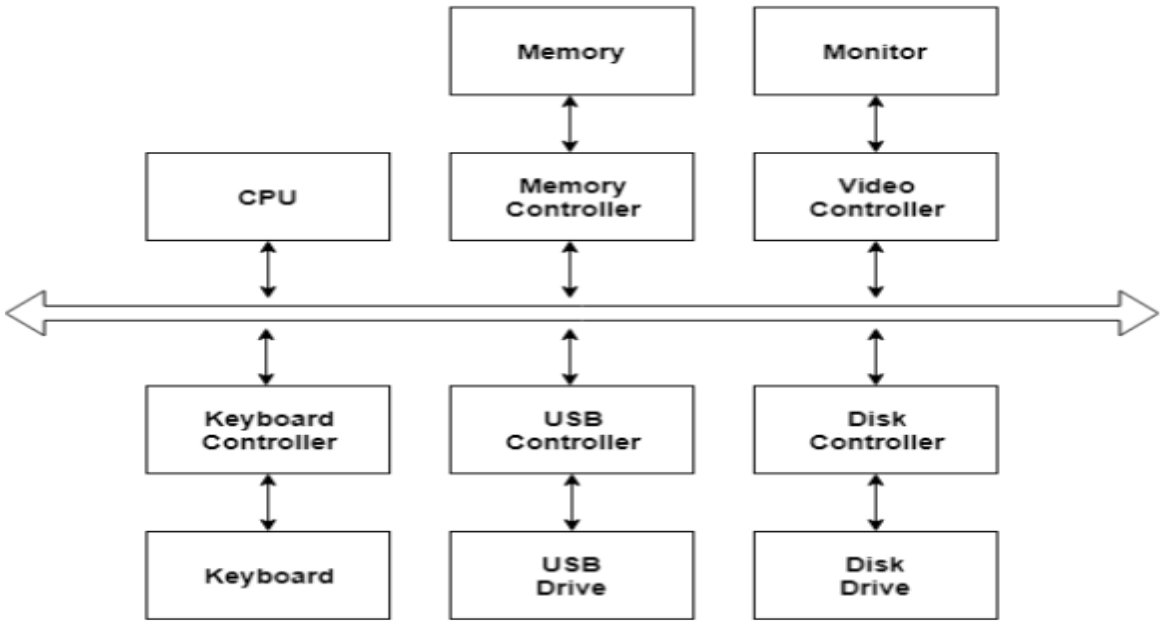
Unit-5th – I/O management and Disk Scheduling

I/O Subsystems

I/O Hardware

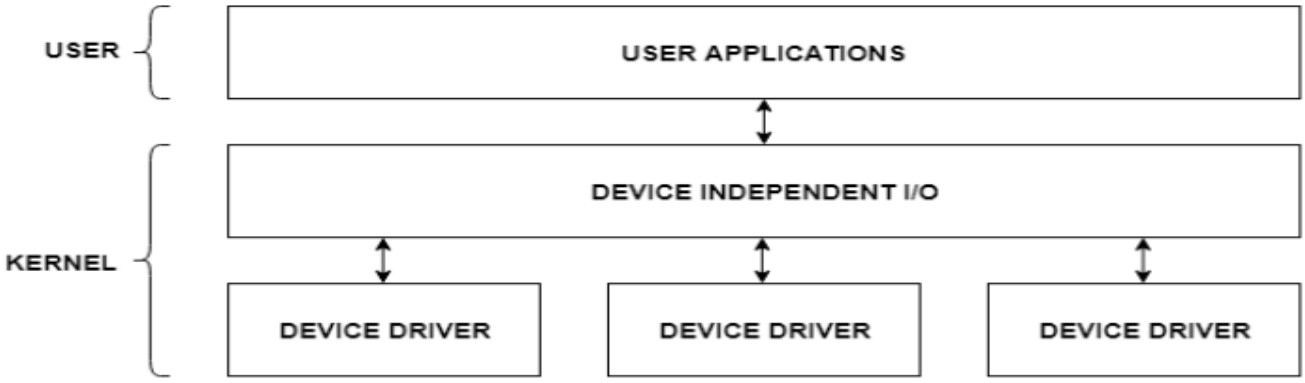
There are many I/O devices handled by the operating system such as mouse, keyboard, disk drive etc. There are different device drivers that can be connected to the operating system to handle a specific device. The device controller is an interface between the device and the device driver.



I/O Application Interface

The user applications can access all the I/O devices using the device drivers, which are device specific codes. The application layer sees a common interface for all the devices.

This is illustrated using the below image −



Most of the devices are either block I/O or character I/O devices. Block devices are accessed one block at a time whereas character devices are accessed one character at a time.

I/O Software

The I/O software contains the user level libraries and the kernel modules. The libraries provide the interface to the user program to perform input and output. The kernel modules provide the device drivers that interact with the device controllers.

The I/O software should be device independent so that the programs can be used for any I/O device without specifying it in advance. For example - A program that reads a file should be able the read the file on a hard disk, floppy disk, CD-ROM etc. without having to change the program each time.

I/O Buffering

The process of temporarily storing data that is passing between a processor and a peripheral. The usual purpose is to smooth out the difference in rates at which the two devices can handle data.

Disk Scheduling

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.
Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

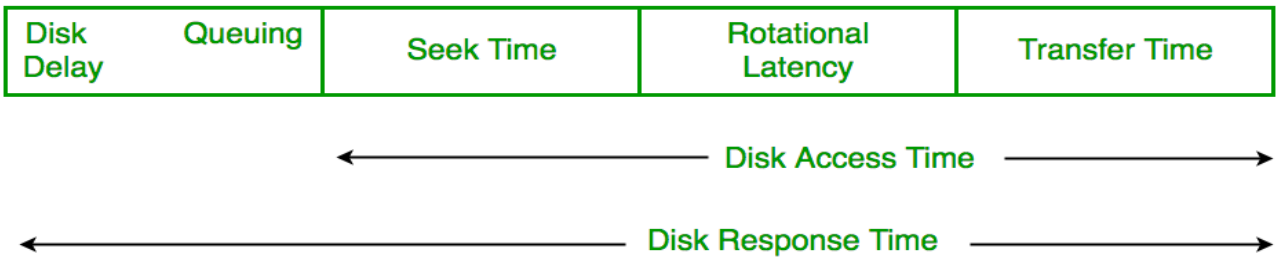
- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

- **Disk Access Time:** Disk Access Time is:

Disk Access Time = Seek Time + Rotational Latency + Transfer Time

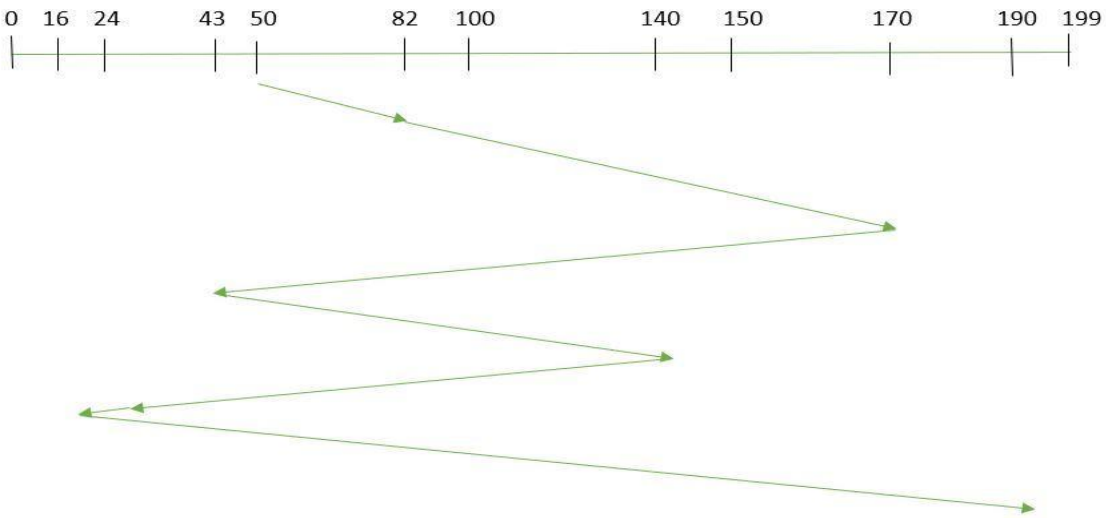


- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Disk Scheduling Algorithms

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue. Let us understand this with the help of an example.

Example: Suppose the order of request is- (82,170,43,140,24,16,190)
And current position of Read/Write head is : 50



So, total seek time:

=(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16)
=642

Advantages:

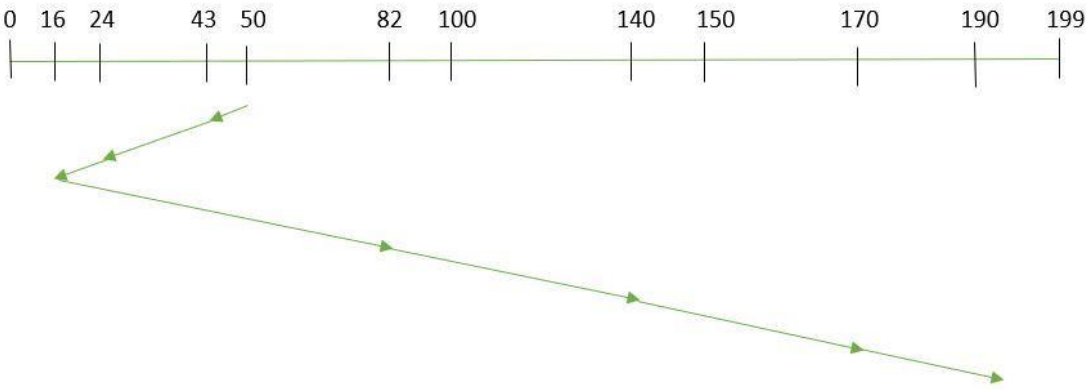
- Every request gets a fair chance.
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service

2. SSTF: In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system. Let us understand this with the help of an example.

Example: Suppose the order of request is- (82,170,43,140,24,16,190) And current position of Read/Write head is : 50



So, total seek time:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170)
=208

Advantages:

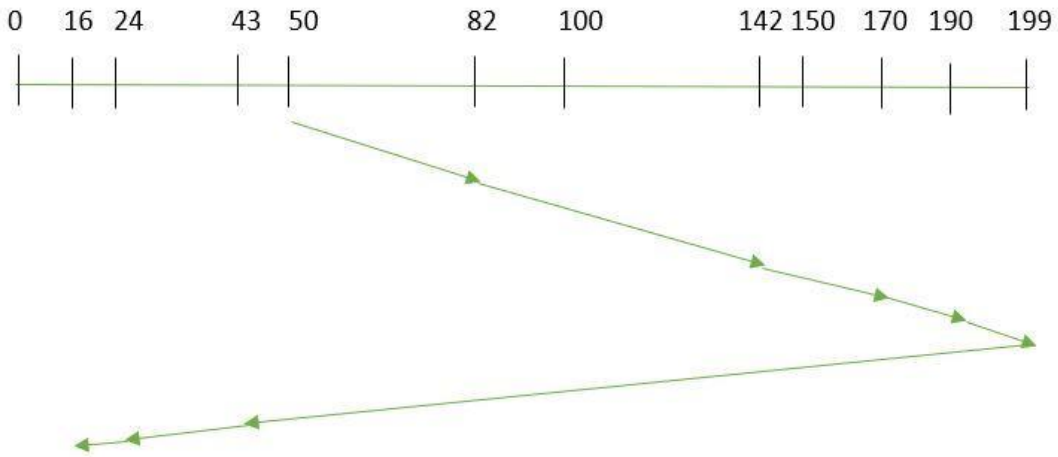
- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favors only some requests

3. SCAN: In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Example: Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**



Therefore, the seek time is calculated as:

=(199-50)+(199-16)
=332

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

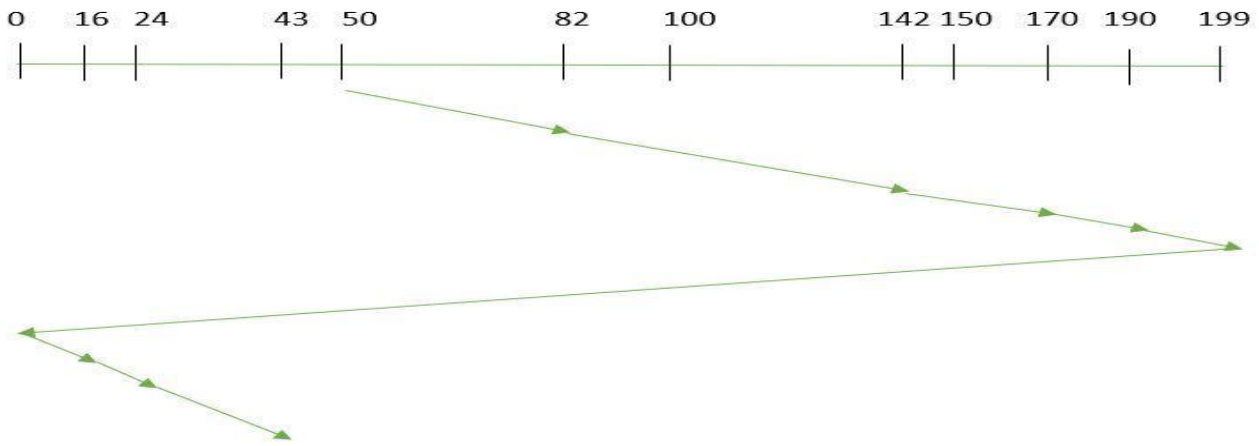
- Long waiting time for requests for locations just visited by disk arm

CSCAN: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Example*:*

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**



Seek time is calculated as:

=(199-50)+(199-0)+(43-0)
=391

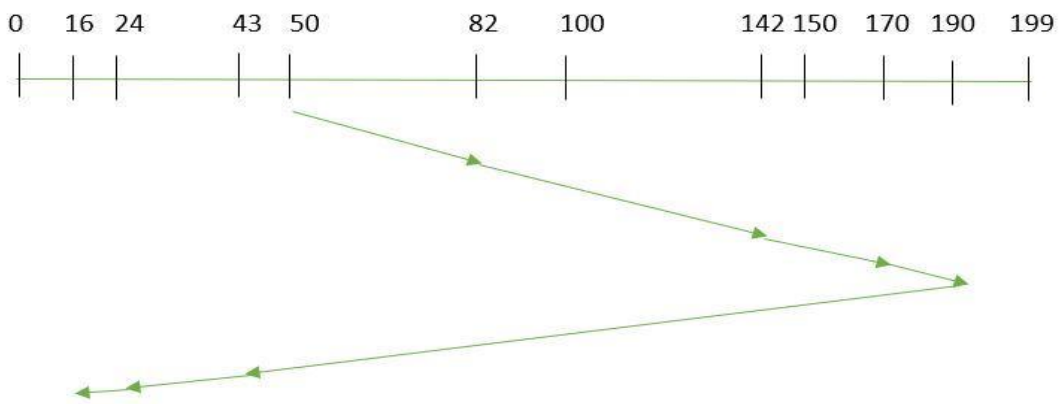
Advantages:

- Provides more uniform wait time compared to SCAN

LOOK: It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**



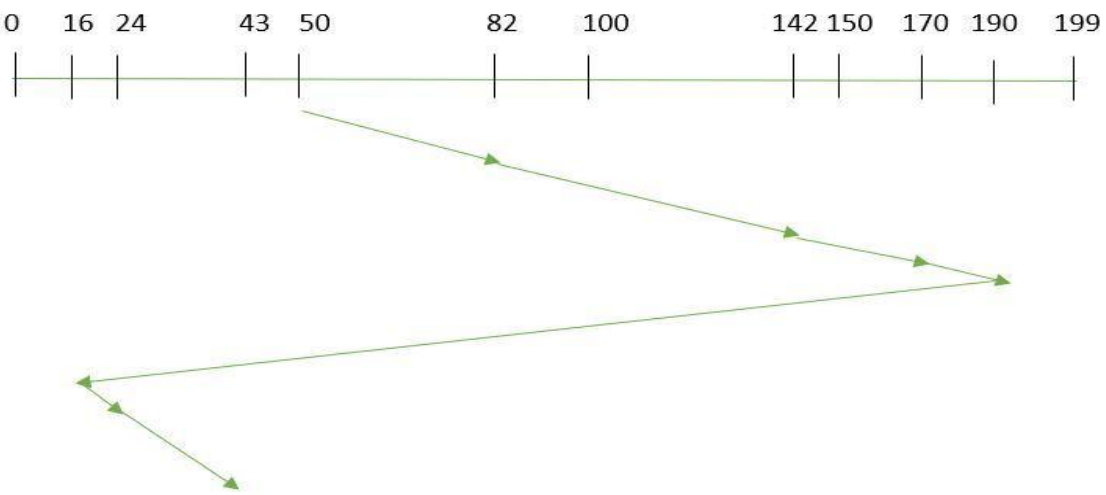
So, the seek time is calculated as:

=(190-50)+(190-16)
=314

CLOOK: As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value"**



So, the seek time is calculated as:

=(190-50)+(190-16)+(43-16)
=341

RSS– It stands for random scheduling and just like its name it is nature. It is used in situations where scheduling involves random attributes such as random processing time, random due dates, random weights, and stochastic machine breakdowns this algorithm sits perfect. Which are why it is usually used for and analysis and simulation.

LIFO– In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.

Advantages

- Maximizes locality and resource utilization
- Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.

N-STEP SCAN – It is also known as N-STEP LOOK algorithm. In this a buffer is created for N requests. All requests belonging to a buffer will be serviced in one go. Also once the buffer is full no new requests are kept in this buffer and are sent to another one. Now, when these N requests are serviced, the time comes for another top N requests and this way all get requests get a guaranteed service

Advantages

- It eliminates starvation of requests completely

FSCAN– This algorithm uses two sub-queues. During the scan all requests in the first queue are serviced and the new incoming requests are added to the second queue. All new requests are kept on halt until the existing requests in the first queue are serviced.

Advantages

- FSCAN along with N-Step-SCAN prevents "arm stickiness" (phenomena in I/O scheduling where the scheduling algorithm continues to service requests at or near the current sector and thus prevents any seeking)

Each algorithm is unique in its own way. Overall Performance depends on the number and type of requests.

Note: Average Rotational latency is generally taken as 1/2(Rotational latency).

Exercise

1. Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing _____ number of requests.

(A) 1
(B) 2
(C) 3
(D) 4

Answer: (C)

Explanation: In Shortest-Seek-First algorithm, request closest to the current position of the disk arm and head is handled first.

In this question, the arm is currently at cylinder number 100. Now the requests come in the queue order for cylinder numbers 30, 85, 90, 100, 105, 110, 135 and 145.

The disk will service that request first whose cylinder number is closest to its arm. Hence 1st serviced request is for cylinder no 100 (as the arm is itself pointing to it), then 105, then 110, and then the arm comes to service request for cylinder 90. Hence before servicing request for cylinder 90, the disk would had serviced 3 requests.

2. Consider an operating system capable of loading and executing a single sequential user process at a time. The disk head scheduling algorithm used is First Come First Served (FCFS). If FCFS is replaced by Shortest Seek Time First (SSTF), claimed by the vendor to give 50% better benchmark results, what is the expected improvement in the I/O performance of user programs?

(A) 50%

(B) 40%

(C) 25%

(D) 0%

Answer: (D)

Explanation: Since Operating System can execute a single sequential user process at a time, the disk is accessed in FCFS manner always. The OS never has a choice to pick an IO from multiple IOs as there is always one IO at a time

3. Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is _____ tracks

(A) 8

(B) 9

(C) 10

(D) 11

Answer: (C)

Explanation: In Shortest seek first (SSTF), closest request to the current position of the head, and then services that request next.

In SCAN (or Elevator) algorithm, requests are serviced only in the current direction of arm movement until the arm reaches the edge of the disk. When this happens, the direction of the arm reverses, and the requests that were remaining in the opposite direction are serviced, and so on.

Given a disk with 100 tracks

And Sequence 45, 20, 90, 10, 50, 60, 80, 25, 70.

Initial position of the R/W head is on track 50.

In SSTF, requests are served as following

Next Served	Distance Traveled
50	0
45	5
60	15
70	10
80	10
90	10
25	65
20	5
10	10

Total Dist = 130

If Simple SCAN is used, requests are served as following

Next Served	Distance Traveled
50	0
60	10
70	10
80	10
90	10
45	65 [disk arm goes to 99, then to 45]
25	20
20	5
10	10

Total Dist = 140

Less Distance traveled in SSTF = 130 - 140 = 10
Therefore, it is **not additional** but it is **less distance** traversed by SSTF than SCAN.

4. Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50×10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512 byte sector of the disk is _____
(A) 6.1

Answer: (A)
Explanation:
Disk latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead

Seek Time? Depends no. tracks the arm moves and seek speed of disk

Rotation Time? depends on rotational speed and how far the sector is from the head

Transfer Time? depends on data rate (bandwidth) of disk (bit density) and the size of request

Disk latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead

Average Rotational Time = (0.5)/(15000 / 60) = 2 milliseconds

[On average half rotation is made]

It is given that the average seek time is twice the average rotational delay

So Avg. Seek Time = 2 * 2 = 4 milliseconds.

Transfer Time = 512 / (50×10^6 bytes/sec)
= 10.24 microseconds
Given that controller time is 10 times the average transfer time
Controller Overhead = 10 * 10.24 microseconds
= 0.1 milliseconds
Disk latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead
= 4 + 2 + 10.24 * 10^{-3} + 0.1 milliseconds
= 6.1 milliseconds

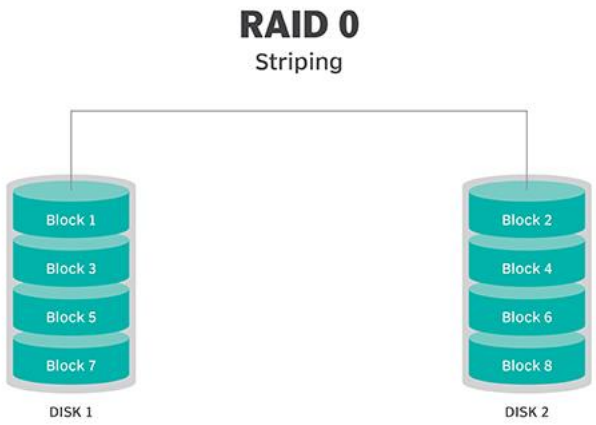
RAID (Redundant Array Of Independent Disk)

RAID (redundant array of independent disks) is a way of storing the same data in different places on multiple hard disks or solid-state drives (SSDs) to protect data in the case of a drive failure. There are different RAID levels, however, and not all have the goal of providing redundancy.

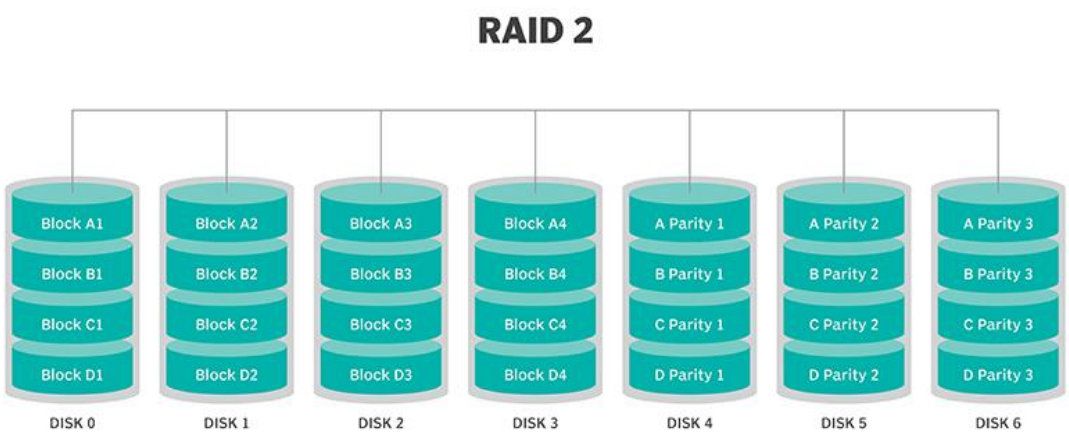
RAID levels

RAID devices use different versions, called levels. The original paper that coined the term and developed the RAID setup concept defined six levels of RAID -- 0 through 5. This numbered system enabled those in IT to differentiate RAID versions. The number of levels has since expanded and has been broken into three categories: standard, nested and nonstandard RAID levels.

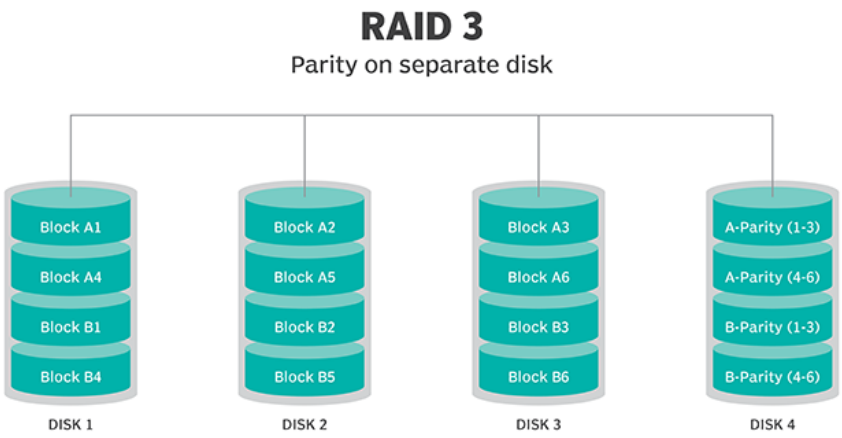
RAID 0. This configuration has striping but no redundancy of data. It offers the best performance, but it does not provide fault tolerance.



RAID 2. This configuration uses striping across disks, with some disks storing error checking and correcting (ECC) information. RAID 2 also uses a dedicated Hamming code parity, a linear form of ECC. RAID 2 has no advantage over RAID 3 and is no longer used

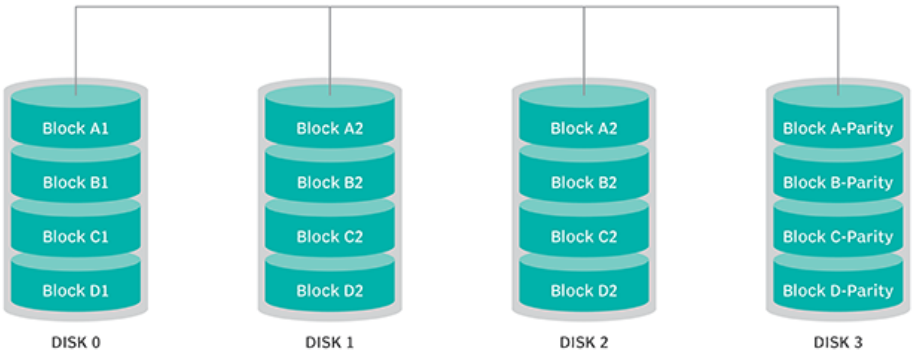


RAID 3. This technique uses striping and dedicates one drive to storing parity information. The embedded ECC information is used to detect errors. Data recovery is accomplished by calculating the exclusive information recorded on the other drives. Because an I/O operation addresses all the drives at the same time, RAID 3 cannot overlap I/O. For this reason, RAID 3 is best for single-user systems with long record applications



RAID 4. This level uses large stripes, which means a user can read records from any single drive. Overlapped I/O can then be used for read operations. Because all write operations are required to update the parity drive, no I/O overlapping is possible.

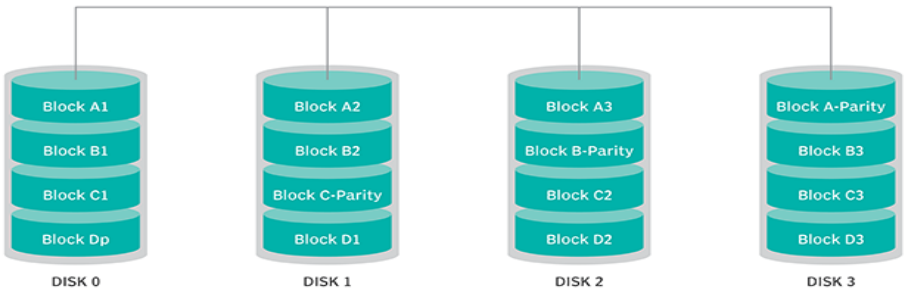
RAID 4



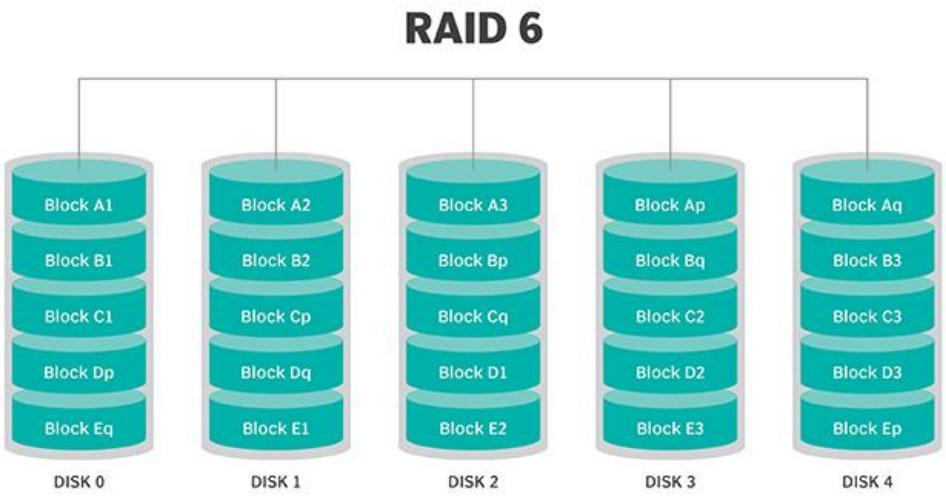
RAID 5. This level is based on parity block-level striping. The parity information is striped across each drive, enabling the array to function, even if one drive were to fail. The array's architecture enables read and write operations to span multiple drives. This results in performance better than that of a single drive, but not as high as a RAID 0 array. RAID 5 requires at least three disks, but it is often recommended to use at least five disks for performance reasons.

RAID 5 arrays are generally considered to be a poor choice for use on write-intensive systems because of the performance impact associated with writing parity data. When a disk fails, it can take a long time to rebuild a RAID 5 array.

RAID 5



RAID 6. This technique is similar to RAID 5, but it includes a second parity scheme distributed across the drives in the array. The use of additional parity enables the array to continue functioning, even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 arrays often have slower write performance than RAID 5 arrays.



Advantages of RAID include the following:

- Improved cost-effectiveness because lower-priced disks are used in large numbers.
- Using multiple hard drives enables RAID to improve the performance of a single hard drive.
- Increased computer speed and reliability after a crash, depending on the configuration.
- Reads and writes can be performed faster than with a single drive with RAID 0. This is because a file system is split up and distributed across drives that work together on the same file.
- There is increased availability and resiliency with RAID 5. With mirroring, two drives can contain the same data, ensuring one will continue to work if the other fails.

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. UNIX, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

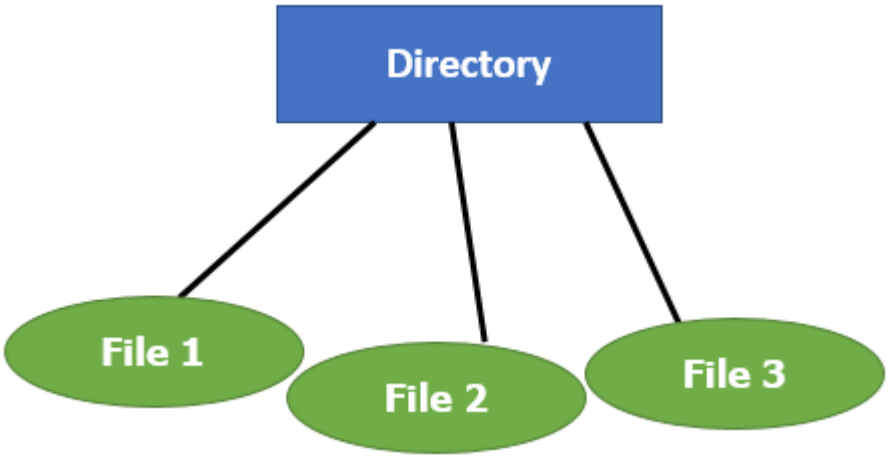
File Directories

A single directory may or may not contain multiple files. It can also have sub-directories inside the main directory. Information about files is maintained by Directories. In Windows OS, it is called folders.

Following is the information which is maintained in a directory:

- **Name** The name which is displayed to the user.
- **Type**: Type of the directory.
- **Position**: Current next-read/write pointers.
- **Location**: Location on the device where the file header is stored.

- **Size**: Number of bytes, block, and words in the file.
- **Protection**: Access control on read/write/execute/delete.
- **Usage**: Time of creation, access, modification



Single Level Directory

File Sharing:-

File sharing is the act of sharing one or more computer files over a network with someone in the same house, a team member at work, or a friend in another country. You can also use file sharing to access your files from anywhere.

You can share files over a local network in an office or at home, or you can share files over the internet.

File sharing isn't the same as network sharing. To share a file is to send it to another device such as a computer or phone. Network sharing shares a network connection so that nearby devices can access network resources.

Types of File Sharing: -

There are two ways to share files over a network: directly between two computers or between a computer and a server.

1. When a file is shared between a computer and a server, the computer uploads the file to a storage area on the server where you can share it with others. People that want access to the file download it from that server.

2. When a file is shared between two computers over a network, the file is sent directly to the other person. This is often called peer-to-peer (P2P) file sharing and works by communicating directly with the other person's device, with no servers involved.

How to Share Individual Files and Folders

There are several ways to share files over a network, and while some methods are easier than others, all are similar.

a) With a File Transfer Tool

An on-demand file transfer program is a quick way to share files over any network, whether it's the internet with someone in another country or a local network between two computers in an office. You usually don't need a user account to use these file-sharing tools, and the directions are often straightforward.

FTP is one example that involves setting up an FTP server on the computer that has the files to share. Anyone that wants the files uses an FTP client to communicate with the server to download the files.

There are also peer-to-peer (P2P) tools for file sharing, such as ShareDrop. With this file-sharing tool, you're provided a special URL that grants access to download your files, and you choose what to share from your computer.

Takeafile and JustBeamIt are similar services that share files over the internet between two computers. You can also use torrents.

b) From a Cloud Storage Service

Cloud storage services store the file on a server. Others can download the file if they have the link to the file.

Cloud storage services offer high speeds to download files, speeds a normal user might not be able to support (your upload bandwidth caps P2P file-sharing speeds).

Use an online file storage website to share the same file with multiple people or to share a file with someone in the future and not re-upload it. The file is stored in your cloud file storage account for as long as you want.

Temporary cloud storage services keep files for a few hours or days. WeTransfer and WeSendit.com are examples.

c) Over a Messaging Application

Another popular way to share files between computers is with a messaging app. Email and texting are two methods, but there are others. Look for a file selector where you can choose which files to share.

Some apps that support file sharing include Facebook Messenger, WhatsApp, Slack, and Skype. Many are web-based messaging services, meaning you can share files without installing anything.

d) Use Your Operating System

Another way to share files over a network is with your computer's operating system. There are often tools built-in to the OS that can do it, although this method is usually only beneficial for sharing files over a local network.

A mapped drive is an uncomplicated way to set up a file-sharing network in Windows. A mapped drive allows you to download files directly from another computer that sets up a network share. You can also set up file sharing on a Mac and other operating systems.

e) Share a Whole Computer

When a computer is shared over a network, every file and folder on the computer is also shared. This isn't something that you can do with a P2P file transfer service, cloud storage service, or messaging app. Those methods have you pick specific files to share, so a different solution is needed to share the entire computer.

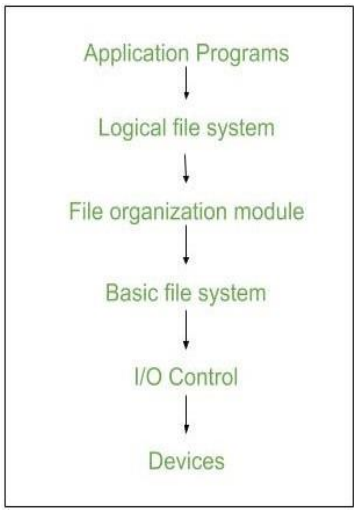
For example, you can share files in Windows by **enabling the admin share** so that anyone on the local network that knows the admin password can access every file on the computer.

Another way to share access to all the computer files is with a **remote access program**. These tools work differently than a typical file sharing utility because instead of sharing the actual files, only the computer screen is shared.

File System Implementation: -

A file is a collection of related information. The file system resides on secondary storage and provides efficient and convenient access to the disk by allowing data to be stored, located, and retrieved.

File system organized in many layers :



- **I/O Control level: -** Device driver's acts as interface between devices and OS, they help to transfer data between disk and main memory. It takes block number an input and as output it gives low level hardware specific instruction.
- **Basic file system: -** It Issues general commands to device driver to read and write physical blocks on disk. It manages the memory buffers and caches. A block in buffer can hold the contents of the disk block and cache stores frequently used file system metadata.
- **File organization Module: -** It has information about files, location of files and their logical and physical blocks. Physical blocks do not match with logical numbers of logical block numbered from 0 to N. It also has a free space which tracks unallocated blocks.
- **Logical file system: -** It manages metadata information about a file i.e. includes all details about a file except the actual contents of file. It also maintains via file control blocks. File control block (FCB) has information about a file – owner, size, permissions, and location of file contents.

Advantages: -

1. Duplication of code is minimized.
2. Each file system can have its own logical file system.

Disadvantages: -

1. If we access many files at same time then it results in low performance.

We can **implement** file system by using two types' data structures:

1. On-disk Structures: - Generally they contain information about total number of disk blocks, free disk blocks, location of them and etc. Below given are different on-disk structures:

a) Boot Control Block: - It is usually the first block of volume and it contains information needed to boot an operating system. In UNIX it is called boot block and in NTFS it is called as partition boot sector.

b) Volume Control Block: - It has information about a particular partition ex:- free block count, block size and block pointers etc. In UNIX it is called super block and in NTFS it is stored in master file table.

c) Directory Structure: - They store file names and associated inode numbers. In UNIX, includes file names and associated file names and in NTFS, it is stored in master file table.

d) Per-File FCB: - It contains details about files and it has a unique identifier number to allow association with directory entry. In NTFS it is stored in master file table.

2. In-Memory Structure: - They are maintained in main-memory and these are helpful for file system management for caching. Several in-memory structures given below:

a) Mount Table: - It contains information about each mounted volume.

b) Directory-Structure cache: - This cache holds the directory information of recently accessed directories.

c) System wide open-file table: - It contains the copy of FCB of each open file.

d) Per-process open-file table: - It contains information opened by that particular process and it maps with appropriate system wide open-file.

How does the file system handle security?

- The file system is crucial to data integrity.
- Main method of protection is through access control
- Accessing file system operations (ex. modifying or deleting a file) are controlled through access control lists or capabilities
- Capabilities are more secure so they tend to be used by operating systems on file systems like NTFS or ext3.
- Secondary method of protection is through the use of backup and recovery systems

Attacks on the file system

There are three most common methods

- Race Condition Attacks
- Using ADS to hide files
- Directory traversal

Attacks on the file system

Race Condition Attacks

- Occurs when a process performs a sequence of operations on a file, under the assumption that they are executed atomically.
- Can be used by the attacker to change the characteristics of that file between two successive operations on it resulting in the victim process to operate on the modified file.

Attacks on the file system

Using ADS to hide files

- Alternate Data Streams(ADS) allows multiple data streams to be attached to a single file.
- A file can be hidden behind a file as an attached stream that could be hundreds of megabytes in size, however a directory listing will only display the file's normal size.

Attacks on the file system

Directory traversal

- An exploit caused by lack of insufficient security validation of user supplied input file names
- For example the attacker would pass this as input. ../../../../../../../../../etc/password to retrieve the password file from the server.

How does the file system ensure data integrity?

There are various methods of protecting the files on a file system.

- Access Controls
- Encryption
- RAID
- Recovery when data is corrupted

How does the file system ensure data integrity?

Access Control

- Access Control plays a huge part in file system security
- The system should only allow access to files that the user is permitted to access
- Almost all major file systems support ACL's or capabilities in order to prevent malicious activity on the file system
- Depending on the users rights they can be allowed to read, write and/or execute and object. In some file systems schemes only certain users are allowed to alter the ACL on a file or see if a file even exists.
- Ultimately the less the user has access to the less that can go wrong and the integrity of the disk can be more guaranteed.

How does the file system ensure data integrity?

General File System Encryption

- Encryption is also a method used by file systems to secure data, NTFS for example offers file encryption using DESX
- Two method of disk encryption
  - Full Disk Encryption
  - File System Encryption
- File system encryption has a few advantages over full disk encryption for example
  - File based key management
  - Individual management of encrypted files
  - Access control can be further strengthened through the use of public key cryptography
  - Keys are only held in memory while the file is being used

How does the file system ensure data integrity?

General File System Encryption

Encrypting File System(EFS)

- Provides security beyond user authentication and access control lists. For example when the attacker has physical access to the computer.
- EFS uses public key cryptography however it is susceptible to brute-force attacks against the user account passwords.

How does the file system ensure data integrity?

General File System Encryption

EFS Encryption

- EFS works by encrypting a file with a bulk symmetric key, aka File Encryption Key or FEK.
- The FEK is encrypted with a public key that is associated with the user that encrypted the file.

How does the file system ensure data integrity?

General File System Encryption

EFS Decryption

- The EFS uses the private key that matches the EFS digital certificate (that was used to encrypt the file) to decrypt the symmetric key.
- The resulting symmetric key is then used to decrypt the file.

How does the file system ensure data integrity?

RAID

- RAID stands for Redundant Array of Independent Disks
- Offers drawbacks and advantages over a single disk, each with different applications
- Types of RAID
  - RAID 0 "Striping set without parity"
  - RAID 1 "Mirrored set without parity"
  - RAID 3 "Striped set with byte level parity"
  - RAID 4 "Striped set with block level parity"
  - RAID 5 "Striped set with distributed parity"
  - RAID 6 "Striped set with dual distributed parity"

How does the file system ensure data integrity?

Recovery when data is corrupted

What happens when something is corrupted?

- Checksum codes
- Reed Soloman Codes (cd's to fix errors caused by scratches)
- Given the right type of RAID, the system can recover easily.
  - Parity Schemes
  - Protection against individual drive failure