OS Chapter 6
Input/ Output

-- R.G.B

I/O

- Basic I/O hardware
  - ports, buses, devices and controllers
- I/O Software
  - Interrupt Handlers, Device Driver, Device-Independent Software, User-Space I/O Software
- Important concepts
  - Three ways to perform I/O operations
    - Polling, interrupt and DMAs

Input / Output Devices

1. **Block Devices**

- e.g. disk drives
- Access blocks of data
- Raw I/O or file-system access
- Memory-mapped file access possible

2. Character Devices:

- E.g. keyboards, serial ports
- Single characters at a time
- Libraries layered on top allow line editing

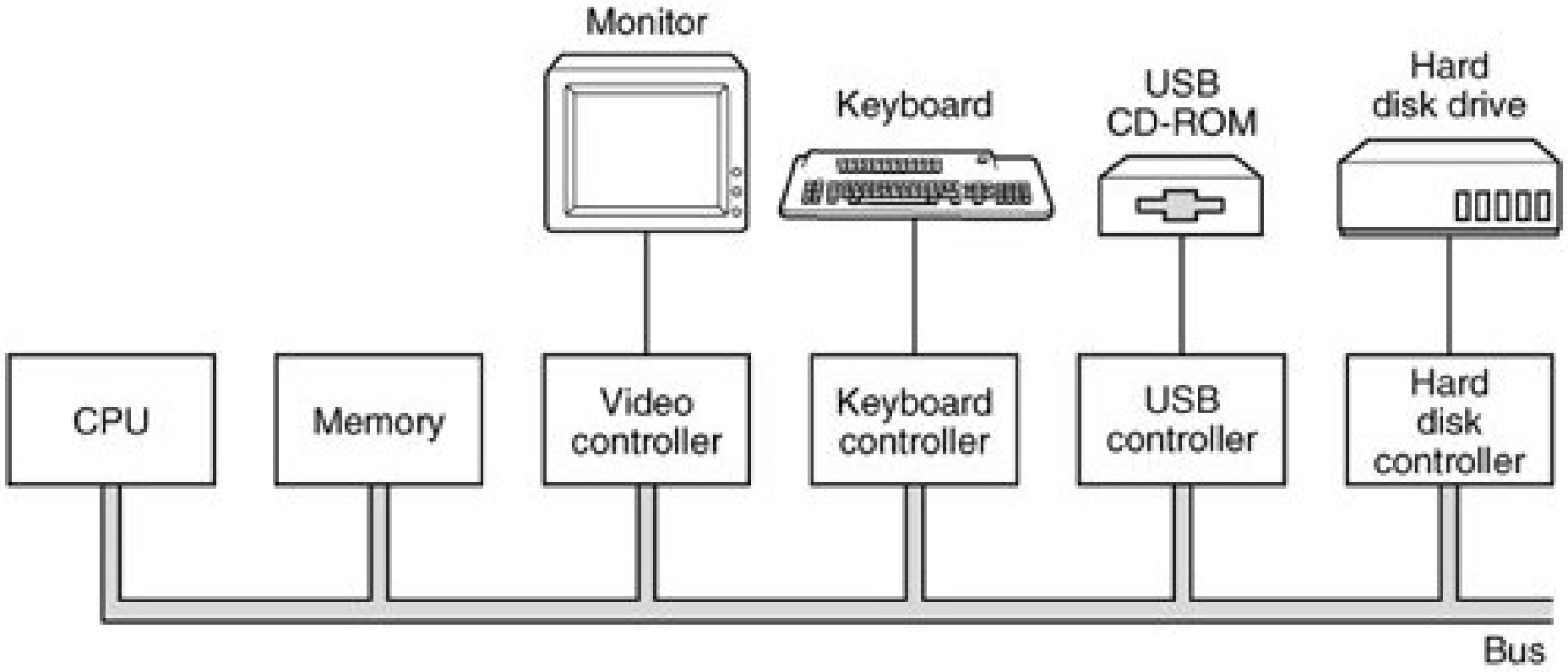
3. Network Devices:

- e.g. Ethernet, Wireless, Bluetooth
- Different enough from block/character to have own interface
- Unix and Windows include socket interface
- Separates network protocol from network operation
- Usage: pipes, FIFOs, streams, queues, mailboxes

Device Controllers

- Hardware unit which is attached with the input/output
- Provides a hardware interface between the computer and the i/o
- How to communicate with input/output devices
- A device controller usually can control several input/output devices
- Most controllers have DMA capability, that means they can directly read/write memory in the system
- **DMA** is a memory-to-device communication method that by passes the CPU

A model for connecting the CPU, memory, controllers, and I/O devices



Memory-mapped Input/Output:

- Controller has a few registers that are used for communicating with the CPU
- 4 registers - status, control, data-in, data-out
  - **Status** - states whether the current command is completed, byte is available, device has an error, etc
  - **Control** - host determines to start a command or change the mode of a device
  - **Data-in** - host reads to get input
  - **Data-out** - host writes to send output
- Size of registers - 1 to 4 bytes

Port Mapped I/O

- Each control register is assigned an I/O port number, an 8- or 16-bit integer
- Using a special I/O instruction such as **IN REG,PORT** the CPU can read in control register PORT store the result in CPU register REG.
- using **OUT PORT,REG** the CPU can write the contents of REG to a control register.
- Most early computers, including nearly all mainframes, such as the IBM 360 and all of its successors worked this ways

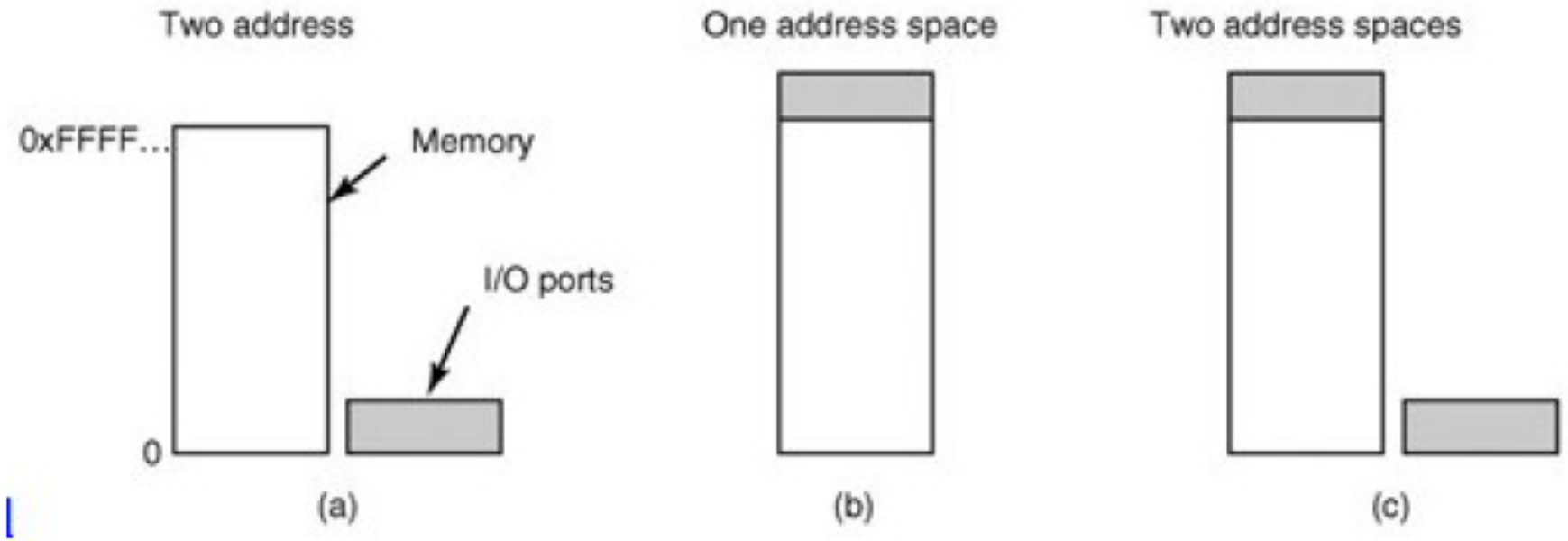
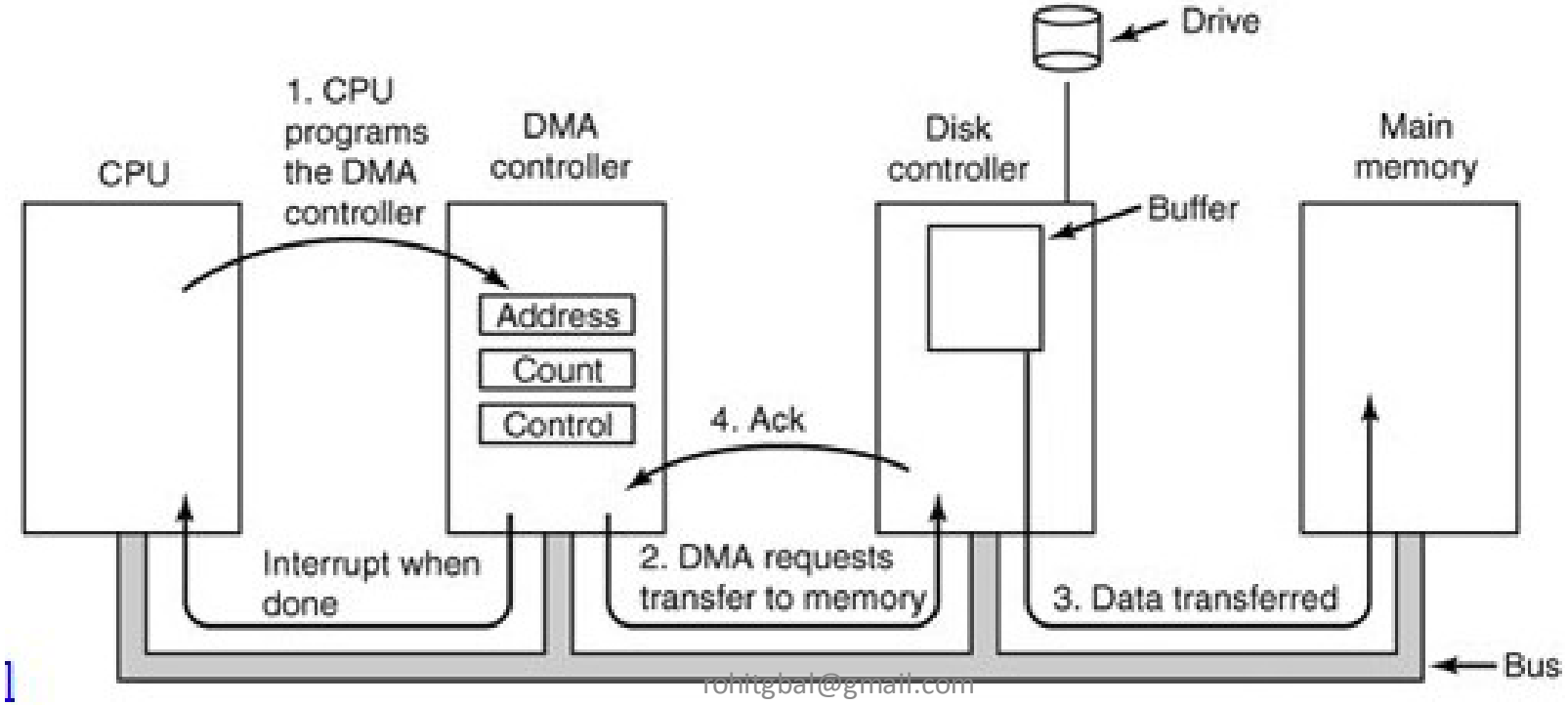


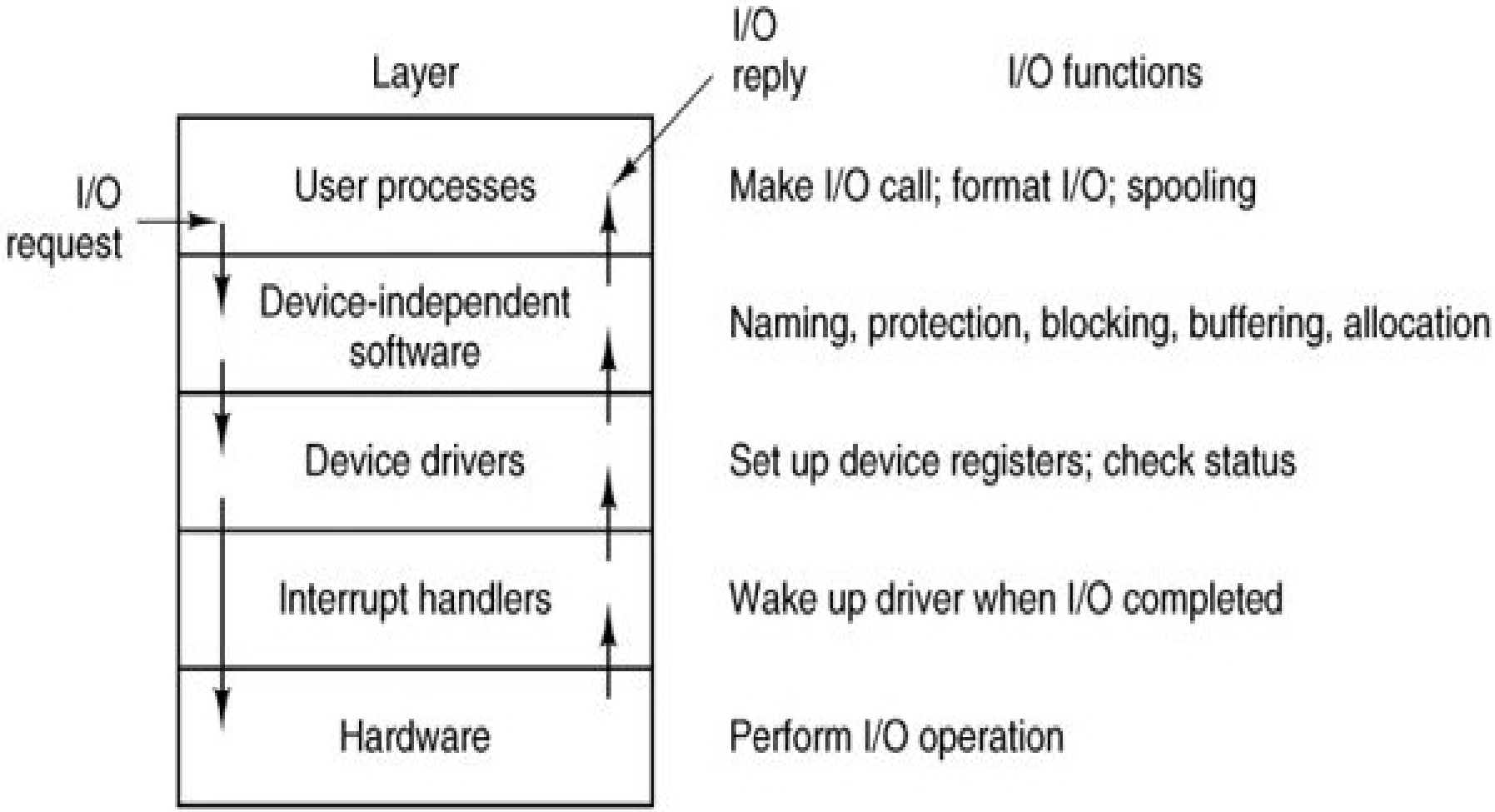
Fig:(a) Separate I/O and memory space. (b) Memory-mapped I/O. (c) Hybrid.

DMA (Direct Memory Access)

- Allowing data to be moved from one location to another in a computer without intervention from the central processor
- More speed for data transfer



Layers of I/O Software



I/O

There are three fundamentally different ways to do I/O.

1. Programmed I/O -- means the program is polling or checking some hardware item
2. Interrupt-driven the same mouse will trigger a signal to the program to process the mouse event.
3. Direct Memory access

Device Driver:

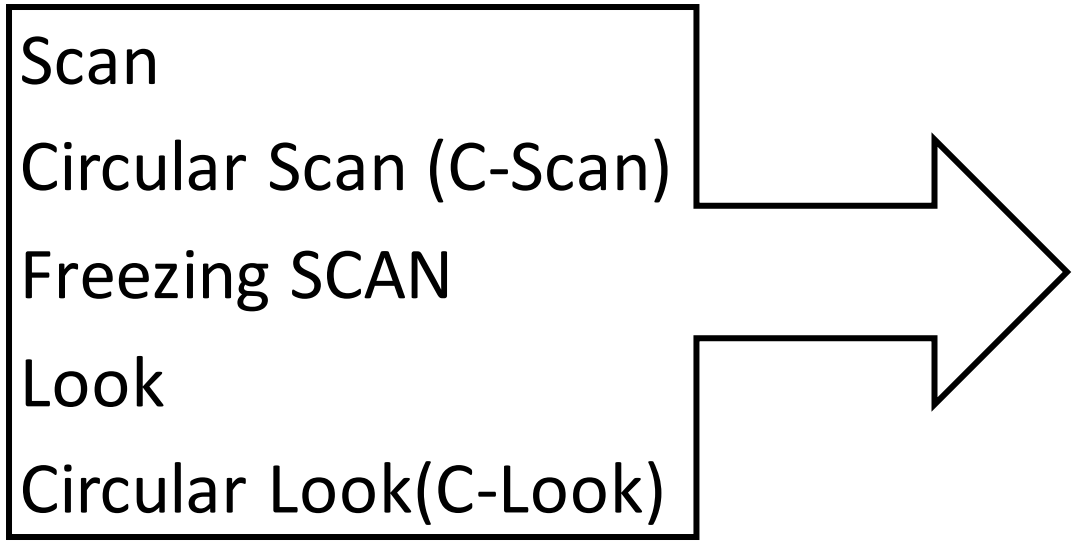
- Device driver or software driver is a computer program allowing higher-level computer programs to interact with a hardware device
- A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects
- Drivers are hardware-dependent and operating-system specific
- They usually provide the interrupt handling

Disk

- **HDD is used to store data**
- **Cylinders(platers)→ Tracks→ sectors**
- **Arm**
- **Head**
- Seek time-- time taken for a disk drive to locate the area on the disk where the data to be read is stored
- Transfer time – Time taken to read data
- Rotational Delay - The time for the proper sector to rotate under the head

Disk Arm Scheduling Algorithm

1. First Come, First Serve(FCFS)
2. Shortest Seek First(SSF)
3. Scan
4. Circular Scan (C-Scan)
5. Freezing SCAN
6. Look
7. Circular Look(C-Look)



Variance of elevator algorithms

First Come First Serve(FCFS)

- FCFS scheduling uses a FIFO queue so that requests are serviced in the order in which they arrive

Pros

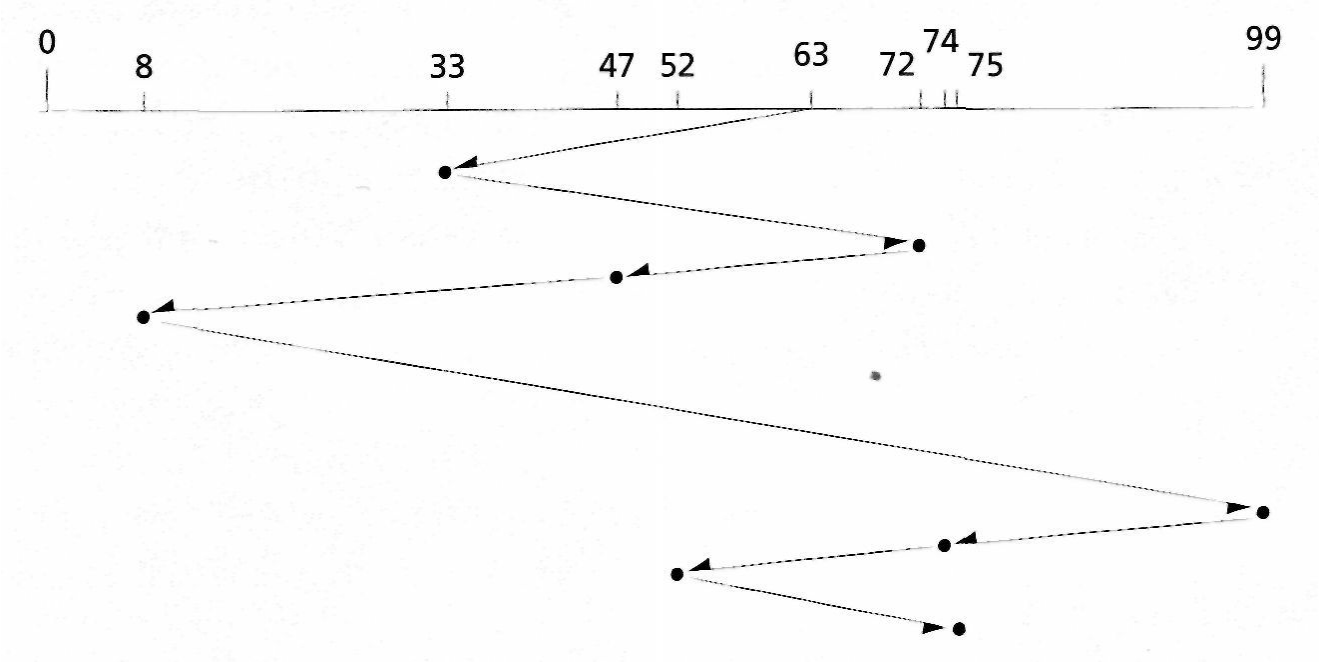
- Ensures that no request can be indefinitely postponed
- All request are treated fairly
- Low time overhead

Cons

- Might have lengthy seek operation
- Low Throughput

First Come First Serve(FCFS)

- **Question**
- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



First Come First Serve(FCFS)

- **Solution**

1. Seek time from Sector 63 to Sector 33 = 30 units
2. Seek time from Sector 33 to Sector 72 = 39 units
3. Seek time from Sector 72 to Sector 47 = 25 units
4. Seek time from Sector 47 to Sector 8 = 39 units
5. Seek time from Sector 8 to Sector 99 = 91 units
6. Seek time from Sector 99 to Sector 74 = 25 units
7. Seek time from Sector 74 to Sector 52 = 22 units

NB: Consider one units for seeking one sector

First Come First Serve(FCFS)

8. Seek time from Sector 52 to Sector 75 = 23 units

- Total seek time =(23+22+25+91+39+25+39+30) 294 units
- Average Seek time=294/8=36.75 units

- *NB: Consider one units for seeking one sector*

Shortest Seek Time First(SSTF)

- Scheduling next services the request that is closest to the read-write head's current cylinder

Pros

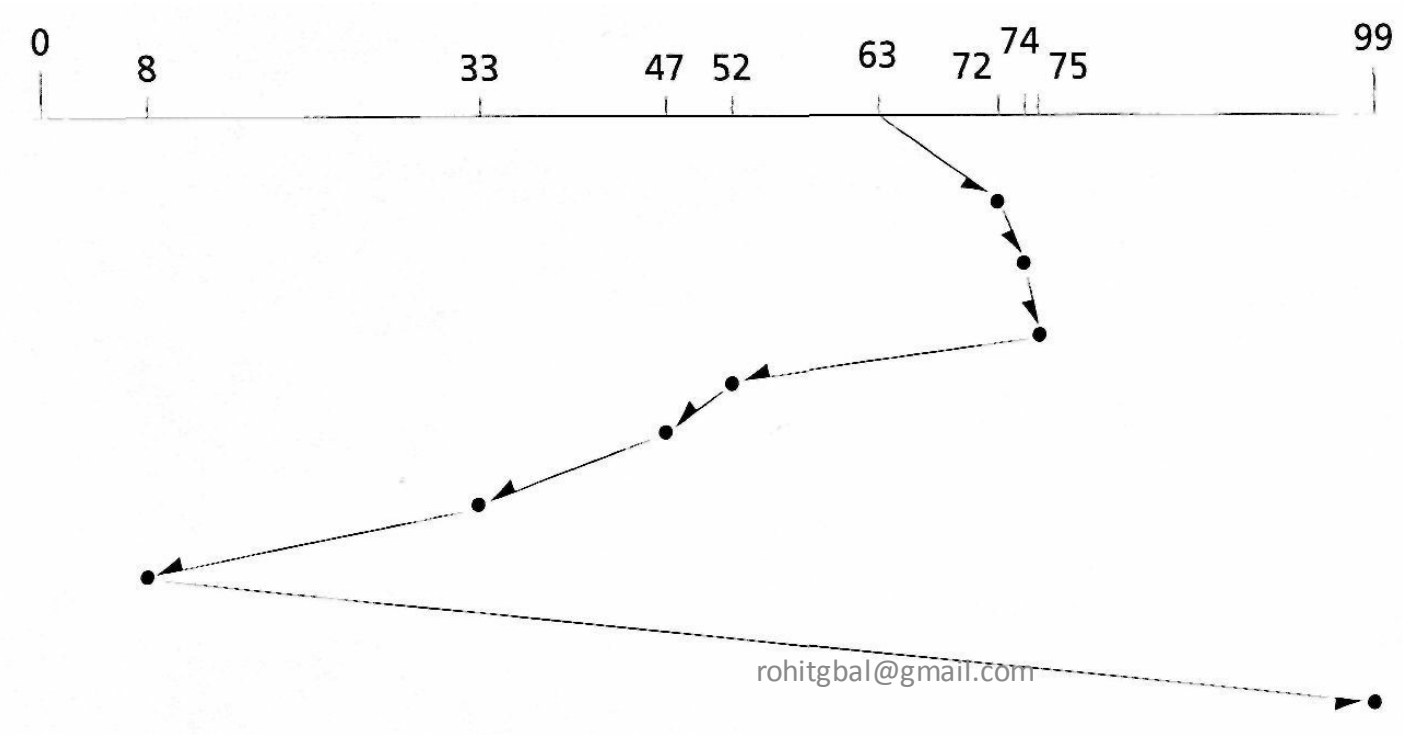
- High mean response time
- High through put

Cons

- Doesn’t ensure fairness
- Some request may lead to indefinite postponement

Shortest Seek Time First(SSTF)

- **Question**
- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



Shortest Seek Time First(SSTF)

- **Solution:**
- First arrange the request to nearest seek time
- Original order of request - 33,72,47,8,99,74,52,75 (63 starting position)
- Shortest seek order – 72,74,75,52,47,33,8,99 (63 starting position)

1. Seek time from Sector 63 to Sector 72 = 9 units
2. Seek time from Sector 72 to Sector 74 = 2 units
3. Seek time from Sector 74 to Sector 75 = 1 units

NB: Consider one units for seeking one sector

Shortest Seek Time First(SSTF)

4. Seek time from Sector 75 to Sector 52 = 23 units
5. Seek time from Sector 52 to Sector 47 = 2 units
6. Seek time from Sector 47 to Sector 33 = 14 units
7. Seek time from Sector 33 to Sector 8 = 25 units
8. Seek time from Sector 8 to Sector 91 = 83 units

Total seek time=(83+25+14+2+23+1+2+9)=159

Average Seek time= 159/8=19.875

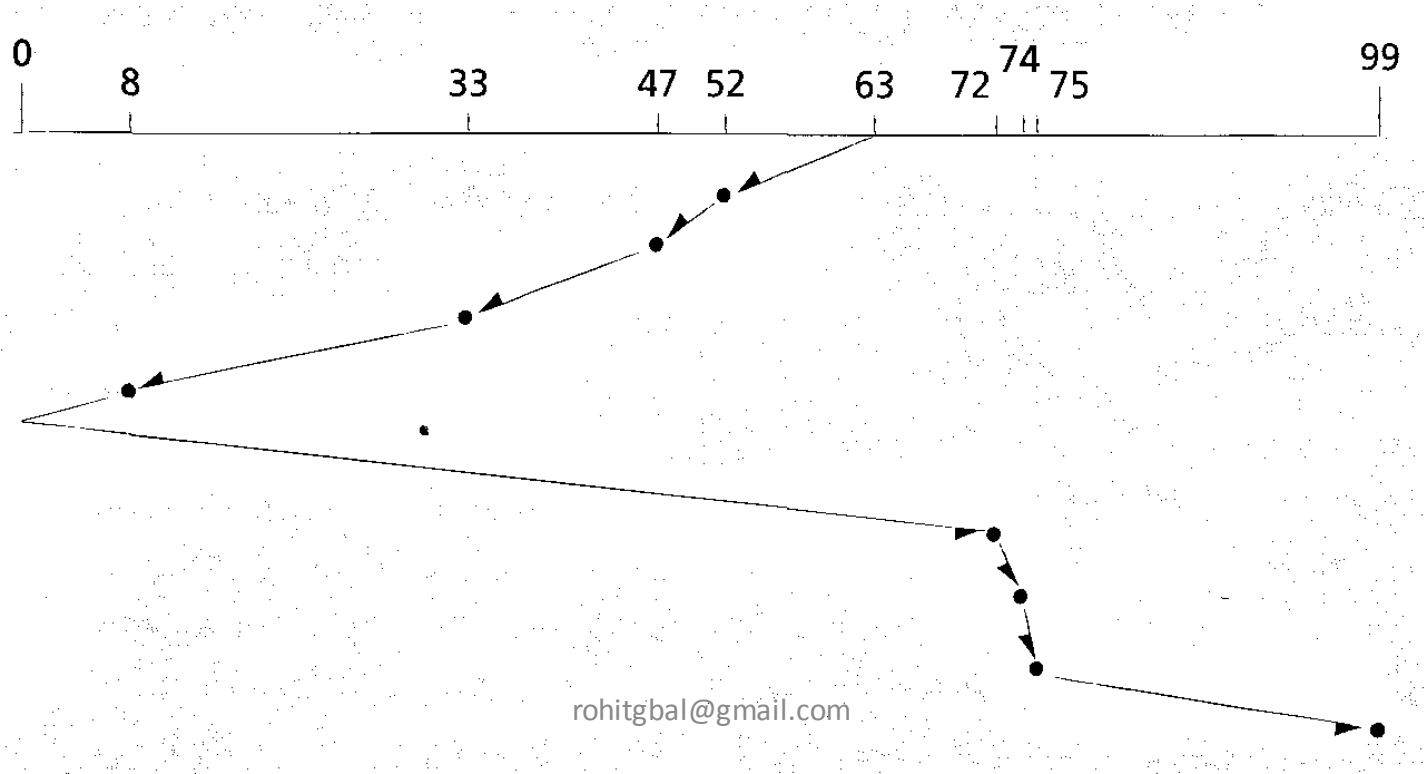
NB: Consider one units for seeking one sector

SCAN(Elevator)

- Strategy to reduce the unfairness and variance of response times exhibited by SSTF
- SCAN chooses the request that requires the shortest seek distance in a **preferred direction(one direction… either outward/inward)**
- if the preferred direction is currently outward, the SCAN strategy chooses the shortest seek distance in the outward direction
- SCAN does not change its preferred direction until it reaches the outermost cylinder or the innermost cylinder.
- In this sense, it is called the **elevator algorithm,** because an elevator continues in one direction servicing requests before reversing direction

SCAN

- **Question**
- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



SCAN

- **Solution**
- Consider disk arm is moving in inward direction(63 starting)
- Original order of request - 33,72,47,8,99,74,52,75 (63 starting position)
- SCAN – (outward- 52,47,33,8,0(outmost point)- direction change – inward-72,74,75,99)

1. Seek time from Sector 63 to Sector 52 = 11 units
2. Seek time from Sector 52 to Sector 47 = 5 units
3. Seek time from Sector 47 to Sector 33 = 14 units

SCAN

4. Seek time from Sector 33 to Sector 8 = 25 units
5. Seek time from Sector 8 to Sector 0 (outmost point) = 8 units
6. Seek time from Sector 0 to Sector 72 = 72 units
7. Seek time from Sector 72 to Sector 74 = 2 units
8. Seek time from Sector 74 to Sector 75 = 1 units
9. Seek time from Sector 75 to Sector 99 = 24 units

Total seek time=(24+1+2+72+8+25+14+5+11)=162 units

Average seek time = 162/8=20.25 units

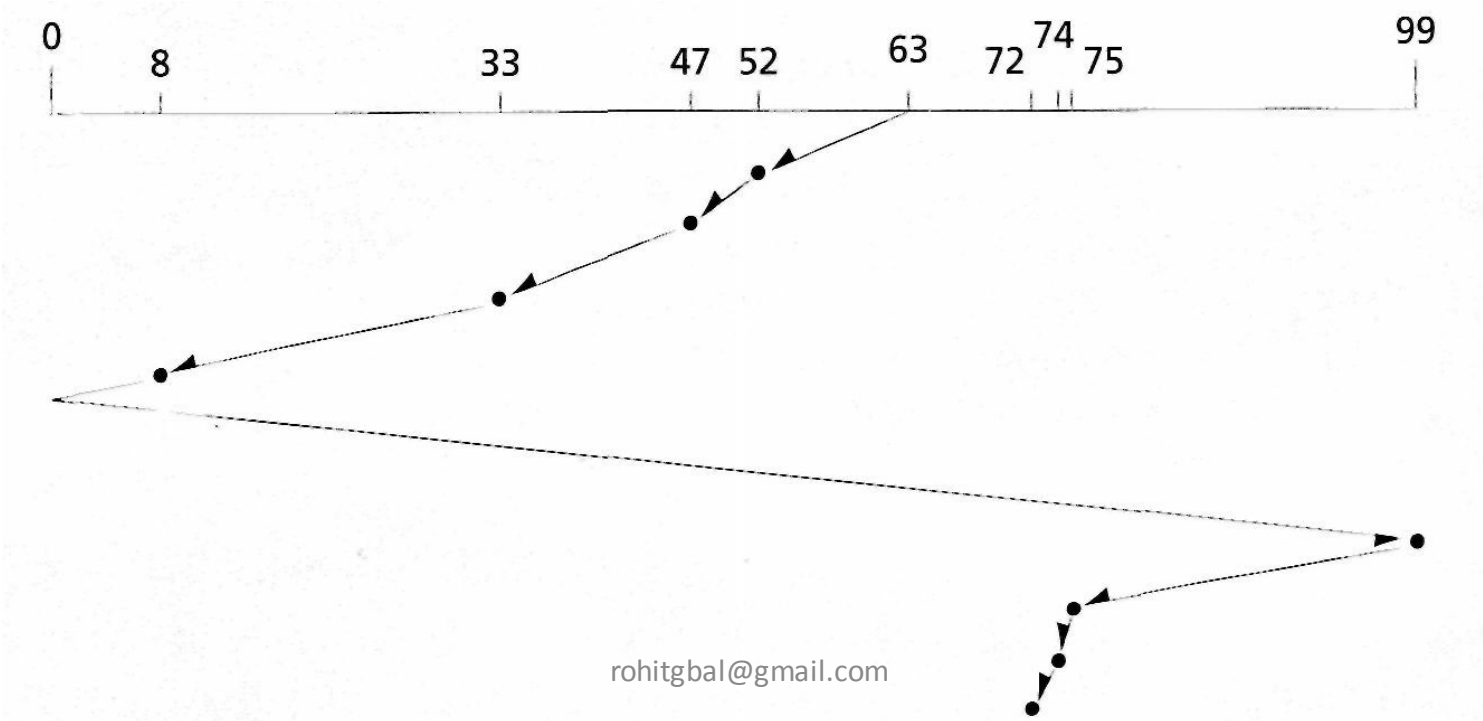
NB : for average seek time from 8 to 0 is not counted as seeking of data, so only 8 data seeking

C-SCAN

- **Circular SCAN (C-SCAN)** modification to the SCAN disk scheduling strategy, the arm moves from the outer cylinder to the inner cylinder, servicing requests on a shortest-seek basis. When the arm has completed its inward sweep, it jumps to the outermost cylinder, then resumes its inward sweep, processing requests.

C-SCAN

- **Question**
- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



C-SCAN

- **Solution**
- Consider disk arm is moving in inward direction(63 starting)
- Original order of request - 33,72,47,8,99,74,52,75 (63 starting position)
- SCAN – (outward- 52,47,33,8,0(outmost point)- direction change – inmost point (99) -99,75,74,72)

1. Seek time from Sector 63 to Sector 52 = 11 units
2. Seek time from Sector 52 to Sector 47 = 5 units
3. Seek time from Sector 47 to Sector 33 = 14 units

C-SCAN

4. Seek time from Sector 33 to Sector 8 = 25 units
5. Seek time from Sector 8 to Sector 0 = 8 units
6. Seek time from Sector 0 to Sector 99 = 99 units
7. Seek time from Sector 99 to Sector 75 = 24 units
8. Seek time from Sector 75 to Sector 74 = 1 units
9. Seek time from Sector 74 to Sector 72 = 2 units

Total Seek Time=(2+1+24+99+8+25+14+5+11)=189 units

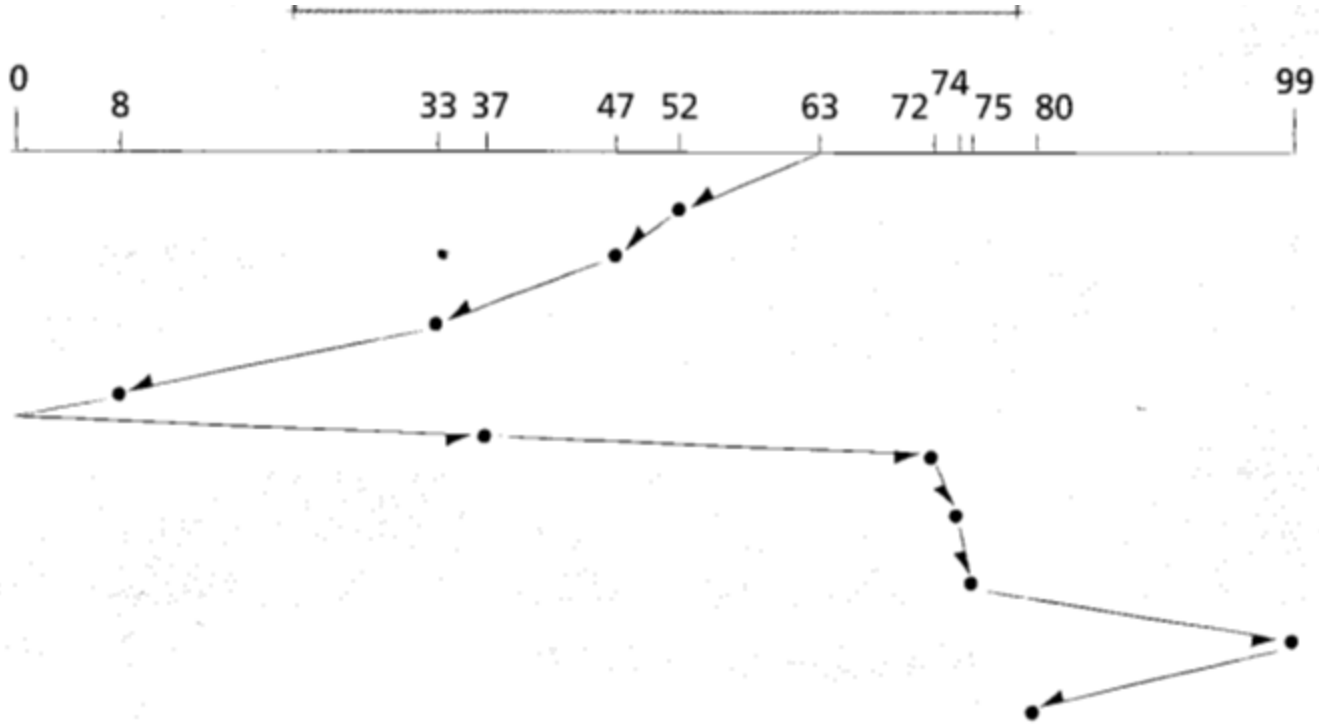
Average Seek time= 189/8=23.625 units

F-SCAN

- **Freezing SCAN** modifications to the SCAN strategy eliminate the possibility of indefinitely postponing requests
- FSCAN uses the SCAN strategy to service only those requests waiting when a particular sweep begins
- Requests arriving during a sweep are grouped together and ordered for optimum service during the return sweep

F-SCAN

- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63(37 came only after 47 processed and 80 read request came only after 75)



F-SCAN

- **Solution**

1. Seek time from Sector 63 to 52 Sector = 11 units
2. Seek time from Sector 52 to 47 Sector = 5 units
3. Seek time from Sector 47 to 33 Sector = 14 units
4. Seek time from Sector 33 to 8 Sector = 25 units
5. Seek time from Sector 8 to 0 Sector = 8 units
6. Seek time from Sector 0 to 37 Sector = 37 units

(Request 37 came to queue only after processing 47)

F-SCAN

7. Seek time from Sector 37 to 72 Sector = 35 units
8. Seek time from Sector 72 to 74 Sector = 2 units
9. Seek time from Sector 74 to 75 Sector = 1 units
10. Seek time from Sector 75 to 99 Sector = 14 units
11. Seek time from Sector 99 to 80 Sector = 18 units

Total seek time=(18+14+1+2+35+37+8+25+14+5+11)= 170 units

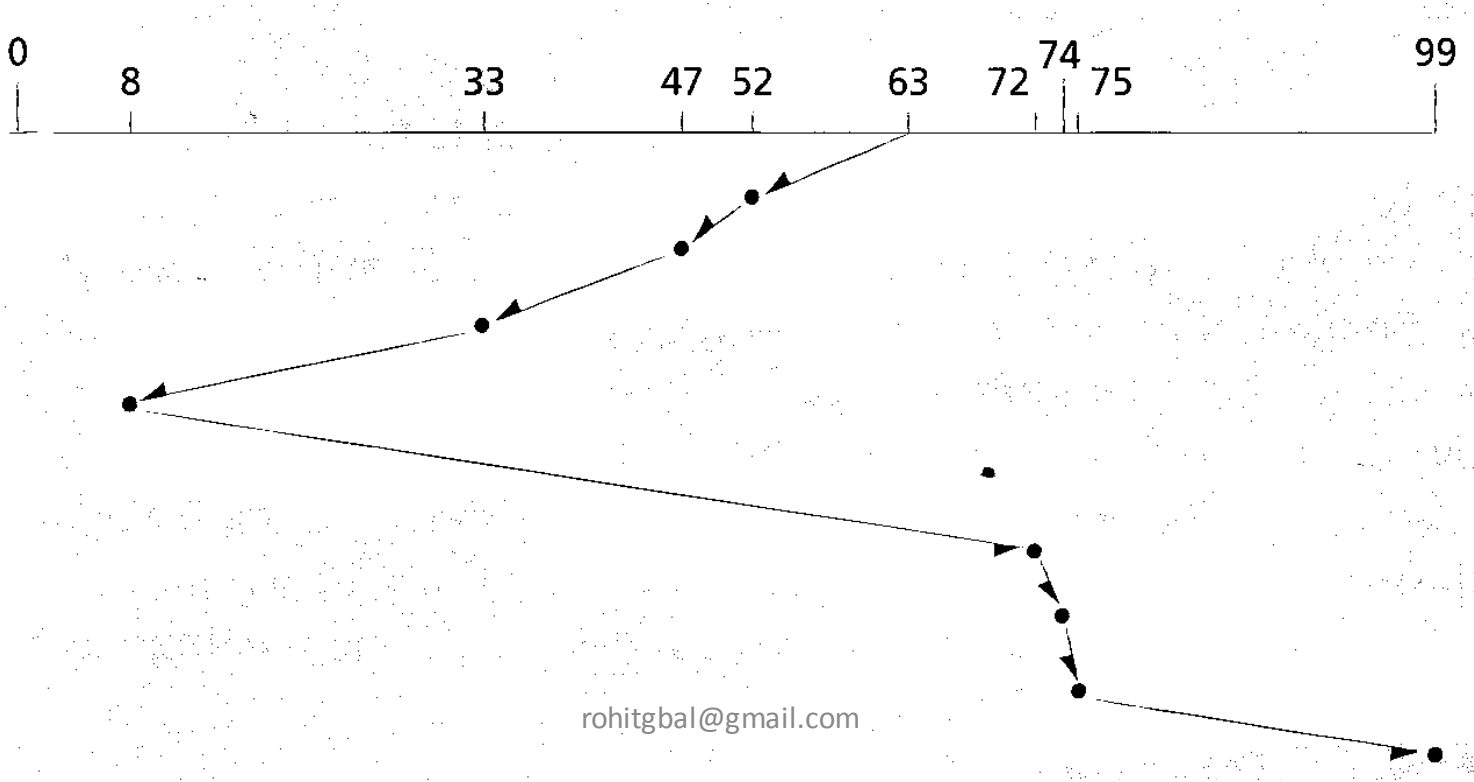
Average seek time = 170/10=17 units

LOOK

- **LOOK** variation of the SCAN strategy "looks" ahead to the end of the current sweep to determine the next request to service.
- If there are no more requests in the current direction, LOOK changes the preferred direction and begins the next sweep
- Only difference from **SCAN** is it doesn’t go to outermost/innermost sector

LOOK

- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



LOOK

Solution

After reaching the last read request i.e 8 th sector arm will change direction inward go to 72

1. Seek time from Sector 63 to 52 Sector = 11 units
2. Seek time from Sector 52 to 47 Sector = 5 units
3. Seek time from Sector 47 to 33 Sector = 14 units
4. Seek time from Sector 33 to 8 Sector = 25 units
5. Seek time from Sector 8 to 72 Sector = 64 units
6. Seek time from Sector 72 to 74 Sector = 2 units

LOOK

7. Seek time from Sector 74 to 75 Sector = 1 units
8. Seek time from Sector 75 to 99 Sector = 25 units

Total seek time=(25+1+2+64+25+14+5+11)=147 units

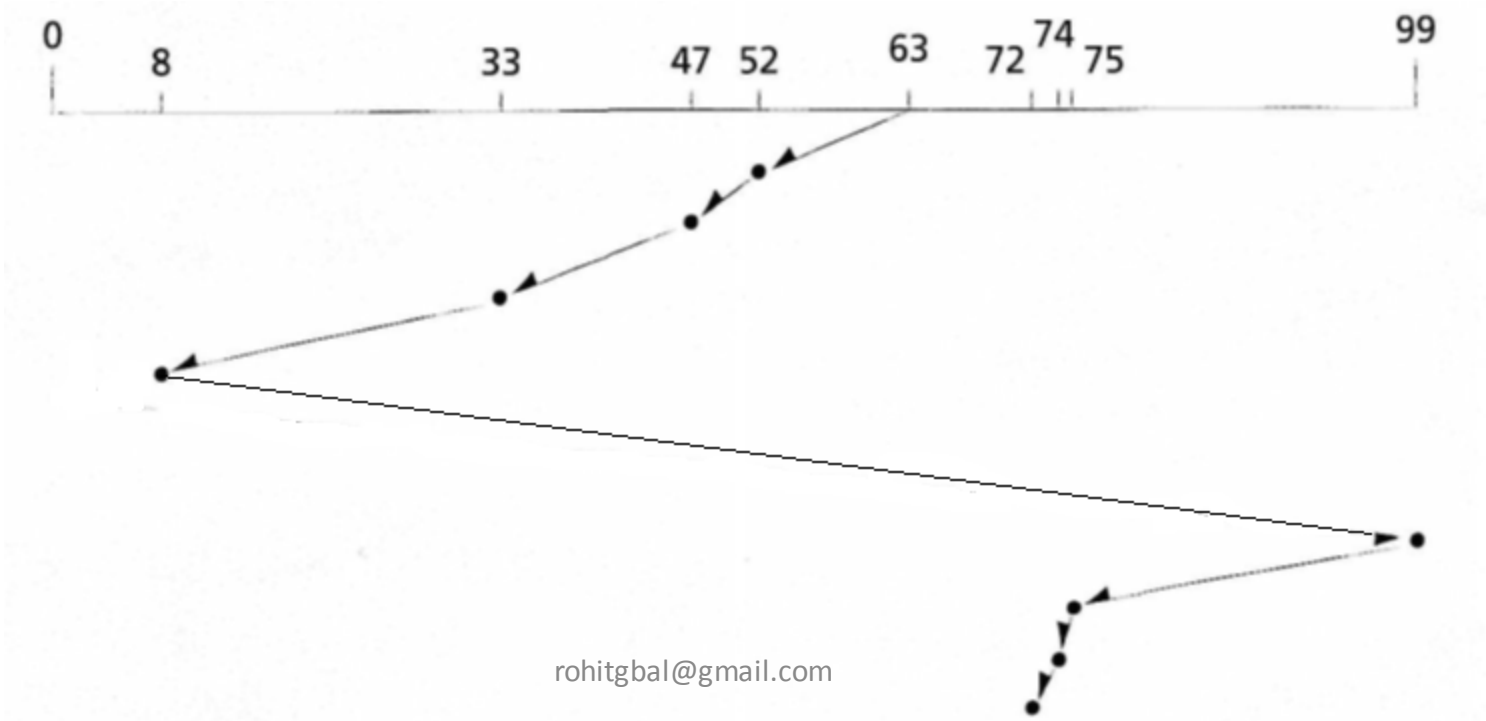
Average Seek time= 18.375 units

C-LOOK

- **Circular LOOK (C-LOOK)** variation of the LOOK strategy uses the same
- technique as C-SCAN to reduce the bias against requests located at the extreme ends of the platters.
- When there are no more requests on a current inward sweep, the read/write head moves to the request closest to the outer cylinder and begins the next sweep

C-LOOK

- **<u>Question</u>**
- Consider disk read request 33,72,47,8,99,74,52,75 and starting position of arm on 63



C-LOOK

- **Solution**
- Consider disk arm is moving in inward direction(63 starting)
- Original order of request - 33,72,47,8,99,74,52,75 (63 starting position)
- SCAN – (outward- 52,47,33,8,- direction change to nearest request in innermost -99,75,74,72)

1. Seek time from Sector 63 to Sector 52 = 11 units
2. Seek time from Sector 52 to Sector 47 = 5 units
3. Seek time from Sector 47 to Sector 33 = 14 units

C-LOOK

4. Seek time from Sector 33 to Sector 8 = 25 units
5. Seek time from Sector 8 to Sector 99 = 99 units
6. Seek time from Sector 99 to Sector 75 = 24 units
7. Seek time from Sector 75 to Sector 74 = 1 units
8. Seek time from Sector 74 to Sector 72 = 2 units

Total Seek Time=(2+1+24+91+8+25+14+5+11)=173 units

Average Seek time= 173/8=21.625 units

Disk Arm Algorithm in Short

Strategy	*Description*
FCFS	Services requests in the order in which they arrive.
SSTF	Services the request that results in the shortest seek distance first.
SCAN	Head sweeps back and forth across the disk, servicing requests according to SSTF in a preferred direction.
C-SCAN	Head sweeps inward across the disk, servicing requests according to SSTF in the preferred (inward) direction. Upon reaching the innermost track, the head jumps to the outermost track and resumes servicing requests on the next inward pass.
FSCAN	Requests are serviced the same as SCAN, except newly arriving requests are postponed until the next sweep. Avoids indefinite postponement.
N-Step SCAN	Services requests as in FSCAN, but services only *n* requests per sweep. Avoids indefinite postponement.
LOOK	Same as SCAN except the head changes direction upon reaching the last request in the preferred direction.
C-LOOK	Same as C-SCAN except the head stops after servicing the last request in the preferred direction, then services the request to the cylinder nearest the opposite side of the disk.

RAID(*Redundant Arrays of Independent Disk*)

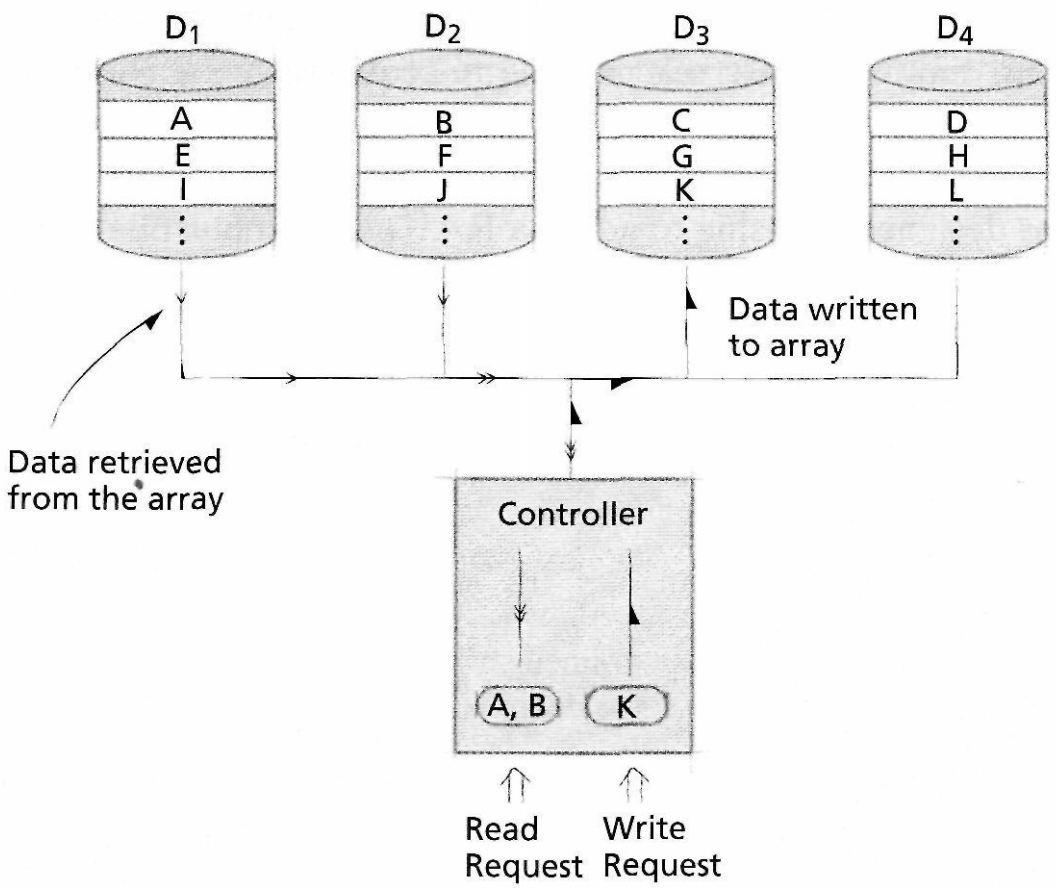
- Techniques that use multiple disks (called an array of disks) that are organized to provide high performance and/or reliability
- Five different organizations, or **levels**, of disk arrays
- **RAID controller(Hardware)** to perform operations quickly
- **RAID** controller, which then performs striping and maintains redundant information as necessary

RAID

- Five RAID levels
- Level 0 (Stripping)
- Level 1 (Mirroring)
- Level 2 (Bit-Level Hamming ECC Parity)
- Level 3 (Bit-level XOR ECC Parity)
- Level 4 (Block-Level XOR ECC Parity)
- Level 5 (Block-Level Distributed XOR ECC Parity)

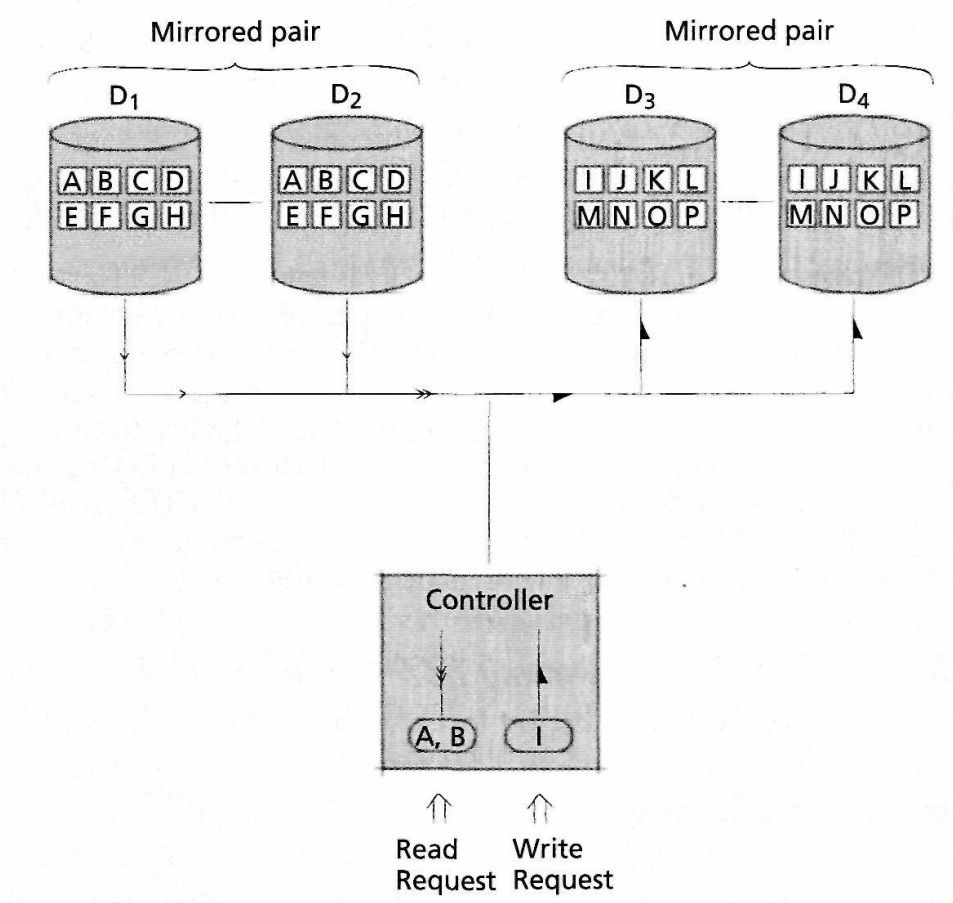
RAID-Level 0 (Striping)

- Striped disk array with no redundancy
- Data will be sliced into pieces and stored in different disk
- If an application requests to read data stored in strips A and B of the array data from both strips can be read simultaneously, because they are on separate disks



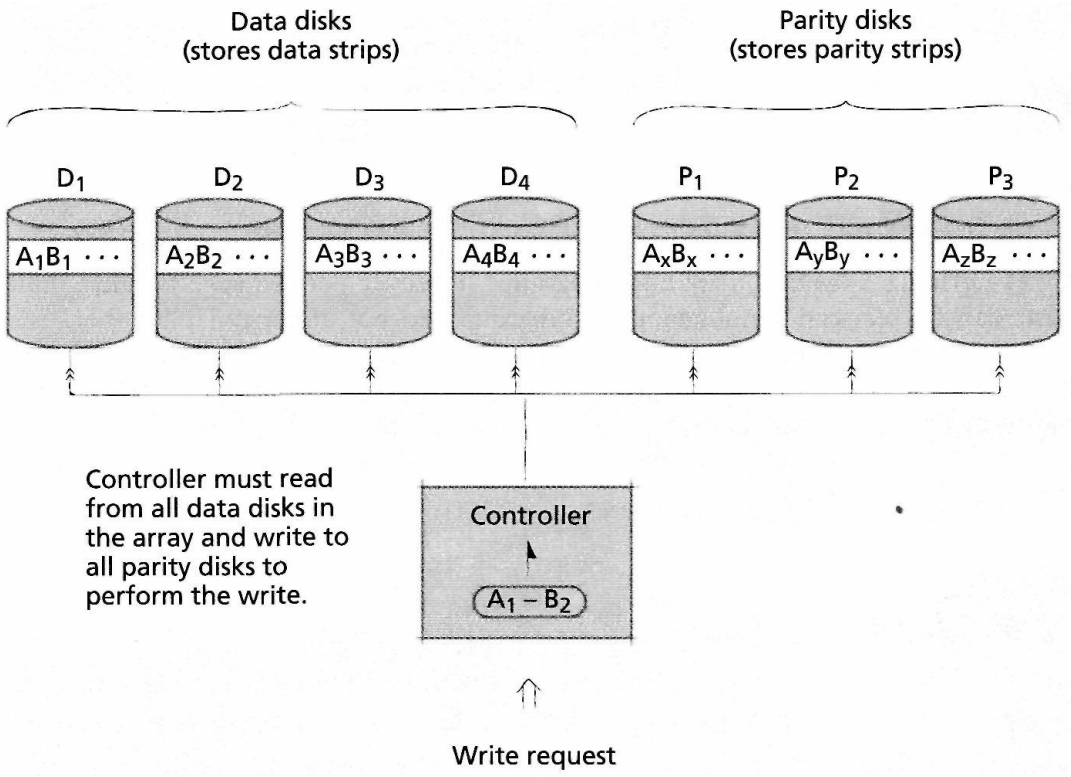
RAID-Level 1 (Mirroring/Shadowing)

- Each disk in the array is duplicated, provides redundancy
- D1 and D2 contain the same data and disks D3 and D4 contain the same data



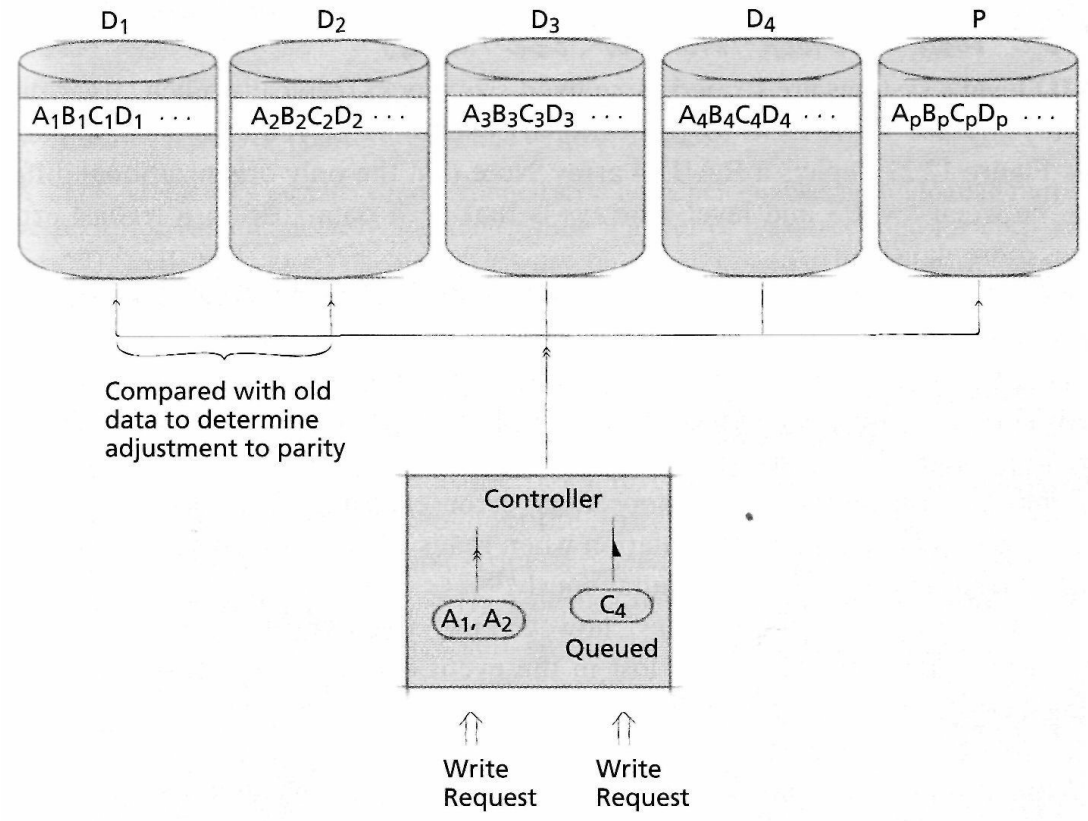
RAID-Level 2(Hamming bit level ECC Parity)

- arrays are striped at the bit level, so each strip stores one bit
- Level 2 arrays are not mirrored, which reduces the storage overhead incurred by level 1 arrays
- Hamming code is used for Error Corrections in bit level



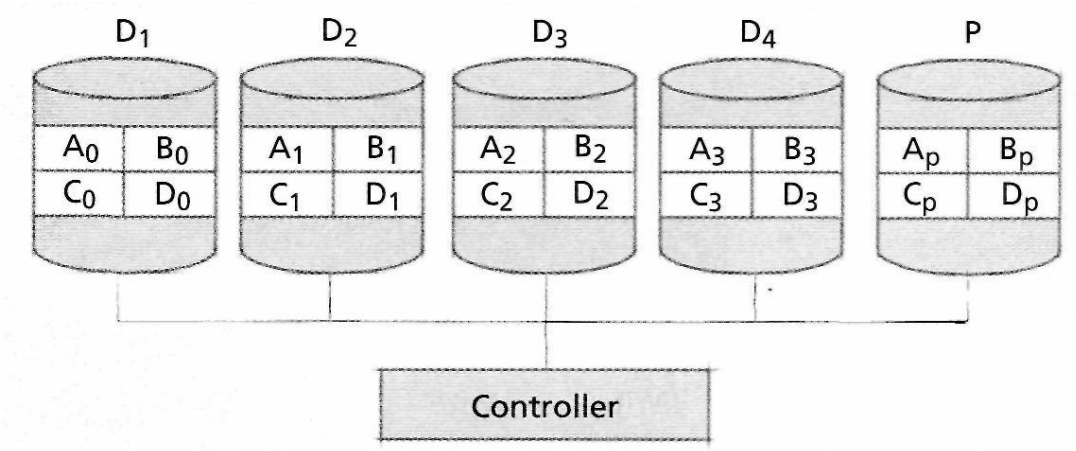
RAID-Level 3 (Bit-level XOR ECC Parity)

- Stripes data at the bit or byte level
- Instead of using Hamming ECC for parity generation, RAID 3 uses **XOR (exclusive or) error-correcting codes**



RAID-Level 4 (Block-level XOR ECC Parity)

- Striped using fixed-size blocks
- Use XOR ECC to generate parity data, which requires a single parity disk
- D1,D2,D3,D4 are the data disk
- And Disk P stores the parity



RAID-Level 5 (Block-level Distributed XOR ECC Parity)

- Striped at the block level
- Use XOR ECC parity, but
- Parity blocks are distributed throughout the array of disks
- Each block has its parity stored in any of the disk array AP,BP,CP,DP are parity blocks