Unit - 5

I/O Management & Disk Scheduling

Page No. _____
Date

I/O devices are the collection of interfaces that different functional units of an information processing system use to communicate with each other.

Inputs are the signals received by the unit and outputs are the signals sent from it. I/O devices are used by a person to communicate with a computer.

Categories of I/O devices are:

1) Human readable:

It is suitable for communicating with the computer user. Ex- Printers, Keyboard

2) Machine Readable:

It is suitable for communicating with electronic equipment. Ex- Disk and tape driver, sensors.

3) Communication:

It is suitable for communicating with remote devices. Ex- Digital drivers, modems.

Techniques / Organization of I/O function

1) Programmed I/O:

Programmed I/O takes place under direct control of CPU. CPU issues a command on behalf of a process to an I/O module. CPU then waits for the operation to be completed before proceeding.

2) Interrupt driven I/O:

In this, CPU issues a command on behalf of a process to an I/O module. If the I/O instruction is non-blocking, CPU continues to execute next instruction from the same process.

If the I/O instruction is blocking, OS takes over suspends the current process and assigns CPU to another process

3) Direct Memory Access (DMA):

DMA module control exchange of data between memory and an I/O module. CPU sends a request to transfer a block of data to the DMA module and is interrupted only after the entire block has been transferred.

Buffer

A buffer is a region of memory used to temporarily hold data while it is being moved from one place to another.

A buffer is a memory area that stores data being transferred between two devices or between a device and application. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream.

Buffering provide adaptations for devices that have different data transfer sizes.

I/O buffering techniques:

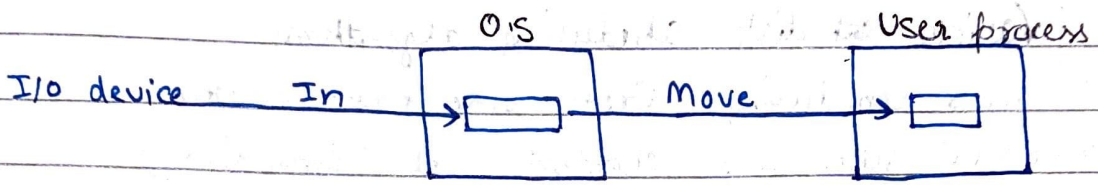
1. Single buffering

When a user process issues an I/O request, the OS assigns a buffer in the system portion of main memory to the operation.

The technique can be used as follows:

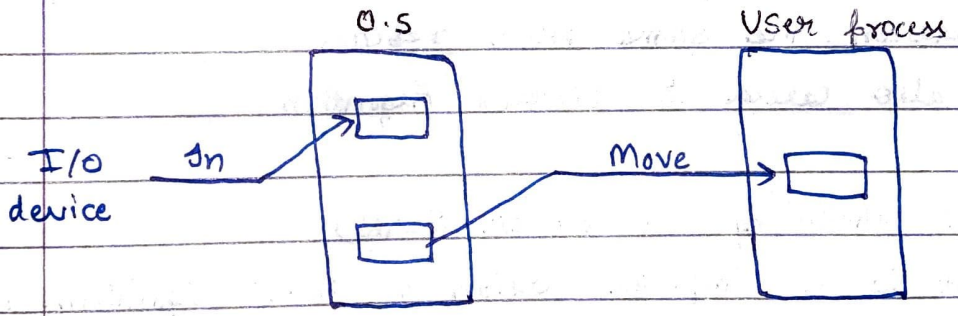
Input transfers are made to the system buffer. When the transfer is complete, the process moves the block into user space and request another block. This is called

reading ahead, it is done in the expectation that the block will be needed sometimes in future. This approach will generally provide a speed up compared to the lack of system buffering. The O.S must keep track of the assignment of the system buffers to user processes.



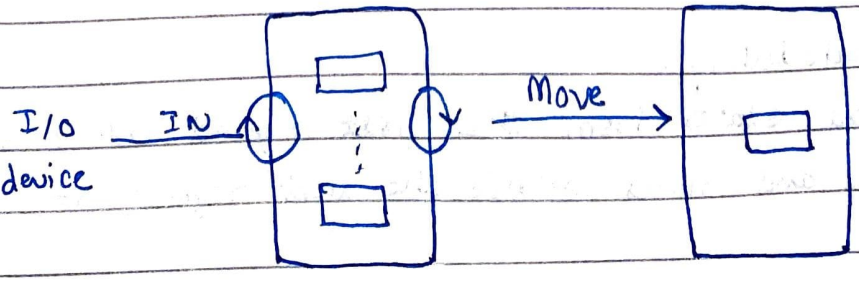
2. Double buffering

A programming technique that uses two buffers to speed up a computer that can overlap I/O with processing. Data in one buffer are being processed while the next set of data is read into the other one.



3. Circular buffering

When more than two buffers are used, the collection of buffers is itself called circular buffer, with each buffer being one unit in the circular buffer.



Services provided in Kernel I/O Subsystem

1) I/O scheduling
2) Buffering
3) Caching
4) Spooling
5) Error handling

Various disk scheduling algorithms:

1. FCFS scheduling (First Come first Serve)
2. SSTF scheduling (Shortest Seek Time first)
3. SCAN scheduling

As the name suggests, this algorithm scans all the cylinders of the disk back and forth. Head starts from one end of the disk and move towards the other end servicing all the requests in between. After reaching the other end, head reverse its direction and move towards the starting end servicing all the requests in between. The same process repeats.
It is also called as Elevator Algorithm.

4. C-SCAN scheduling (Circular - SCAN)

It is an improved version of SCAN algorithm. Head starts from one end of the disk and move towards the other end servicing all the requests in between. After reaching the other end, head reverse its direction. It then returns to the starting end without servicing any request in between. The same process repeats.

5. LOOK Scheduling:

Head starts from the first request at one end of the disk and moves towards the last request at the other

end servicing all the requests in between. After reaching the last request at the other end, head reverse its direction. It then returns to the first request at the starting end servicing all the requests in between. The same process repeats.

Numericals:

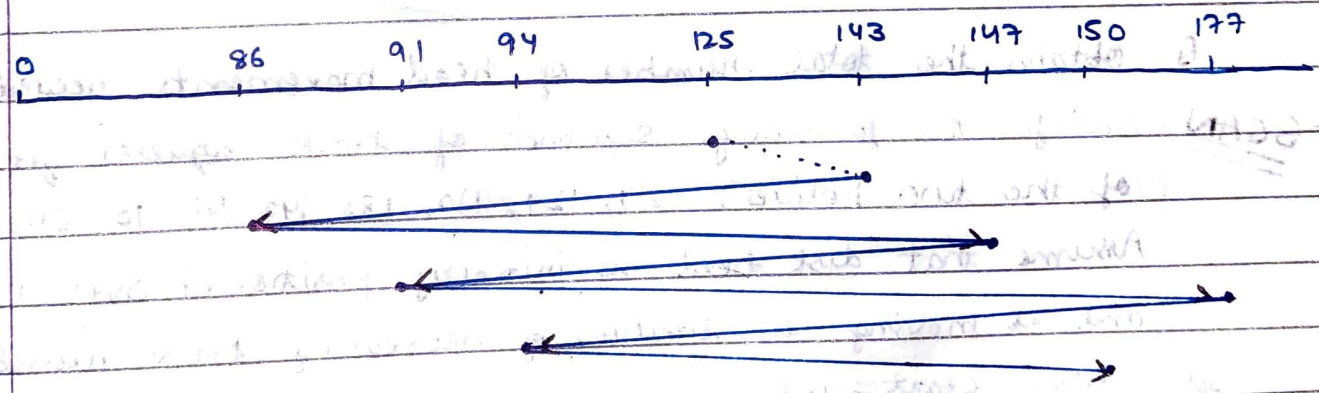
(1) FCFS scheduling

Q. Suppose the moving head disk with 200 tracks is currently serving a request for track 143 and has just finished a request for track 125. If the queue of request is kept in FIFO order 86, 147, 91, 177, 94, 150. What is total head movement for the following scheduling:

- FCFS

Sol. queue = 86, 147, 91, 177, 94, 150

head start = 143



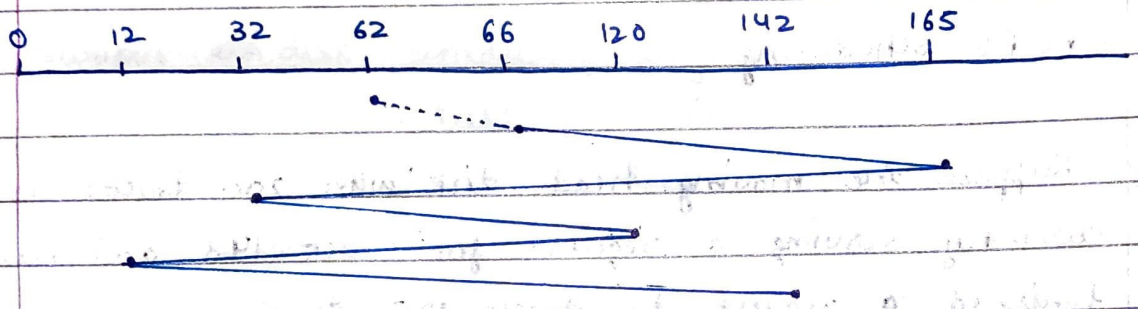
Total distance = (143-86) + (147-86) + (147-91) + (177-91) + (177-94) + (150-94)

= 399.

Page No.
Date

Q. Suppose the head of moving disk is currently servicing a request at track 62. Consider a process requesting to read from the following tracks:
66, 165, 32, 120, 12, 142.

Sol. Queue = 66, 165, 32, 120, 12, 142
head start = 62



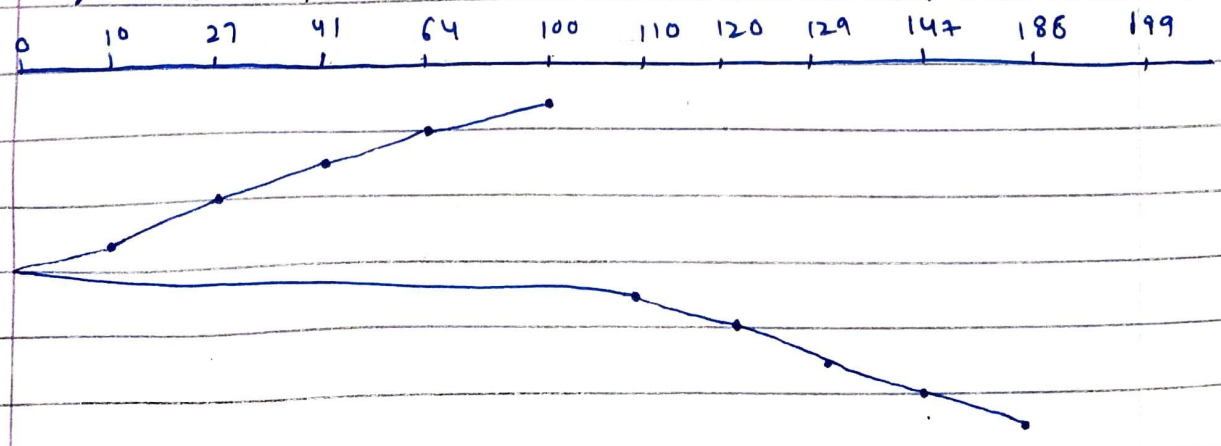
Total head movement = (66-62) + (165-66) + (165-32) + (120-32) + (120-12) + (142-12)
= 562.

(2) SSTF scheduling

Q. Obtain the total number of head movements needed to satisfy the following sequence of track requests for each of the two policies: 27, 129, 110, 186, 147, 41, 10, 64, 120. Assume that disk head is initially positioned over track 100 and is moving in direction of decreasing track number.

SCAN

Sol. head start = 100
queue = 27, 129, 110, 186, 147, 41, 10, 64, 120



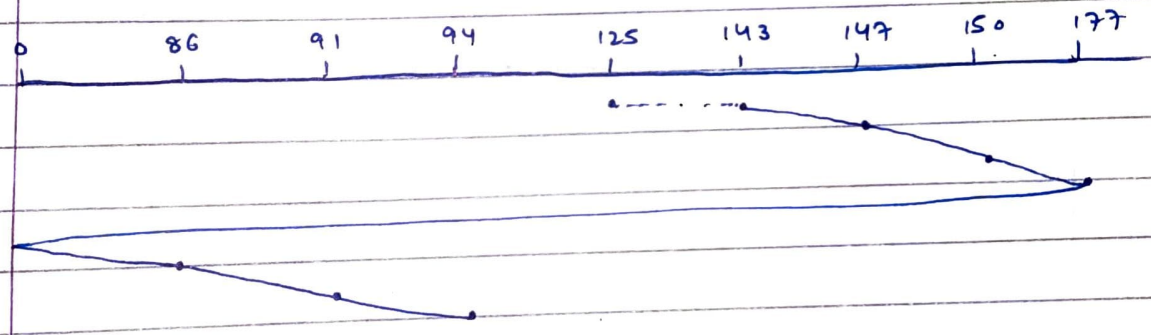
Total head movement = (100-64) + (64-41) + (41-27) + (27-10)
+ (10-0) + (110-0) + (120-110) + (129-120)
+ (147-129) + (186-147)
= 286.

Q. Suppose the moving head disk with 200 tracks is currently serving a request for track 143 and has just finished a request for track 125. If the queue of request is kept in FIFO order 86, 147, 91, 177, 94, 150. what is total head movement for ~~SSTF~~ scheduling?

C-SCAN

Sol. head start = 143

queue = 86, 147, 91, 177, 94, 150

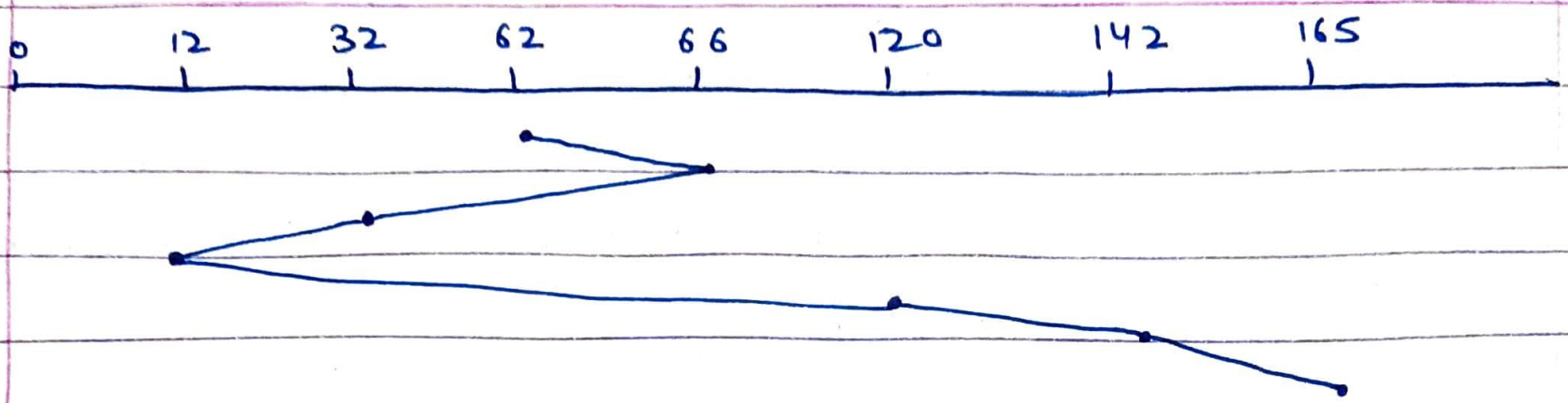


total head movement = (147 - 143) + (150 - 147) + (177 - 150) +
(177 - 0) + (86 - 0) + (91 - 86) + (94 - 91)
= 305.

Q. Suppose the head of moving head disk is currently servicing a request at track 62. Consider a process requesting to read from the following tracks: 66, 165, 32, 120, 12, 142. with SSTF.

Sol. head start = 62

queue = 66, 165, 32, 120, 12, 142.



total head movement = (66-62) + (66-32) + (32-12) + (120-12) + (142-120) + (165-142)

= 212.

(3) SCAN scheduling

Boot Blocks

Operating system requires some information while booting. If the disk is divided into number of partitions, the operating system is stored in the first partition of the disk. If the disk does not contain an OS, this block can be empty. This is called boot block.

Bit Vector:

The free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

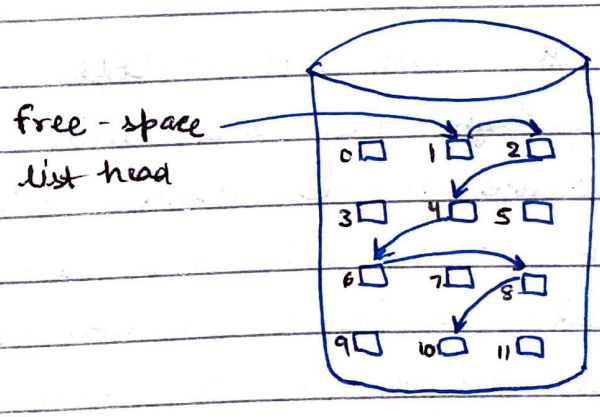
for ex, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free and rest of the blocks are allocated.

The free-space bit map would be:

00111100111111000110000001110000 ...

Linked list:

Another approach to free-space management is to link together all the free disk blocks, keeping a point to the first free block in a special location on the disk and caching it in memory. This first block contains a pointer to the next free disk blocks and so on.



Grouping:

A modification of free list approach stores the addresses of n free blocks in the first free block. The first n-1 of these blocks is actually free. The last block contains the addresses of another n free block and so on. The addresses of a large number of free blocks can now be found quickly.

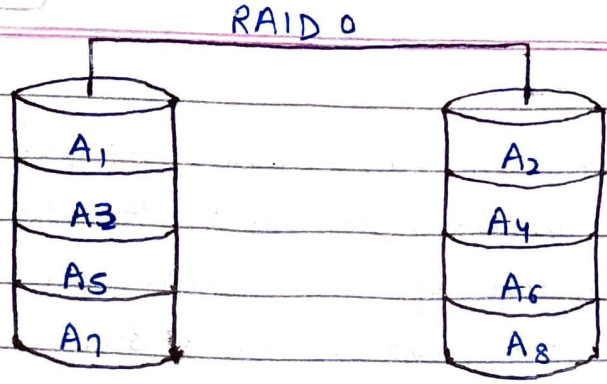
RAID

RAID stands for Redundant Array of Inexpensive disks which was later interpreted to Redundant Array of Independent Disks. This technology is now used in almost all the IT organizations looking for data redundancy and better performance. It combines multiple available disks into 1 or more logical drive and gives you the ability to survive one or more drive failures depending upon the RAID level used.

With the increasing demand in the storage and data world wide the prime concern for the organization is moving towards the security of their data. Here term security does not mean security from vulnerable attacks rather than from hard disk failure and any such relevant accidents which can lead to destruction of data.

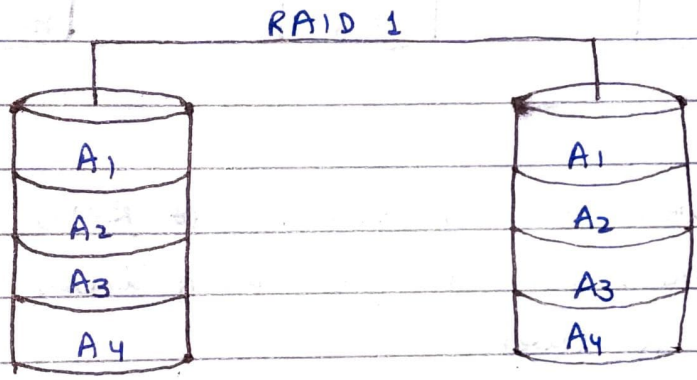
1) RAID 0

This level strips the data into multiple available drives equally giving a very high read and write performance but offering no fault tolerance or redundancy. This level does not provides any of the RAID factor and cannot be considered in an organization looking for redundancy instead it is preferred where high performance is required.



2) RAID 1

This level performs mirroring of data in drive 1 to drive 2. It offers 100% redundancy as array will continue to work even if either disk fails. So organization looking for better redundancy can opt for this solution but again cost can become a factor.

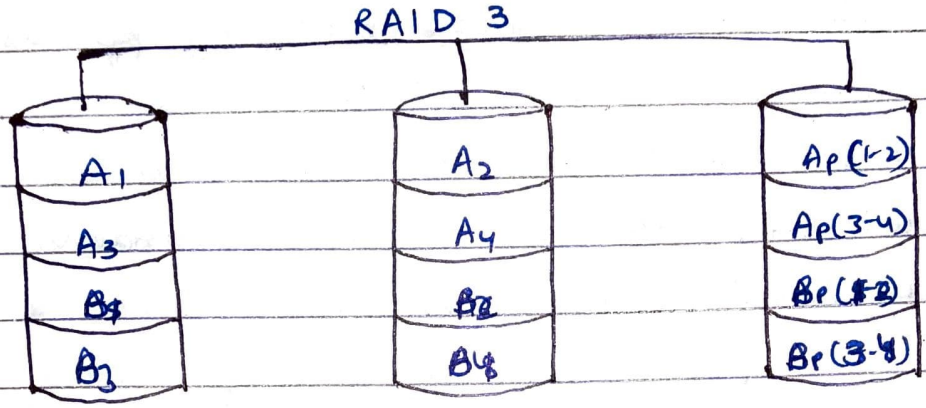


3) RAID 2

This level uses bit-level data stripping rather than block level. To be able to use RAID 2 make sure the disk selected has no self-disk error checking mechanism as this level uses external Hamming code for error detection. This is one of the reason RAID is not in the existence in real IT world as most of the disks used these days come with self error detection. It uses an extra disk for storing all the parity information.

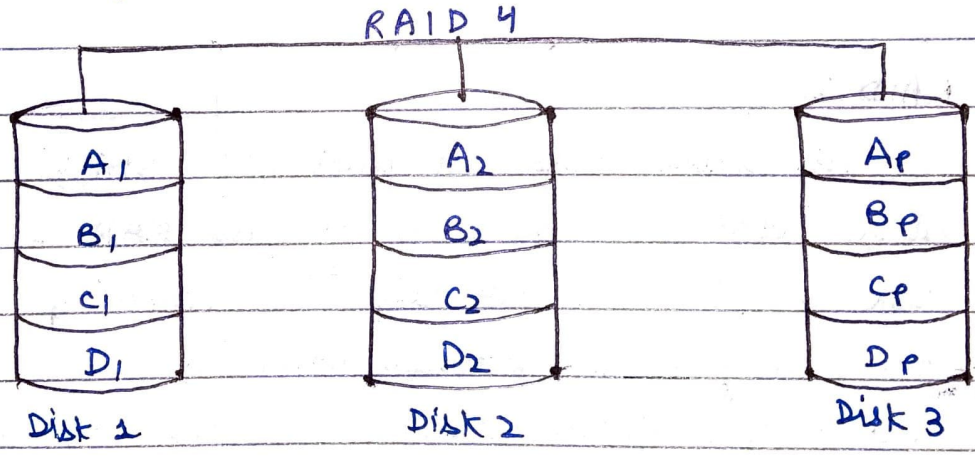
4) RAID 3

This level uses byte level stripping along with parity. One dedicated drive is used to store the parity information and in case of any drive failure the parity is restored using this extra drive. But in case the parity drives crashes then the redundancy gets affected again so not much considered in organizations.



5) RAID 4

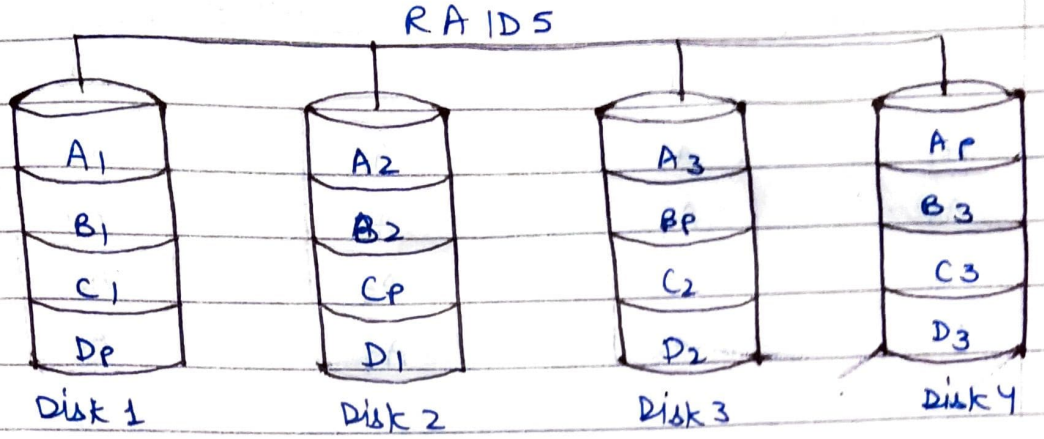
This level is very similar to RAID 3 apart from the feature where RAID 4 uses block level stripping rather than byte level.



6) RAID 5

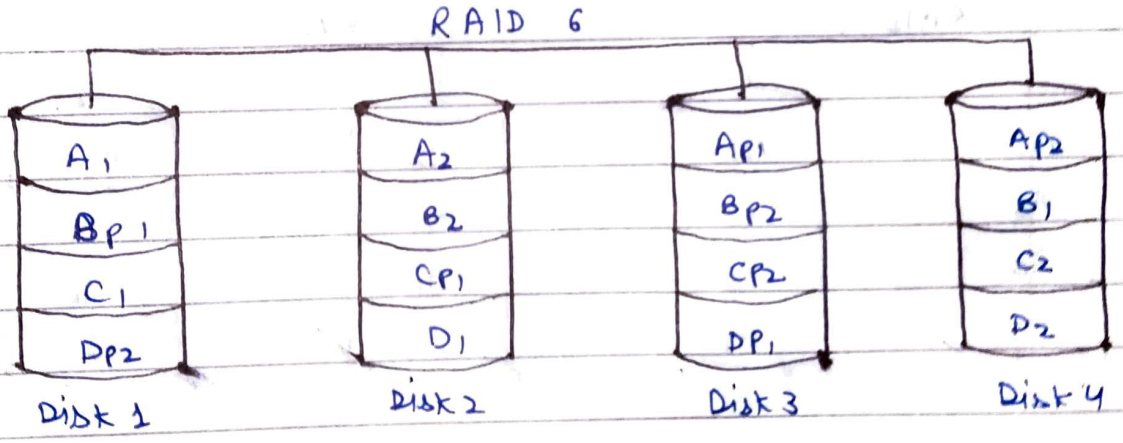
It uses block level striping and with this level distributed parity concept come into the picture leaving behind the traditional dedicated parity as used in RAID 3 and RAID 4. Parity information is written to a different disk in the array for each stripe. In case of single disk failure

data can be recovered with the help of distributed parity without affecting the operation and other read write operations.



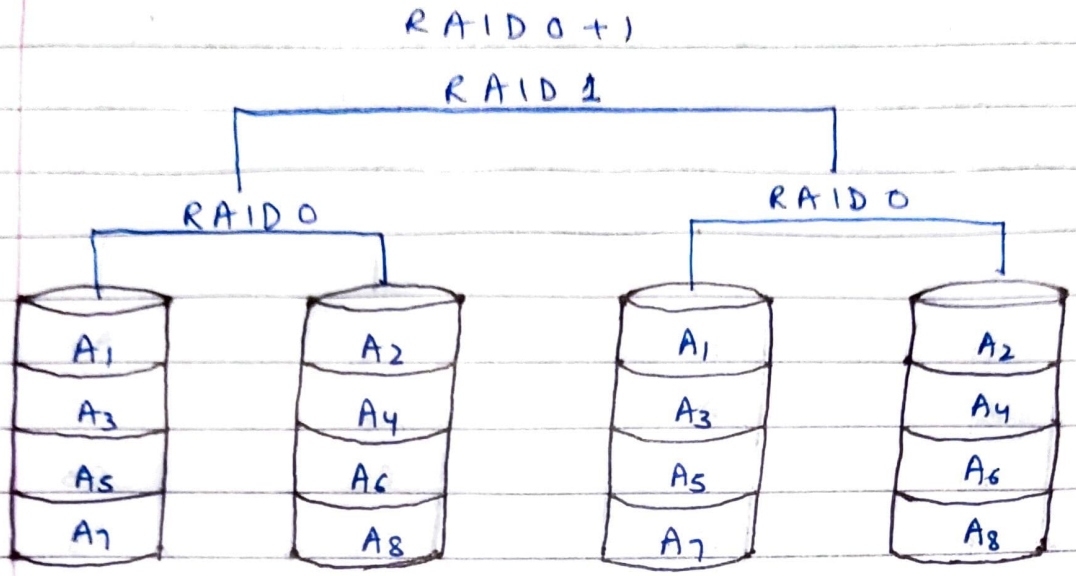
7) RAID 6

This level is an enhancement version of RAID 5 adding extra benefit of dual parity. This level user blocks level stripping with DUAL distributed parity. So now you can get extra redundancy.



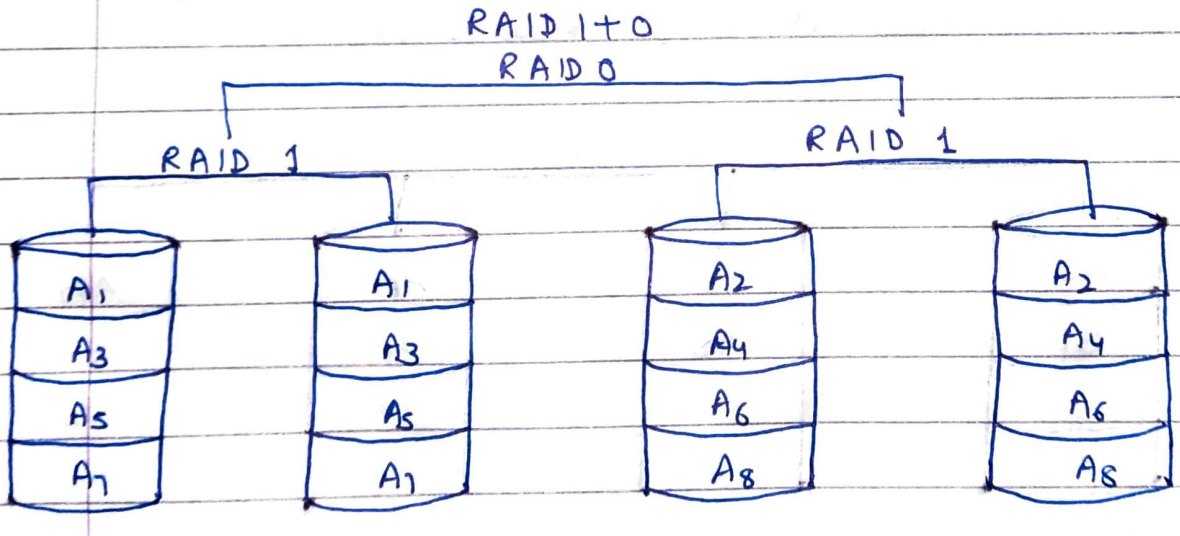
8) RAID 0+1

This level uses RAID 0 and RAID 1 for providing redundancy. Stripping of data is performed before mirroring. In this level, the overall capacity of usable drives is reduced as compared to other RAID levels.



9) RAID 1+0

This level performs mirroring of data prior striping which makes it more efficient and redundant as compared to RAID 0+1. This level can survive multiple simultaneous drive failures.



File Systems

File systems permits users to created data collections. called files with desirable properties:

1) Long term existence:

Files are stored on disk or other secondary storage and do not disappear when a users logs off.

2) Sharable between processes:

Files have names and can have associated access permissions that permit controlled sharing.

3) Structure:

Depending on file systems, a file can have an internal structure that is convenient for particular applications.

File is a collection of data created by user. It provides a means to store data organized as file as well as a collection of function that can be performed on file.

Operations performed on file systems:

1. Create
2. Delete
3. Open
4. Close
5. Read
6. write

File contains various attributes:

1. Name of the file
2. Identifier
3. Type
4. Location
5. Size
6. Protection
7. Date & time

File Organization Methods:

File organization refers to the way, data is stored in a file. File organization is very important because it determines the methods of access, efficiency, flexibility and storage devices to use.

There are four methods of organizing files:

1. Sequential file organization

Records are stored and accessed in a particular sorted order using a key field. Retrieval requires searching sequentially through the entire file record by record to the end.

2. Random file organization

Records are stored randomly but accessed directly. To access a file which is stored randomly, a record key is used to determine where a record is stored on the storage media.

3. Serial file Organization

Records in a file are stored and accessed one after another.

4. Indexed- Sequential file organization

Similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media.

Access Mechanisms:

1. Sequential Access
2. Direct Access
3. Index Access

Directory

The directory can be viewed as a symbol table that translates file names into their directory entries. We can see that the directory itself can be organized in many ways. The organization must be allowed to insert entries, to delete entries, to search for a named entry.

Directory is implemented in two ways:

1. Linear list

It uses a linear list of file names with pointers to the data blocks. Linear list uses a linear search to find a particular entry. Simple for programming but time consuming to execute

2. Hash table

Hash table decreases the directory search time. Insertion and deletion are also straightforward. Hash table takes the value computed from the file name. Then it returns a pointer to the file name in the linear list. Hash table uses fixed size.

Operations performed on a directory are:

1. Searching a file
2. Create a file
3. Delete a file
4. Rename a file
5. List directory