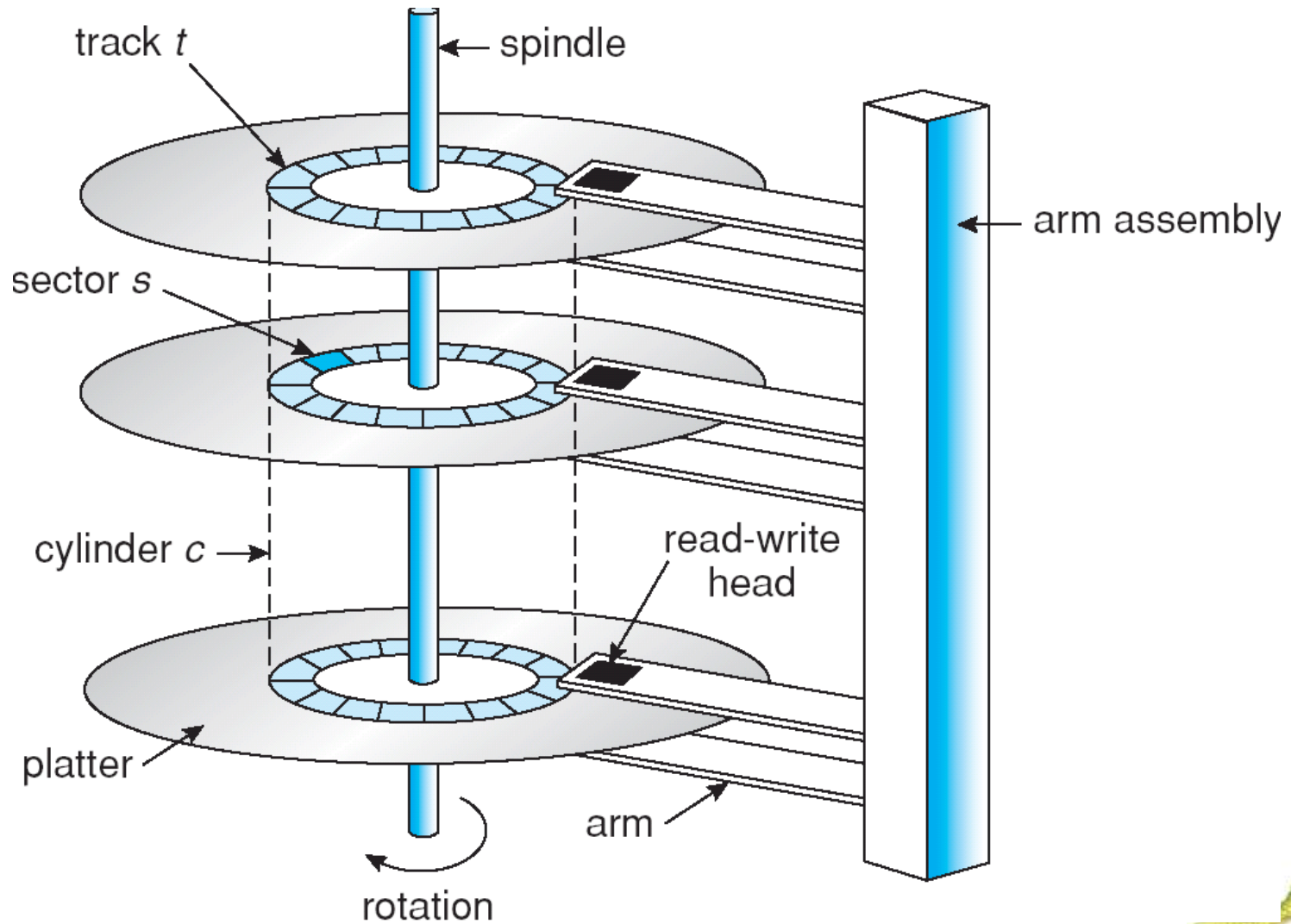
---

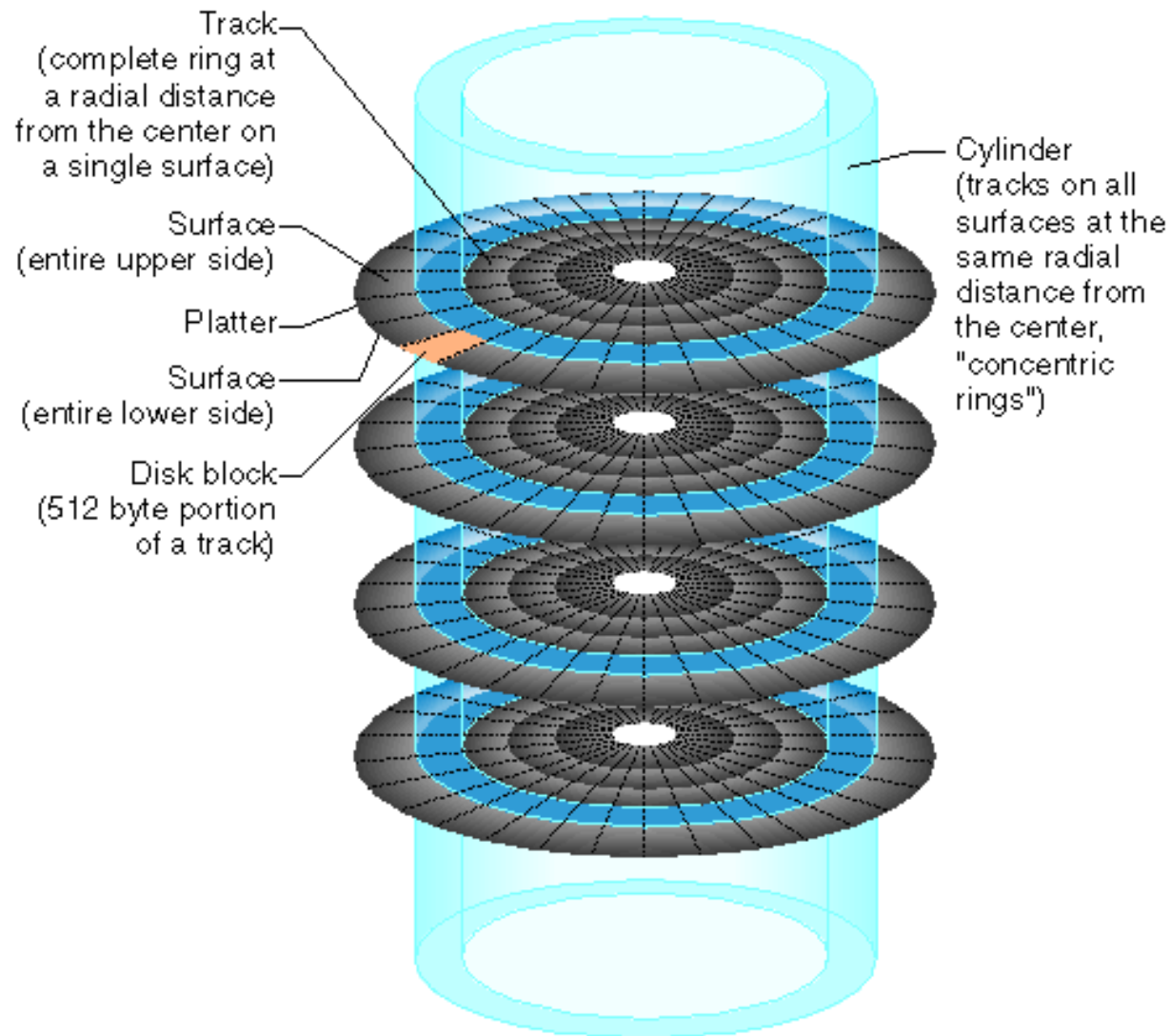
- Overview of Mass Storage Structure
- Disk Structure
- Disk Scheduling
- RAID Structure





# Moving-head Disk Mechanism







# Disk Structure

---

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
- 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





# Disk Scheduling

---

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Access time has two major components
  - **Seek time** is the time for the disk arm to move the heads to the cylinder containing the desired sector
  - **Rotational latency** is the additional time waiting for the disk to rotate the desired sector to the disk head
- Minimize seek time
- Seek time  $\approx$  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





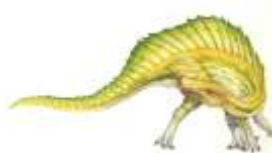
# Disk Scheduling (Cont)

---

- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate them with a request queue (0-199)

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

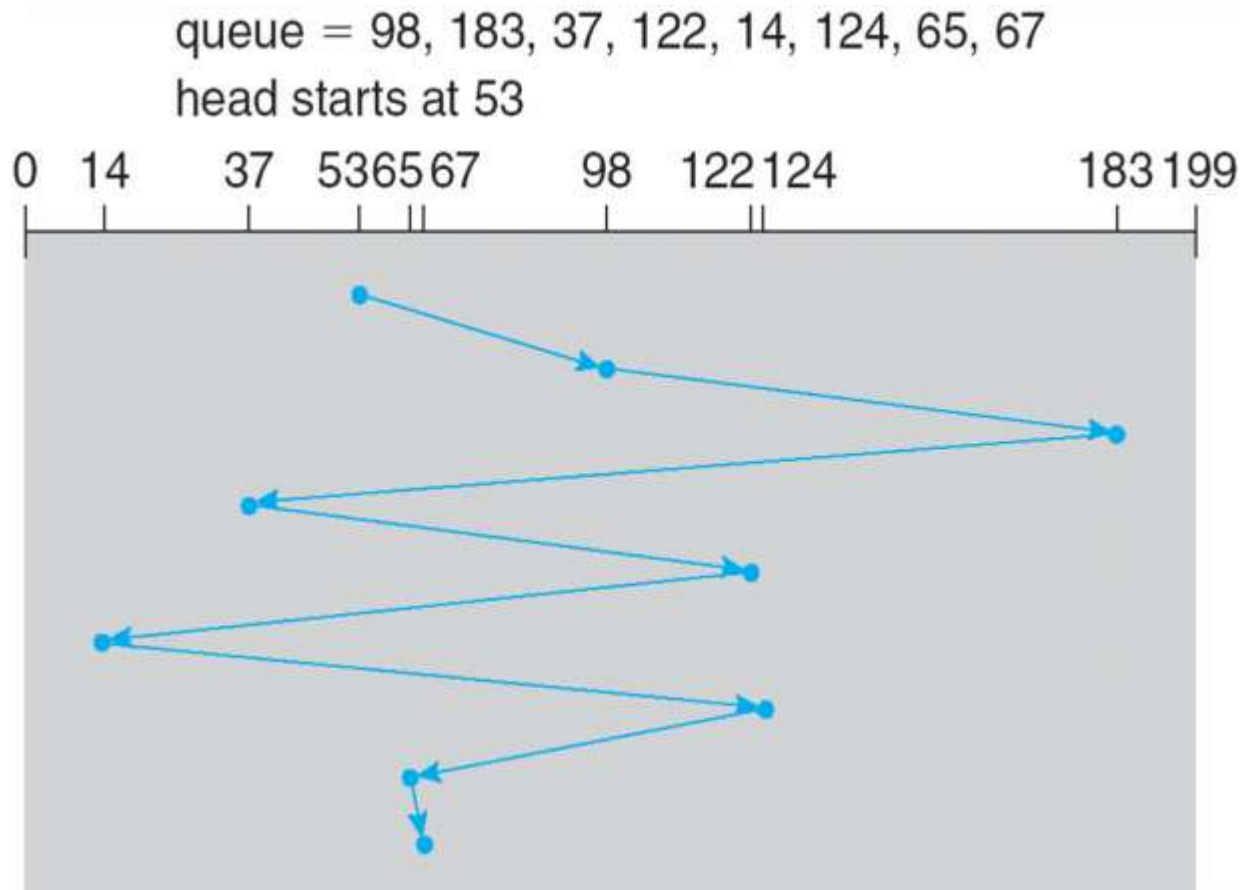
Head pointer 53





# FCFS

Illustration shows total head movement of 636 cylinders







# SSTF

---

- Selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders

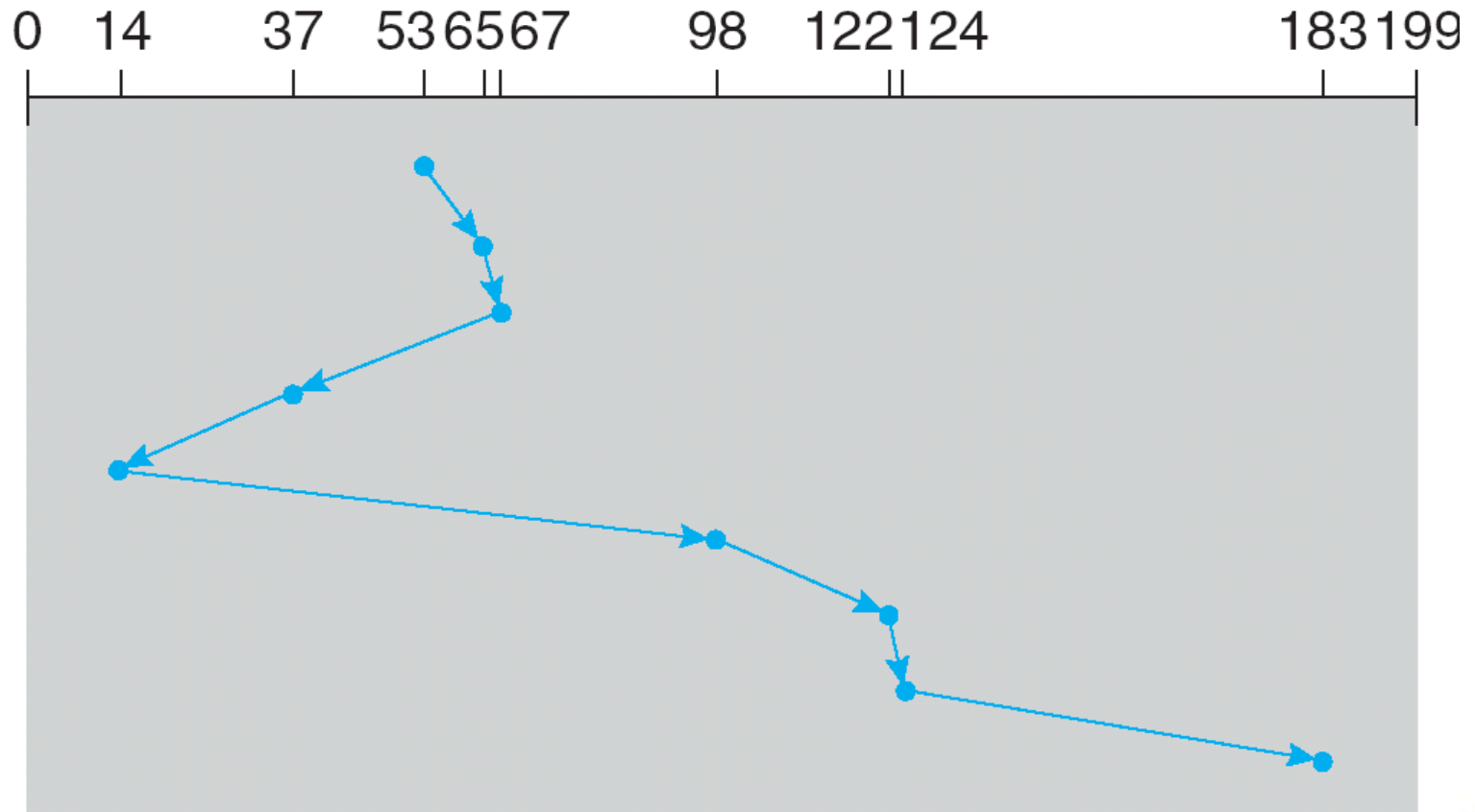




# SSTF (Cont)

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

head starts at 53





# 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 236 cylinders

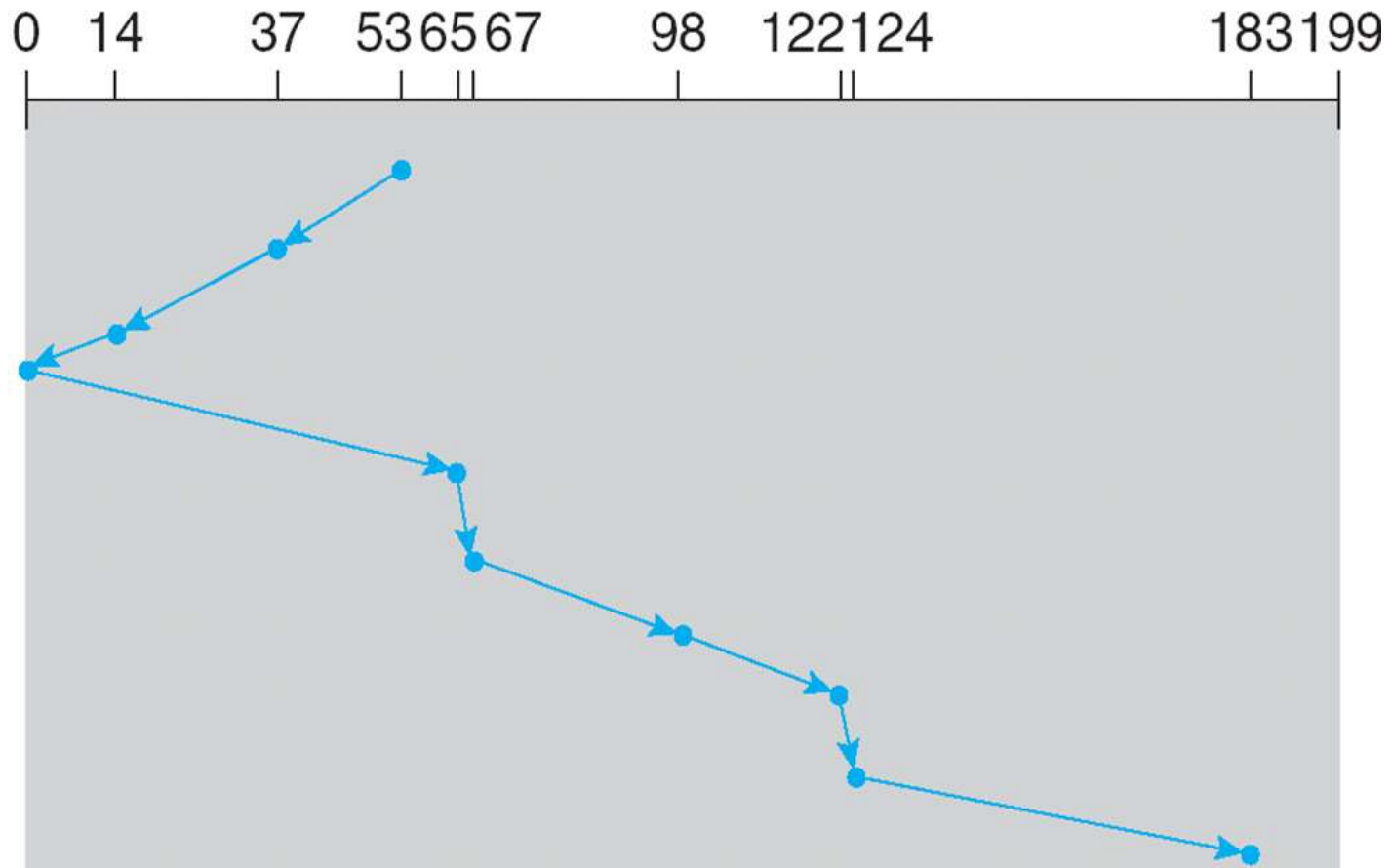




# SCAN (Cont.)

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

head starts at 53





# 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

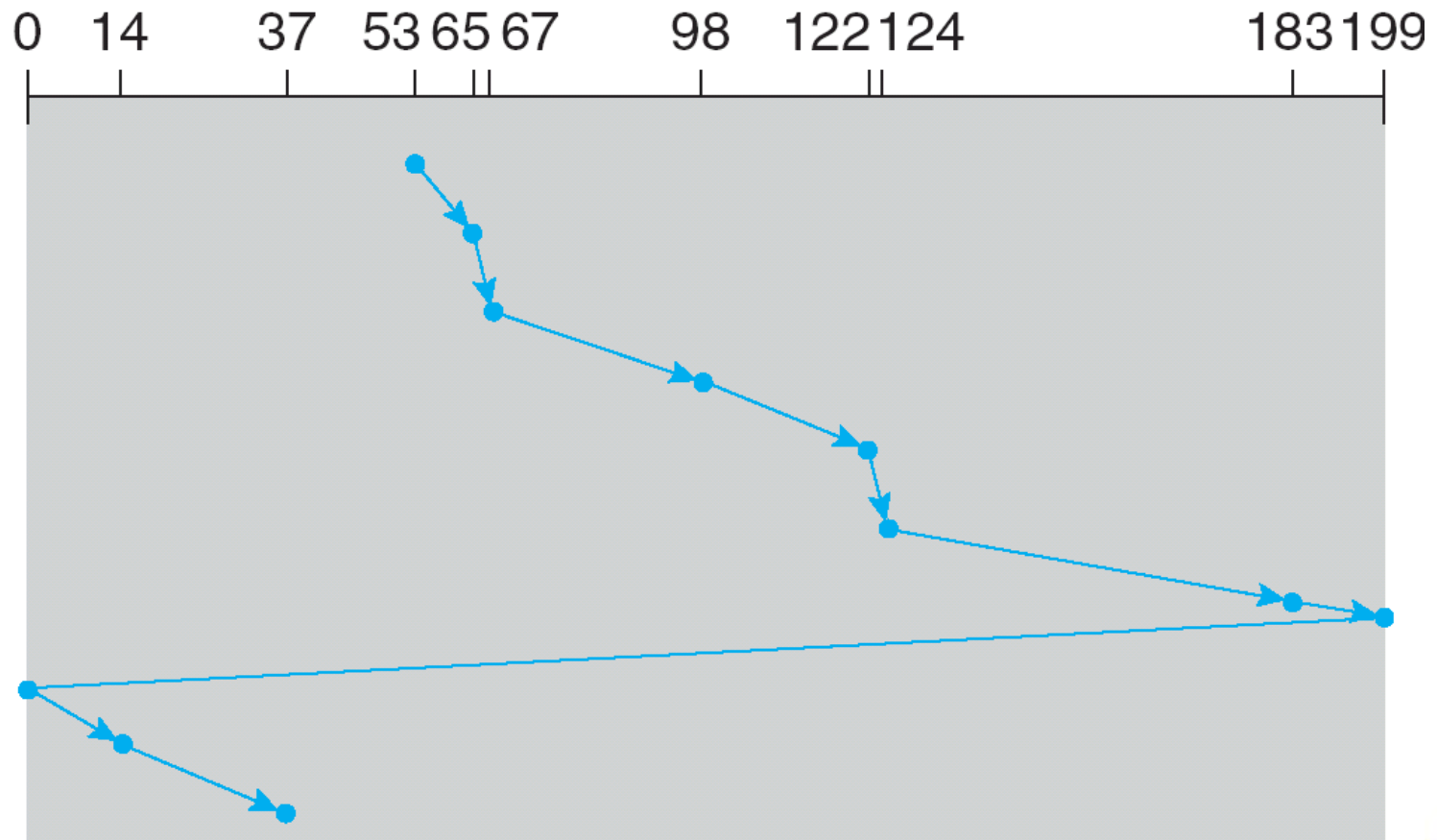




# C-SCAN (Cont)

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

head starts at 53





# C-LOOK

---

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk

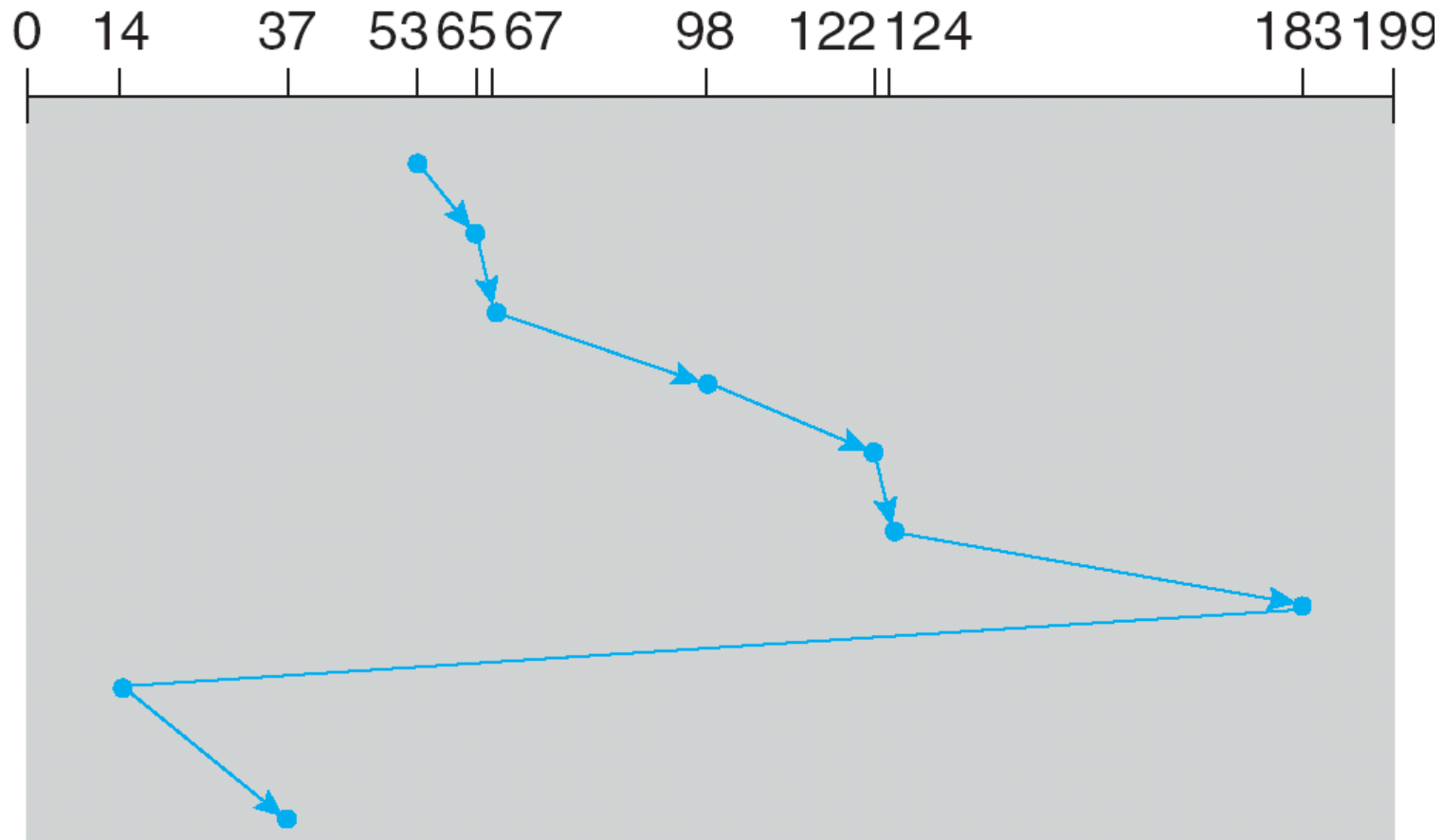




# C-LOOK (Cont)

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

head starts at 53







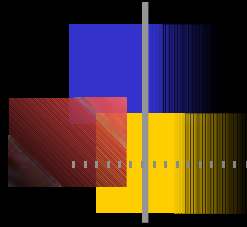
# Selecting a Disk-Scheduling Algorithm

---

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm

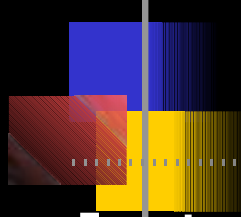


# What is RAID?



- RAID – Redundant Array of Independent Disks

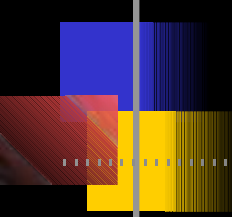
# Motivation for RAID



- Just as additional memory in form of cache, can improve system performance, in the same way **additional disks** can also improve system performance
- In RAID, we use an **array** of disks. These disks operate **independently**
- Since there are many disks, multiple I/O requests can be handled in parallel if the data required is on separate disks



# Benefits of RAID

- 
- Data loss can be very dangerous for an organisation
  - RAID technology prevents data loss due to disk failure
  - Servers make use of RAID technology
  -

# RAID Technology

There are 7 levels RAID schemes. These are called RAID 0, RAID 1, ... RAID 6

■ RAID storage techniques

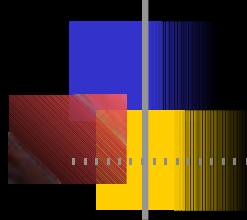
The main methods of storing data in the array are:

□ Striping - splitting the flow of data into blocks of a certain size (called "block size") then writing of these blocks across the RAID one by one. This way of data storage affects on the performance.

□ Mirroring is a storage technique in which the identical copies of data are stored on the RAID members simultaneously. This type of data placement affects the fault tolerance as well as the performance.

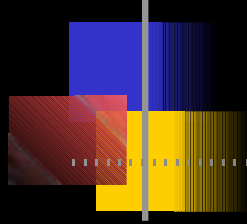
□ Parity is a storage technique which is utilized striping and checksum methods. In parity technique, a certain parity function is calculated for the data blocks. If a drive fails, the missing block are recalculated from the checksum, providing the RAID fault tolerance.

# RAID Level 0 - Characteristics



- RAID level 0 divides data into block units and writes them across a number of disks.
- As data is placed across multiple disks, it is also called “data stripping”
- The advantage of distributing data over disks is that if two different I/O requests are pending for two different blocks of data, then there is a possibility that the requested blocks are on different disks

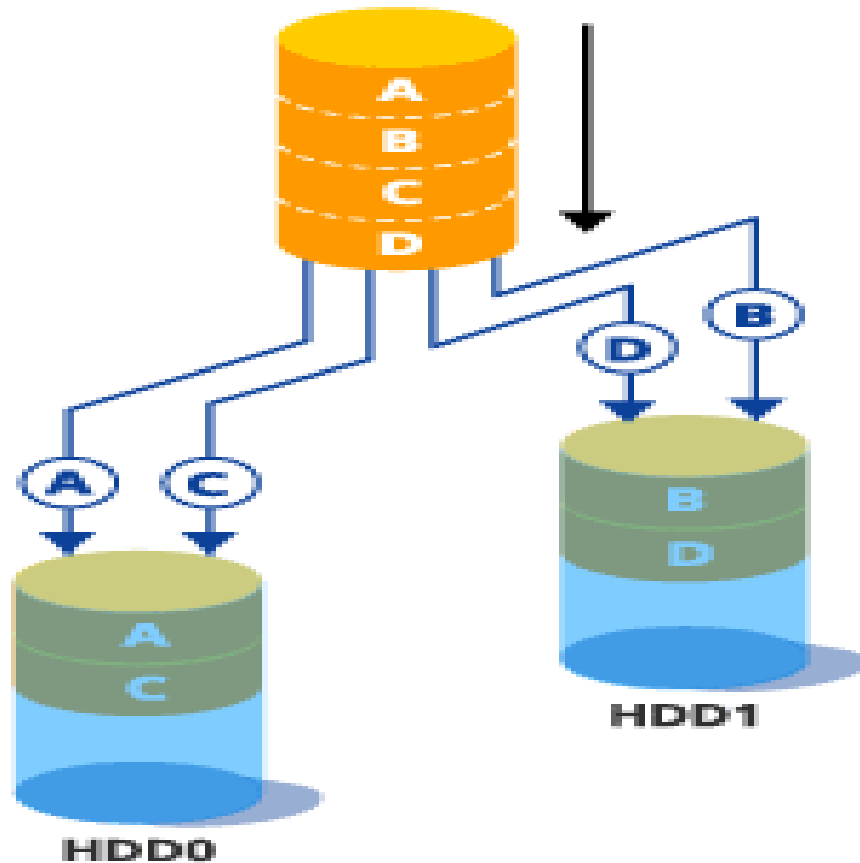
# RAID Level 0



- There is **no parity checking** of data.
- So if data in one drive gets corrupted then all the data would be lost. Thus RAID 0 does **not** support **data recovery**
- **Spanning** is another term that is used with RAID level 0 because the logical disk will span all the physical drives
- RAID 0 implementation requires **minimum 2 disks**

# RAID Level 0 - Diagram

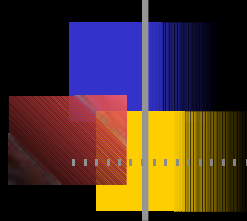
Write order from CPU for data "ABCD"



Data is written by spreading it across multiple disks, this is called "striping"

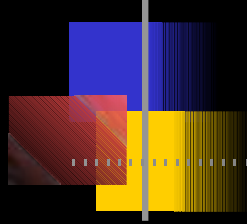


# RAID Level 0 - Advantages



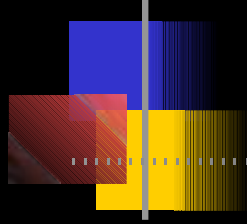
- Advantage of RAID level 0 is that it increases speed.
- Throughput (speed) is increased because :
  - Multiple data requests probably not on same disk
  - Disks seek in parallel
  - A set of data is likely to be striped across multiple disks
- Implementation is easy
- No overhead of parity calculation

# RAID Level 0 - Disadvantages



- Not a true RAID because it is not fault tolerant
- The failure of just one drive will result in all data in an array being lost. Implementation is easy
- Should not be used in mission critical environments

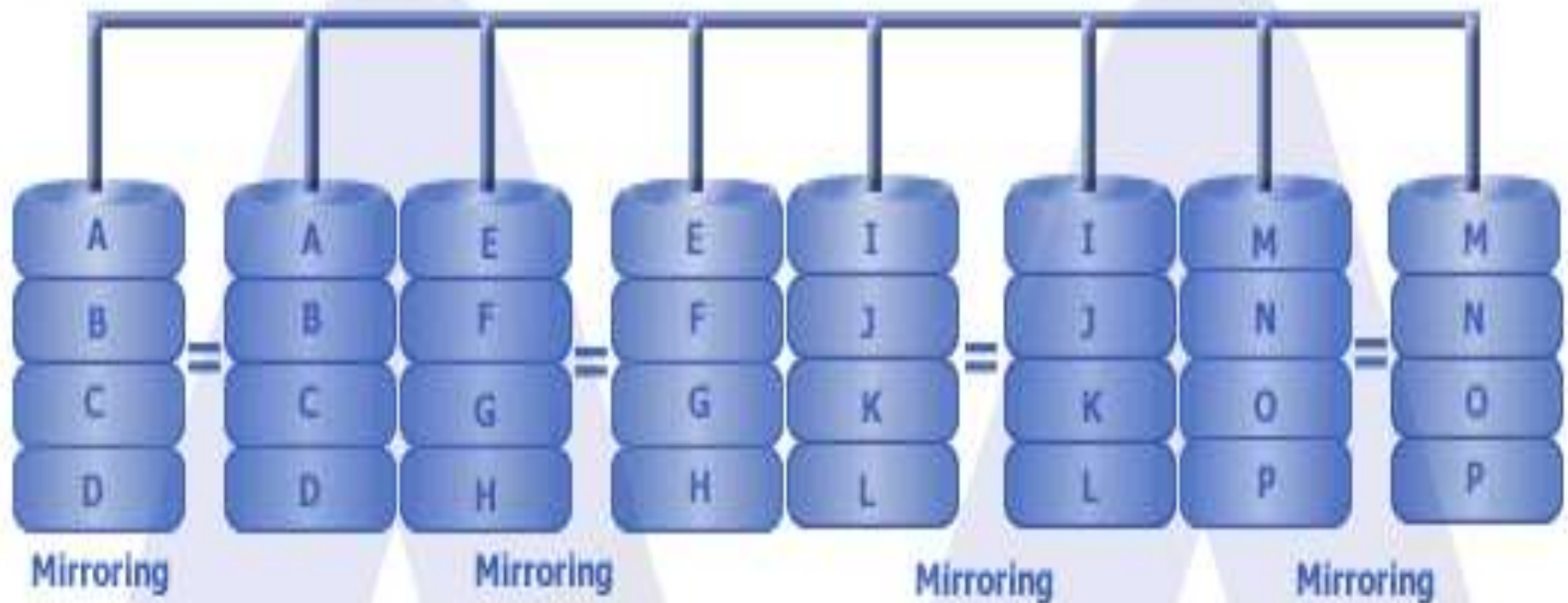
# RAID Level 1 - Characteristics



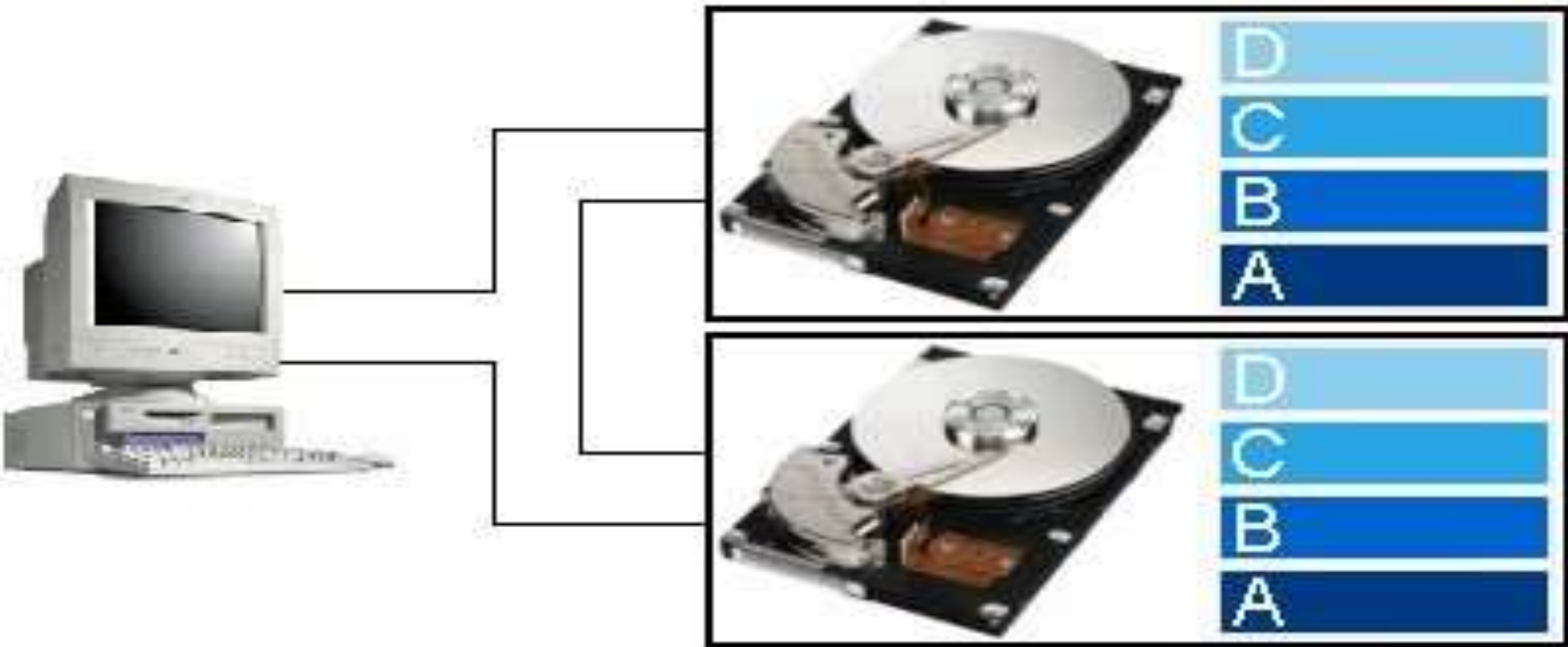
- This level is called "**mirroring**" as it copies data onto two disk drives simultaneously.
- As same data is placed on multiple disks, it is also called "**data mirroring**"
- The automatic duplication of the data means there is little likelihood of data loss or system downtime.

# RAID Level 1 - Diagram

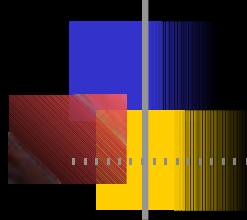
## Raid Level 1: Mirroring & Duplexing



# RAID Level 1 - Animation

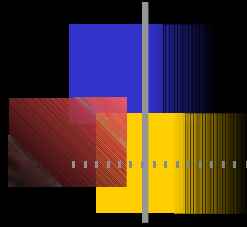


# RAID Level 1 - Characteristics



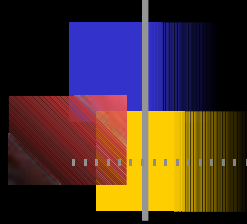
- A read request can be executed by either of the two disks
- A write request means that both the disks must be updated. This can be done in parallel
- There is no overhead of storing parity information
- Recovery from failure is simple. If one drive fails we just have to access data from the second drive

# RAID Level 1 - Advantages



- Main advantage is RAID 1 provides fault tolerance. If one disk fails, the other automatically takes over.
- So continuous operation is maintained.
- RAID 1 is used to store systems software (such as drivers, operating systems, compilers, etc) and other highly critical files.

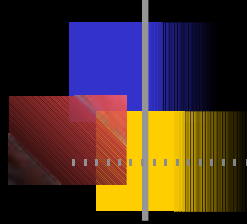
# RAID Level 1 - Disadvantages



- Main disadvantage is cost. Since data is duplicated, storage costs increase.



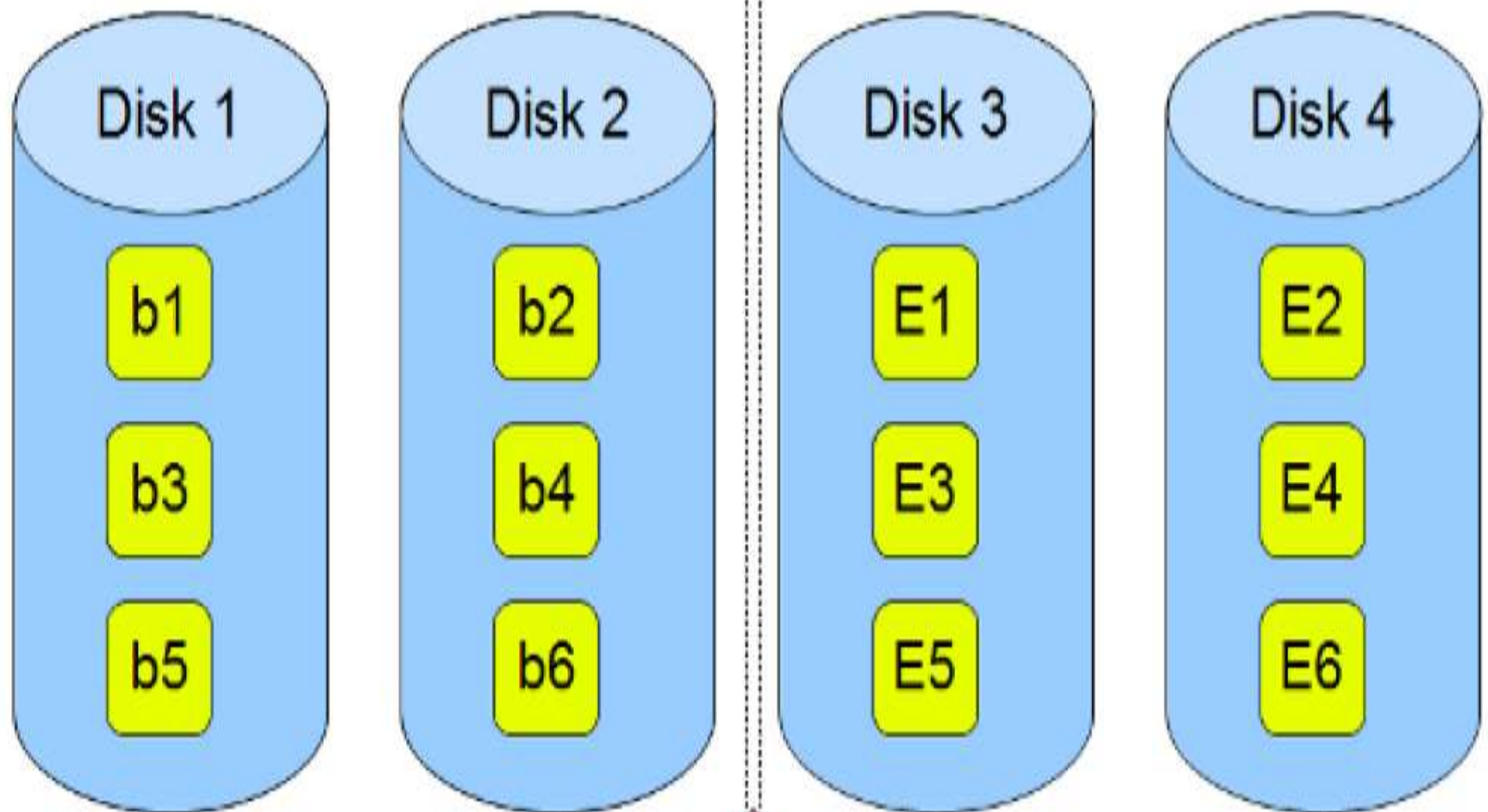
# RAID Level 2



- In RAID 2 mechanism, all disks participate in the execution of every I/O request.
- The spindles of individual disk drives are synchronized so that each disk head is in the same position on each disk at any given time.
- Data stripping is used.
- Error correcting code is also calculated and stored with data
- Not implemented in practice due to high costs and overheads

Data Disks

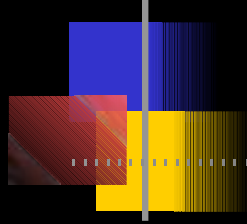
ECC Disks



**RAID 2** – Bits Striped. ( and stores ECC)

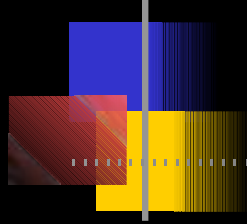
- This uses bit level striping. i.e Instead of striping the blocks across the disks, it stripes the bits across the disks.
- In the above diagram b1, b2, b3 are bits. E1, E2, E3 are error correction codes.
- You need two groups of disks. One group of disks are used to write the data, another group is used to write the error correction codes.
- This uses Hamming error correction code (ECC), and stores this information in the redundancy disks.
- When data is written to the disks, it calculates the ECC code for the data on the fly, and stripes the data bits to the data-disks, and writes the ECC code to the redundancy disks.
- When data is read from the disks, it also reads the corresponding ECC code from the redundancy disks, and checks whether the data is consistent. If required, it makes appropriate corrections on the fly.

# RAID Level 3



- Data is divided into byte units and written across multiple disk drives.
- Parity information is stored for each disk section and written to a dedicated parity drive.
- All disks can be accessed in parallel
- Data can be transferred in bulk. Thus high speed data transmission is possible

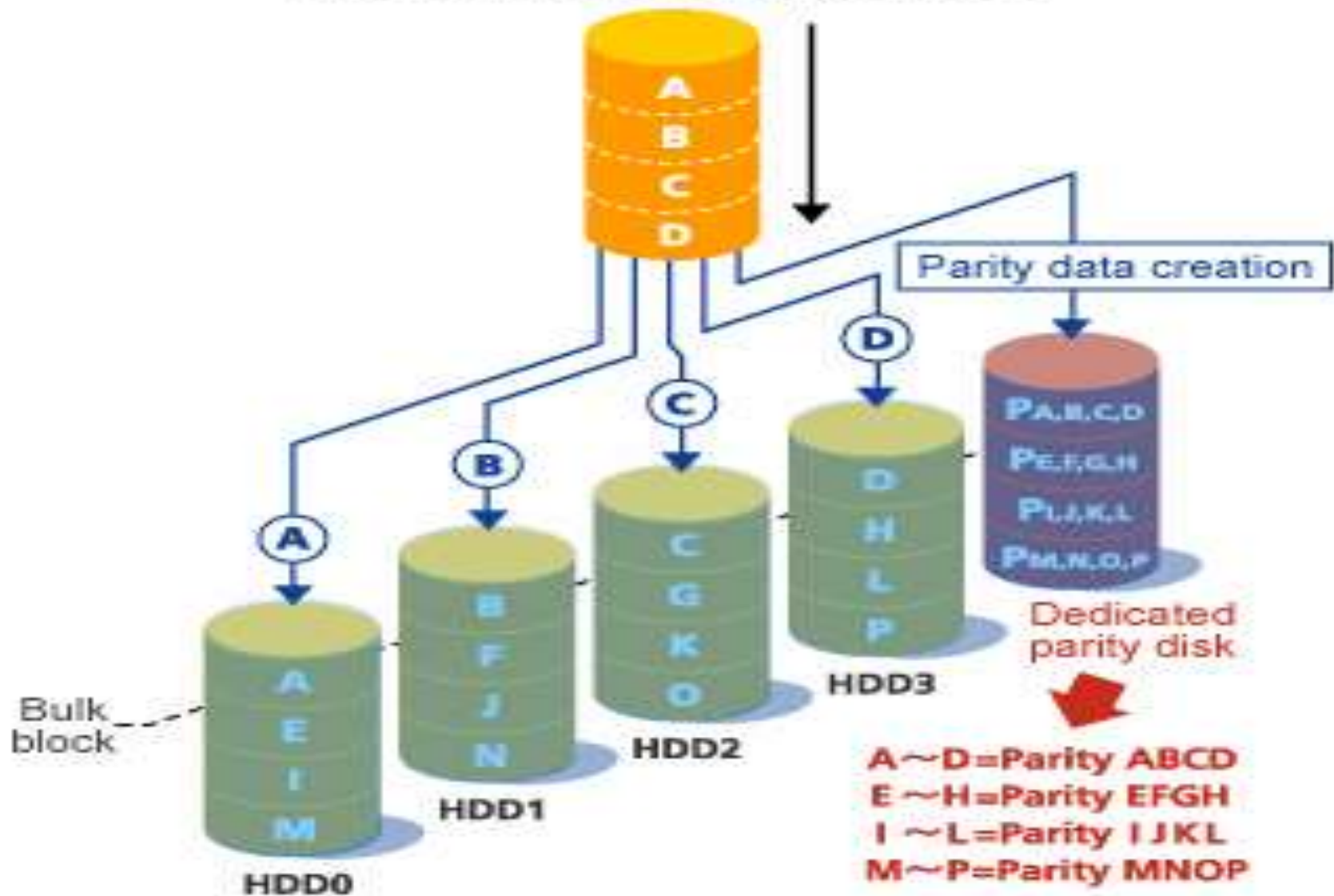
# RAID Level 3



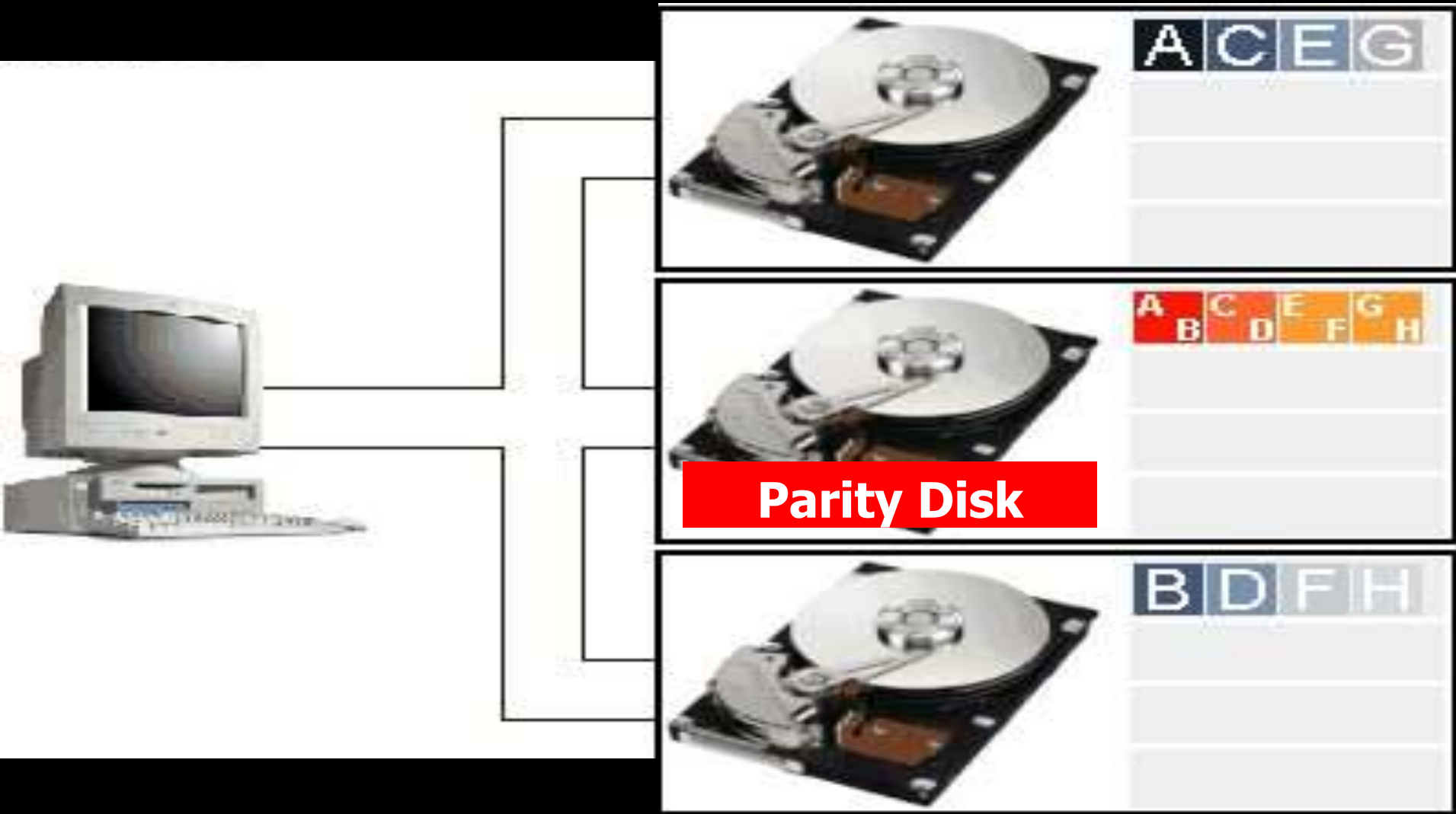
- In case of drive failure, the parity drive is accessed and data is reconstructed from the remaining devices.
- Once the failed drive is replaced, the missing data can be restored on the new drive
- RAID 3 can provide very high data transfer rates

# RAID Level 3

Write order from CPU for data "ABCD"

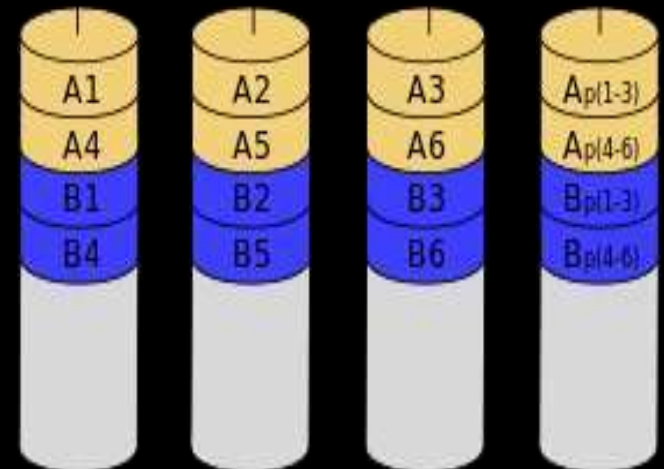
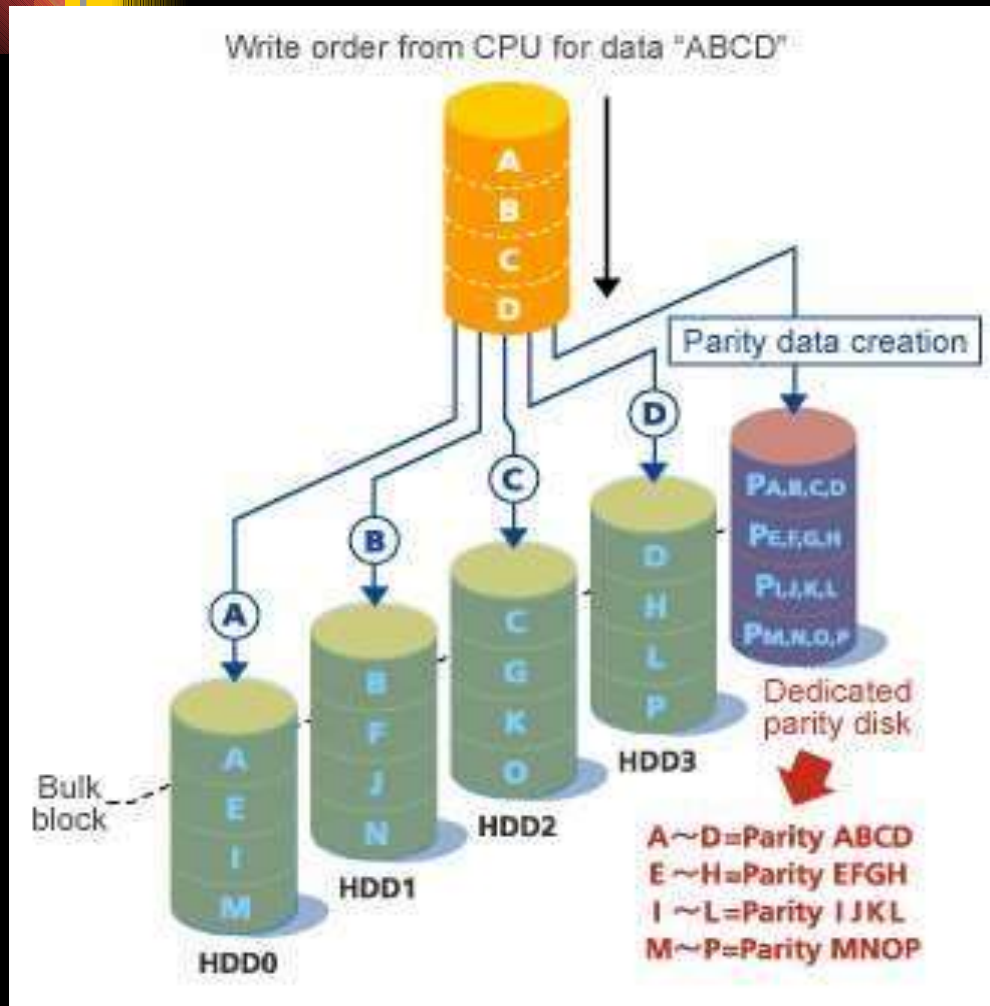


# RAID Level 3



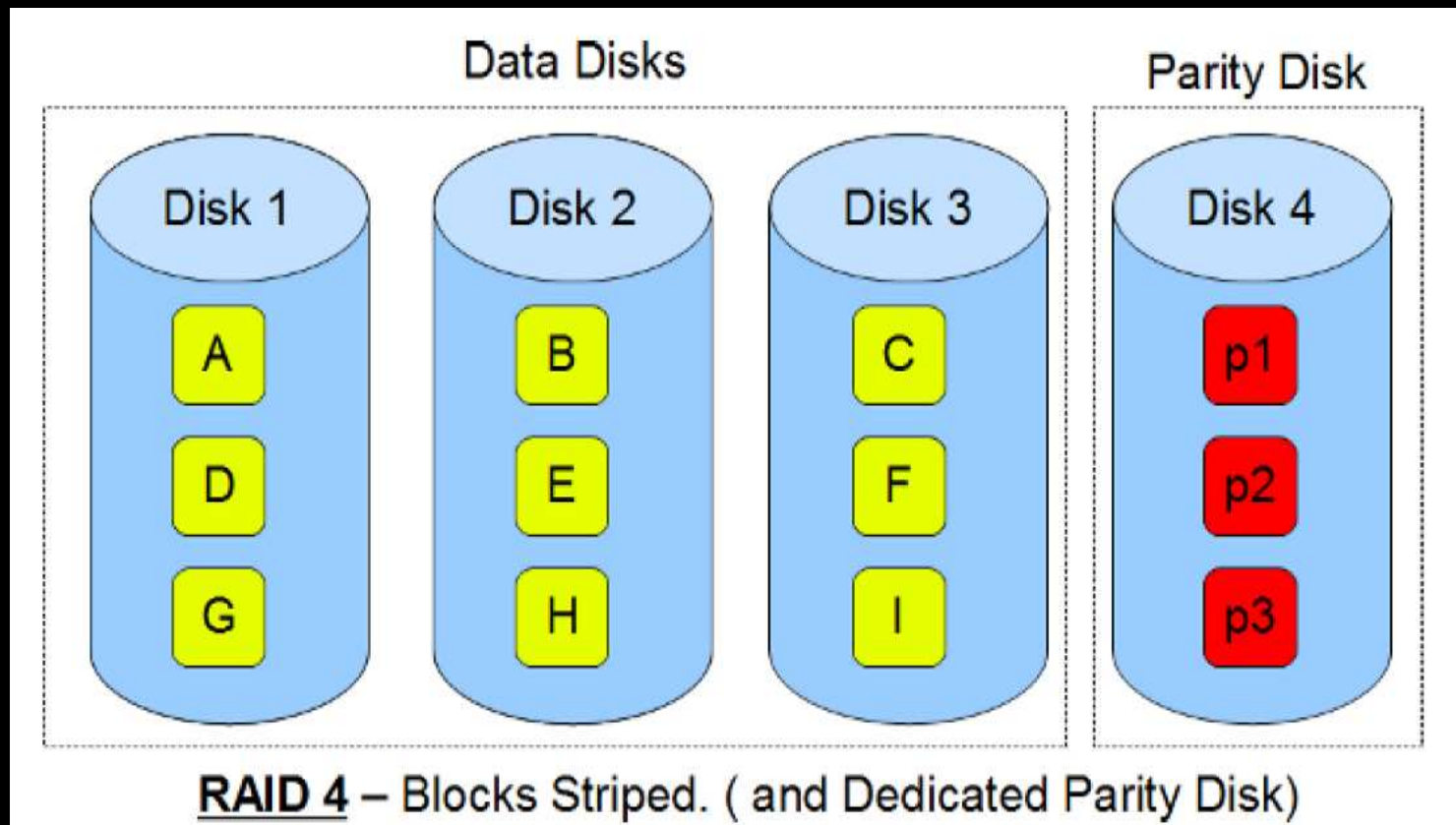


# RAID

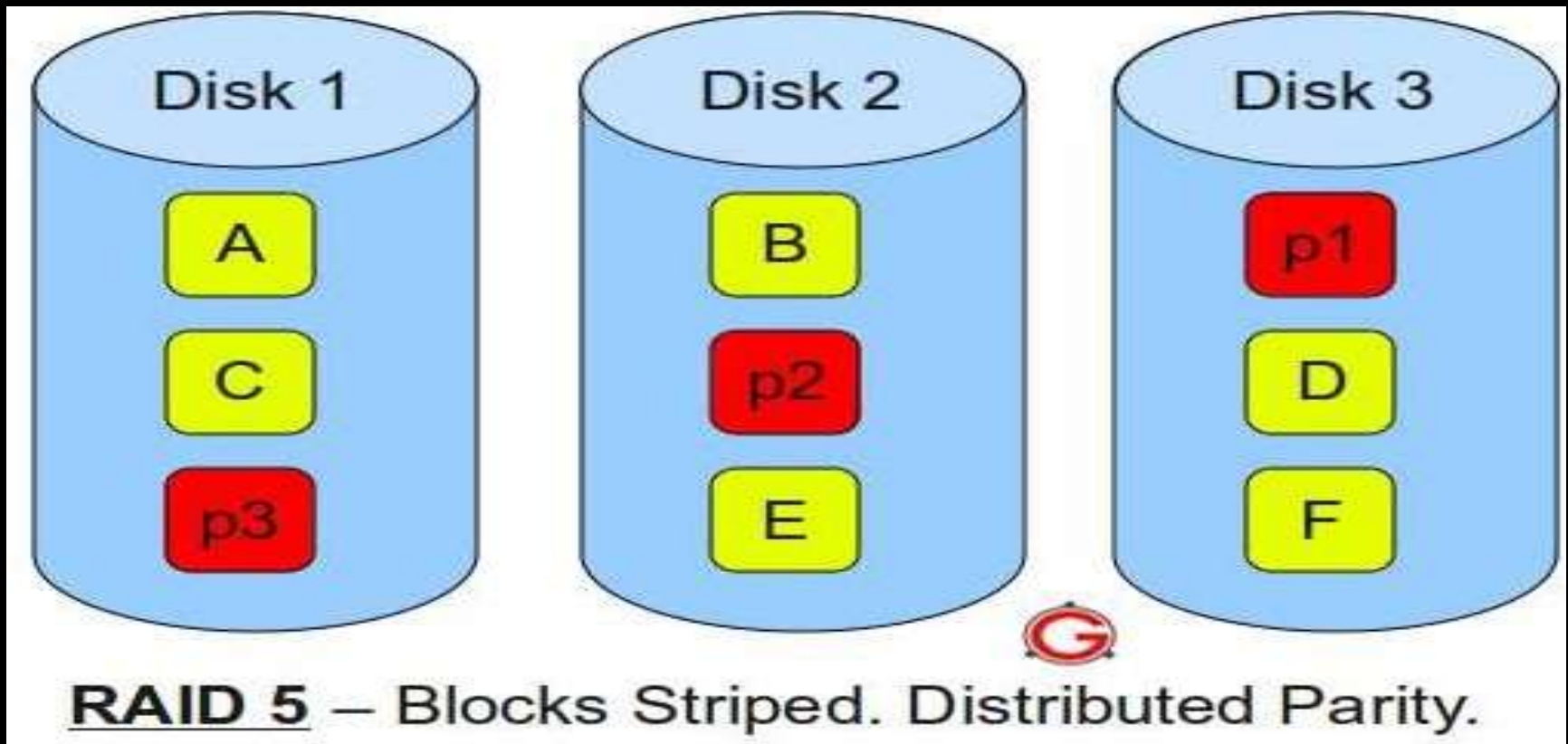




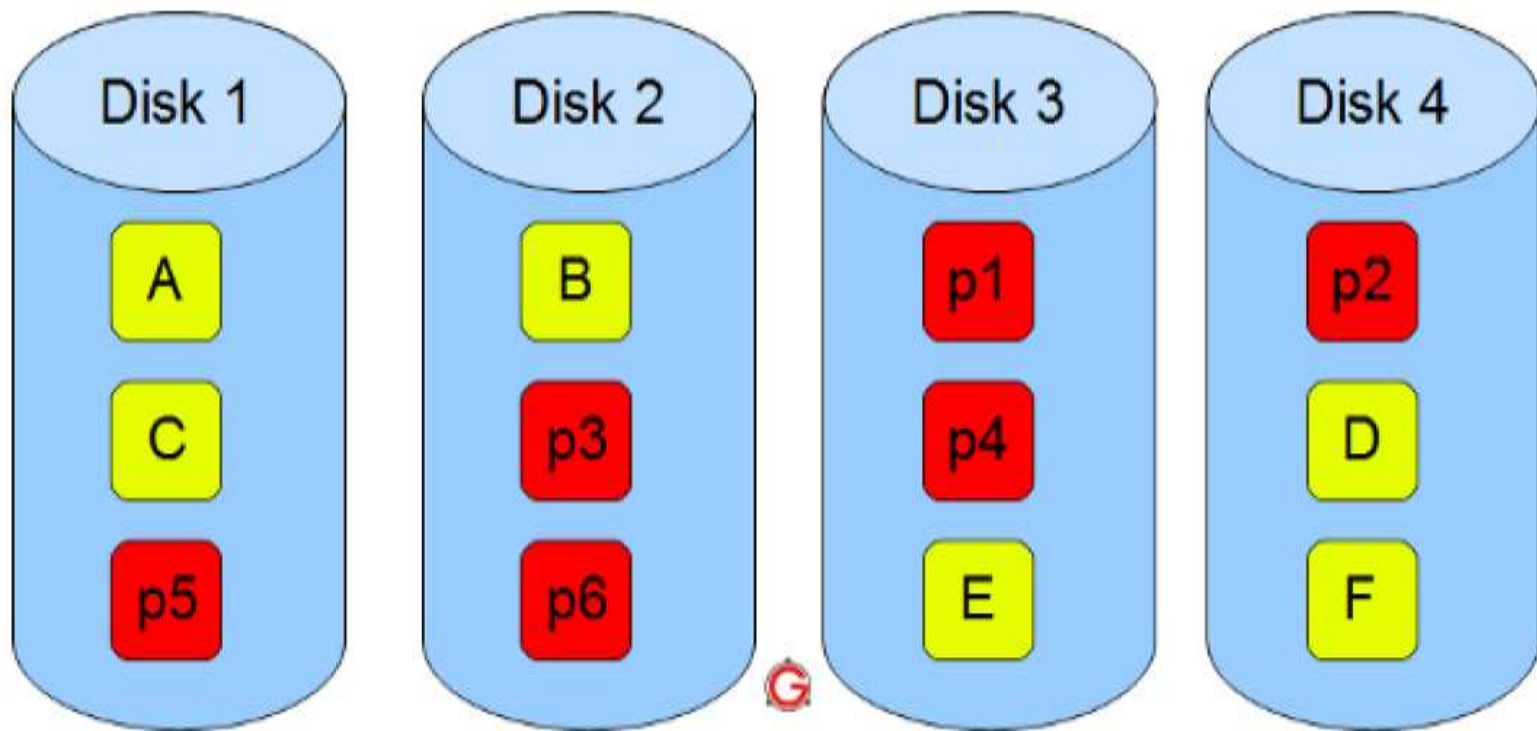
# RAID LEVEL 4



## RAID LEVEL 5: Block interleaved parity



# RAID 6

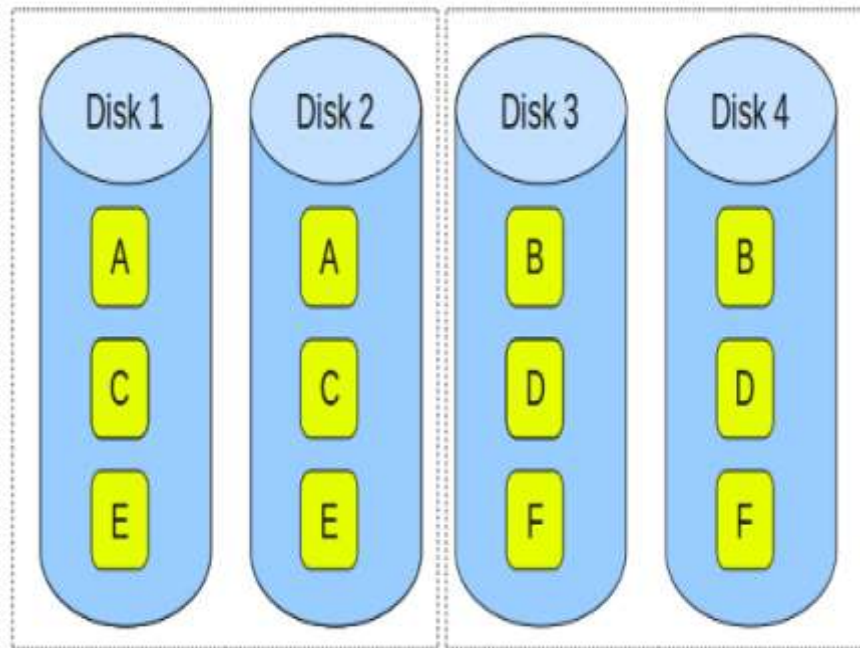


**RAID 6** – Blocks Striped. Two Distributed Parity.

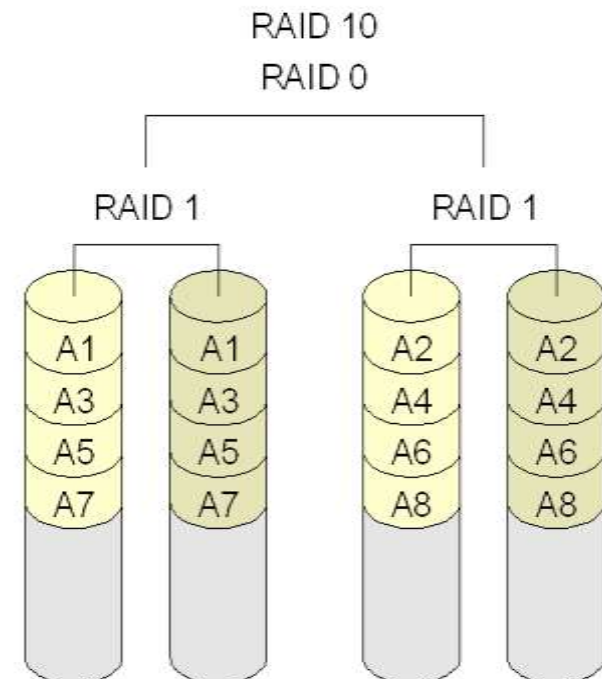
# RAID 10

- Uses multiple (mirrored) RAID 1 in a single array
- Data striped across all mirrored sets
- Very high fault tolerance
- High performance rate

# RAID 10:stripe of mirrors

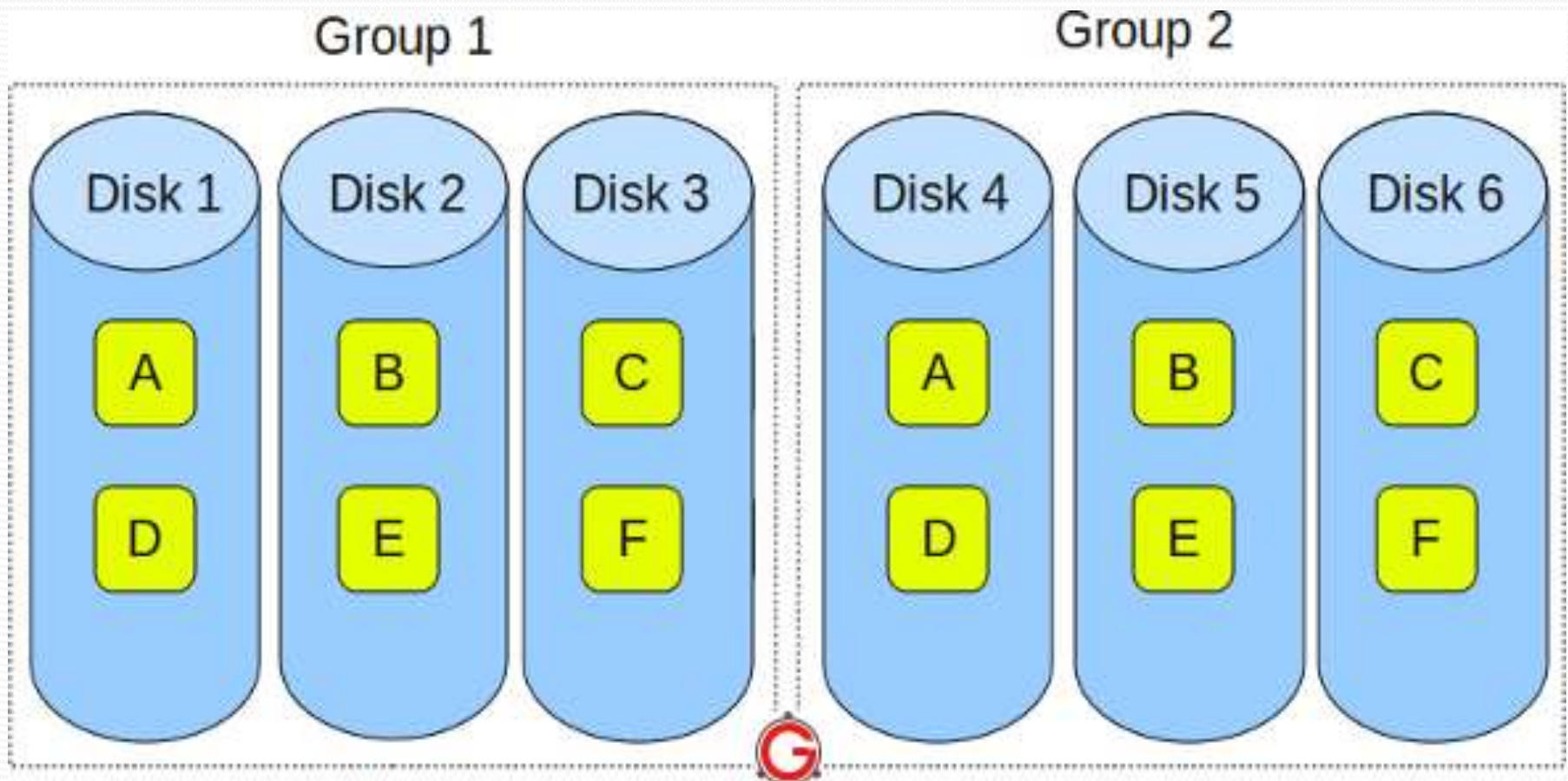


**RAID 10** – Blocks Mirrored. ( and Blocks Striped)





# RAID 0+1: Mirrors of stripes



**RAID 01** – Blocks Striped. ( and Blocks Mirrored)

# How are the HDD are designed in RAID technologies





# Advantages of RAID

- The foremost advantage of using a RAID drive is that it increases the **performance and reliability** of the system.
- The RAID drive is a credible example that could be used in a server.
- The RAID increases the **parity check** and thus it regularly checks for any possibility of a system crash. Disk stripping is also a hot topic when we discuss about the RAID drives.
- The performance is much **highlighting** and increases a lot when the disk stripping is done.
- The mirroring is the complete duplication of the data. Or in the other sense the mirroring is the **100% duplication** of the data on two drives.



# Disadvantages of Raid

- A major disadvantage regarding the RAID drive is that there needs to be written the drivers for a Network Operating System (NOS).
- Hence the major fact and also the most important usage of the RAID system is that it is essentially designed and extensively used in a server.
- Another disadvantage regarding the RAID is that it is very much difficult for an administrator to configure the RAID system.
- The ability to dynamically enlarge the RAID server is also complex process; especially for those administrators who are the IS managers and also the LAN administrators.

# PROTECTION

## Goals of Protection

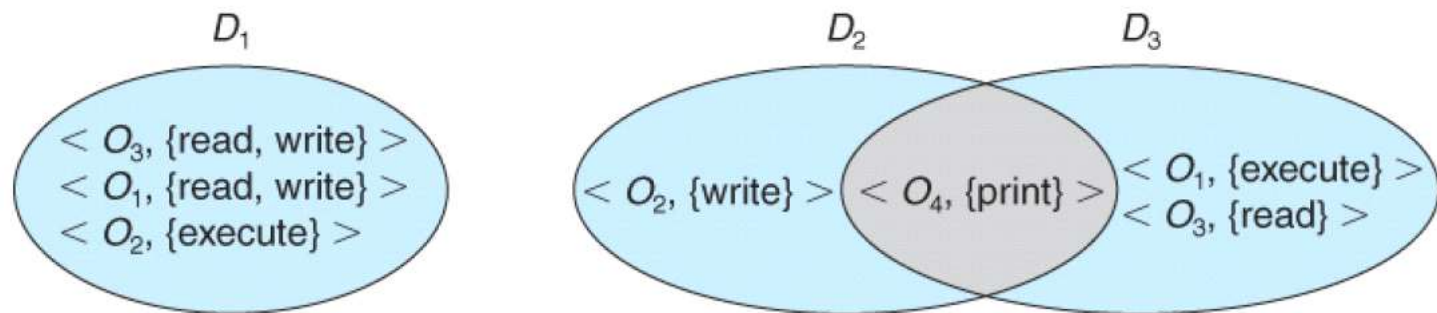
- Obviously to prevent malicious misuse of the system by users or programs.
- To ensure that each shared resource is used only in accordance with system policies, which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the mechanisms for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

## Principles of Protection

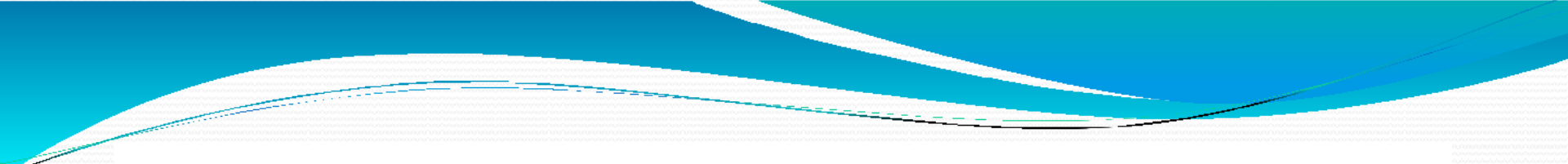
- The principle of least privilege dictates that programs, users, and systems be given just enough privileges to perform their tasks.
- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of "network" or "backup" or some other pseudo group, rather than SUID with root ownership. This limits the amount of damage that can occur if something goes wrong.
- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities - The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges.

## Domain Structure

- A protection domain specifies the resources that a process may access.
- Each domain defines a set of objects and the types of operations that may be invoked on each object.
- An access right is the ability to execute an operation on an object.
- A domain is defined as a set of  $\langle \text{object}, \{ \text{access right set} \} \rangle$  pairs, as shown below. Note that some domains may be disjoint while others overlap.



System with three protection domains.



The association between a process and a domain may be static or dynamic.

- If the association is static, then the objects as well as access rights are fixed throughout the complete execution.
- If the association is dynamic, then there will be chance of changing objects and access rights.
- Domains may be realized in different fashions - as users, or as processes, or as procedures.

## Access Matrix:

Our model of protection can be viewed abstractly as a matrix, called an access matrix. The rows of the access matrix represent domains, and the columns represent objects. Each entry in the matrix consists of a set of access rights.

| domain \ object |               |       |               |         |
|-----------------|---------------|-------|---------------|---------|
|                 | $F_1$         | $F_2$ | $F_3$         | printer |
| $D_1$           | read          |       | read          |         |
| $D_2$           |               |       |               | print   |
| $D_3$           |               | read  | execute       |         |
| $D_4$           | read<br>write |       | read<br>write |         |



Allowing controlled change to the contents of the access matrix entries requires three additional operations:

- switch
- copy
- owner
- control.

## Switch

| domain \ object |               |       |               |               |        |        |        |        |
|-----------------|---------------|-------|---------------|---------------|--------|--------|--------|--------|
|                 | $F_1$         | $F_2$ | $F_3$         | laser printer | $D_1$  | $D_2$  | $D_3$  | $D_4$  |
| $D_1$           | read          |       | read          |               |        | switch |        |        |
| $D_2$           |               |       |               | print         |        |        | switch | switch |
| $D_3$           |               | read  | execute       |               |        |        |        |        |
| $D_4$           | read<br>write |       | read<br>write |               | switch |        |        |        |

## Copy:

The ability to copy an access right from one domain (or row) of the access matrix to another is denoted by an asterisk (\*) appended to the access right. The copy right allows the copying of the access right only within the column (that is, for the object) for which the right is defined.

| <div>object</div> <div>domain</div> | $F_1$   | $F_2$ | $F_3$   |
|-------------------------------------|---------|-------|---------|
| $D_1$                               | execute |       | write*  |
| $D_2$                               | execute | read* | execute |
| $D_3$                               | execute |       |         |

(a)

| <div>object</div> <div>domain</div> | $F_1$   | $F_2$ | $F_3$   |
|-------------------------------------|---------|-------|---------|
| $D_1$                               | execute |       | write*  |
| $D_2$                               | execute | read* | execute |
| $D_3$                               | execute | read  |         |

(b)



## Owner:

We also need a mechanism to allow addition of new rights and removal of some rights. The owner right controls these operations. If  $\text{access}(i,j)$  includes the owner right, then a process executing in domain  $D$ , can add and remove any right in any entry in column.

| <div>object</div> <div>domain</div> | $F_1$            | $F_2$          | $F_3$                   |
|-------------------------------------|------------------|----------------|-------------------------|
| $D_1$                               | owner<br>execute |                | write                   |
| $D_2$                               |                  | read*<br>owner | read*<br>owner<br>write |
| $D_3$                               | execute          |                |                         |

(a)

| <div>object</div> <div>domain</div> | $F_1$            | $F_2$                    | $F_3$                   |
|-------------------------------------|------------------|--------------------------|-------------------------|
| $D_1$                               | owner<br>execute |                          | write                   |
| $D_2$                               |                  | owner<br>read*<br>write* | read*<br>owner<br>write |
| $D_3$                               |                  | write                    | write                   |

(b)

## Control:

Copy and owner rights only allow the modification of rights within a column. The addition of control rights, which only apply to domain objects, allow a process operating in one domain to affect the rights available in other domains.

| object \ domain | $F_1$ | $F_2$ | $F_3$   | laser printer | $D_1$  | $D_2$  | $D_3$  | $D_4$          |
|-----------------|-------|-------|---------|---------------|--------|--------|--------|----------------|
| $D_1$           | read  |       | read    |               |        | switch |        |                |
| $D_2$           |       |       |         | print         |        |        | switch | switch control |
| $D_3$           |       | read  | execute |               |        |        |        |                |
| $D_4$           | write |       | write   |               | switch |        |        |                |