Chapter Four
Device Management

Introduction

Device management is the process of managing the implementation, operation and maintenance of a physical and/or virtual device. It is a broad term that includes various administrative tools and processes for the maintenance and upkeep of a computing, network, mobile and/or virtual device.
The key features of device management in the operating systems are:

- The operating system interacts with the device controllers through device drivers while allocating the device to the various processes running on the system.
- Device drivers are system software programs that connect processes and device controllers.
- The device management in the operating system is responsible for implementing the APIs.
- The device controller, which is used in device management operations, consists primarily of three registers: command, status, and data.

Device management generally performs the following:

- Installing device and component-level drivers and related software
- Configuring a device so it performs as expected using the bundled operating system, business/workflow software and/or with other hardware devices.
- Implementing security measures and processes.

Devices usually refer to physical/hardware devices such as computers, laptops, servers, mobile phones and more. They could also be virtual, however, such as virtual machines or virtual switches. In Windows, device management is also an administrative module that is used for managing or configuring the physical devices, ports and interfaces of a computer or server.
The I/O system of an OS works by taking I/O request from an application software and sending it to the physical device, which could be an input or output device then it takes whatever response comes back from the device and sends it to the application.

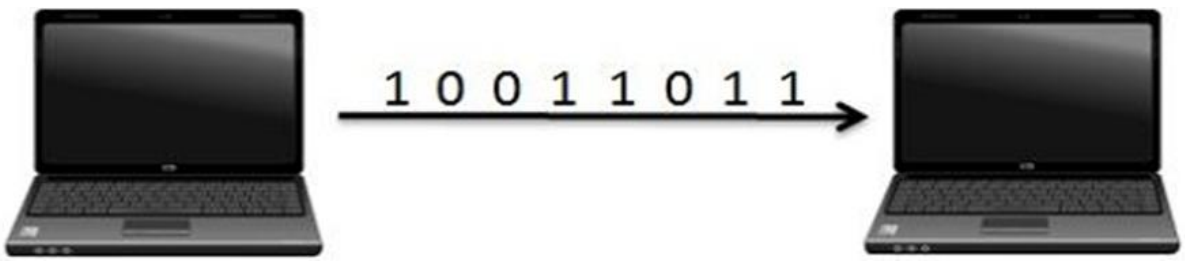
Components of I/O Hardware

I/O Device: I/O devices such as storage, communications, user-interface, and others communicate with the computer via signals sent over wires or through the air. Devices connect with the computer via ports, e.g. a serial or parallel port. A common set of wires connecting multiple devices is termed a bus.
The fundamentals of I/O devices are classified into three types:
Boot Device: It organizes data into fixed-size blocks, each with its own unique address.
Character Device: It sends or receives a continuous stream of characters, none of which can be addressed individually. For example, keyboards, printers, and so on.
Network Device: It is used for data packet transmission.

Characteristics of serial and parallel devices

Serial Transmission

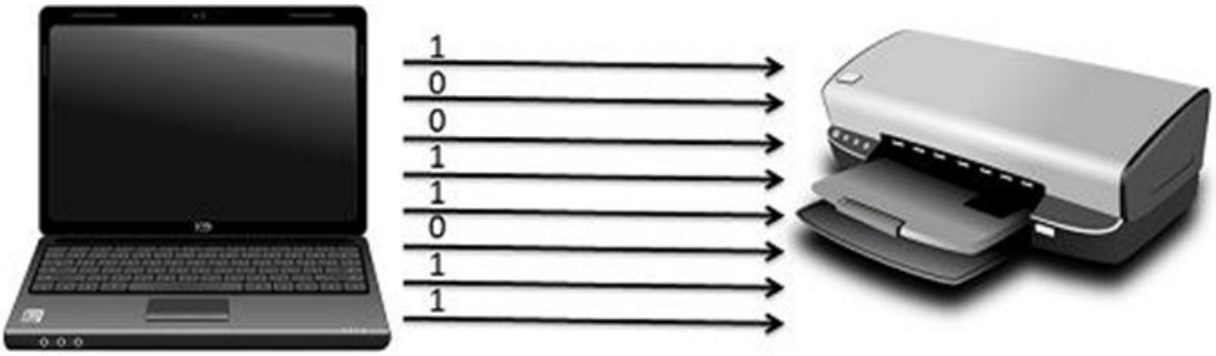
In Serial Transmission, data is sent bit by bit from one computer to another in bi-direction where each bit has its clock pulse rate. Eight bits are transferred at a time having a start and stop bit (usually known as a Parity bit), i.e. 0 and 1 respectively. For transmitting data to a longer distance, serial data cables are used. However, the data transferred in the serial transmission is in proper order. It consists of a D-shaped 9 pin cable that connects the data in series.



Serial Transmission has two subclasses synchronous and asynchronous. In asynchronous transmission, an extra bit is added to each byte so that the receiver is alert about the arrival of new data. Usually, 0 is a start bit, and 1 is the stop bit. In synchronous transmission, no extra bit is added rather the data transferred in the form of frames which contains multiple bytes. The serial transmission system would not be able to work without installing hardware at the sending and receiving. The hardware residing in the sending and receiving end is capable of converting the data from the parallel mode (used in the device) to the serial mode (used in the wires).

Parallel Transmission

In Parallel Transmission, various bits are sent together simultaneously with a single clock pulse. It is a fast way to transmit as it uses many input/output lines for transferring the data. Furthermore, it is advantageous because it conforms to the underlying hardware also, as the electronic devices like computer and communication hardware uses parallel circuitry internally. This is a reason the parallel interface complements the internal hardware well.



The installation and troubleshooting are easier in parallel transmission system due to its placement in a single physical cable. Parallel Transmission uses a 25 pin port having 17 signal lines and 8 ground lines. The 17 signal lines are further divided as

- 4 lines that initiate handshaking,
- Status lines used to communicate and notify errors and
- 8 to transfer data.

Despite the speed of the data, the parallel transmission has a limitation called skew where bits could travel in quite different speeds over the wires.

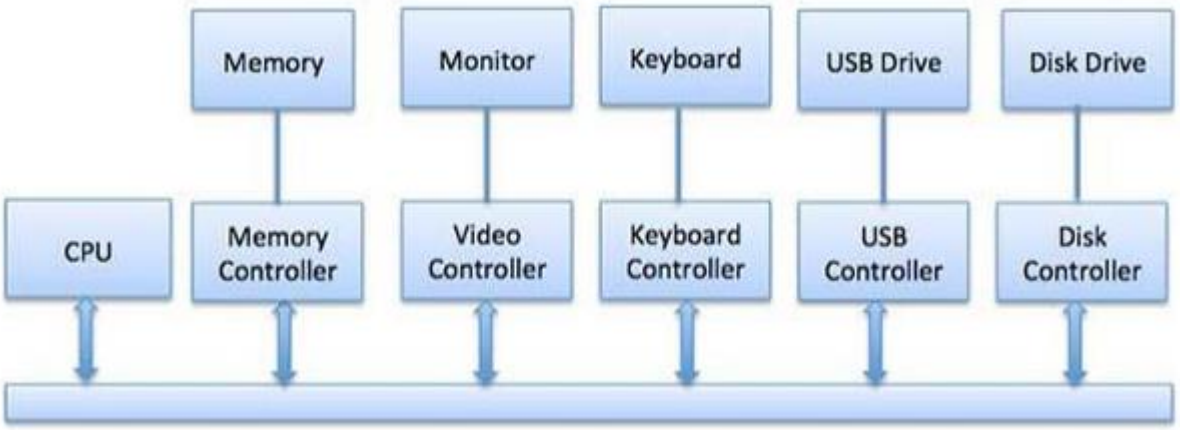
For transferring data between computers, laptops, two methods are used, namely, Serial Transmission and Parallel Transmission. There are some similarities and dissimilarities between them. One of the primary differences is that; in Serial Transmission, data is sent bit by bit whereas, in Parallel Transmission a byte (8 bits) or character is sent at a time. The similarity is that both are used to connect and communicate with peripheral devices. Furthermore, the parallel transmission is time-sensitive, whereas serial transmission is not time-sensitive. Other differences are discussed below.

Both Serial and Parallel Transmission have their advantages and disadvantages, respectively. Parallel Transmission is used for a limited distance, provides higher speed. On the other hand, Serial Transmission is reliable for transferring data to longer distance. Hence, we conclude that both serial and parallel are individually essential for transferring data.

Device Driver: Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

Device Controller: The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary. Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.



Types of devices

There are three types of Operating system peripheral devices: dedicated, shared, and virtual. These are as follows:

1. Dedicated Device
In device management, some devices are allocated or assigned to only one task at a time until that job releases them. Devices such as plotters, printers, tape drivers, and other similar devices necessitate such an allocation mechanism because it will be inconvenient if multiple people share them simultaneously. The disadvantage of such devices is the inefficiency caused by allocating the device to a single user for the whole duration of task execution, even if the device is not used 100% of the time.

2. Shared Devices
These devices could be assigned to a variety of processes. By interleaving their requests, disk-DASD could be shared by multiple processes simultaneously. The Device Manager carefully controls the interleaving, and pre-determined policies must resolve all difficulties.

3. Virtual Devices
Virtual devices are a hybrid of the two devices, and they are dedicated devices that have been transformed into shared devices. For example, a printer can be transformed into a shareable device by using a spooling program that redirects all print requests to a disk. A print job is not sent directly to the printer; however, it is routed to the disk until it is fully prepared with all of the required sequences and formatting, at which point it is transmitted to the printers. The approach can transform a single printer into numerous virtual printers, improving performance and ease of use.

Communication to I/O Devices

The CPU must have a way to pass information to and from an I/O device. There are different approaches available to communicate with the CPU and Device.

Polling

In this case, a CPU continuously checks the device status for data exchange. The advantage is that it is simple, but the disadvantage is busy-waiting. When an I/O operation is required in this case, the computer does nothing but check the status of the I/O device until it is ready, at which point the device is accessed. To put it another way, the computer waits until the device is ready.

Interrupt-Driven I/O

It is the responsibility of the device controller to notify the corresponding driver about the device's availability. The benefits include more efficient use of CPU cycles, and the drawback is slower data copying and movement for character devices—one interrupt per keyboard input.

A device controller converts a serial bit stream to a block of bytes. If necessary, it also performs error correction. It has two main components: device registers to communicate with the CPU and a data buffer that an operating system can read or write.

Direct memory access (DMA)

Many devices can temporarily take control of the bus and perform data transfers to (and from) main memory or other devices. Because the device is doing the work without the help of the CPU, this type of data transfer is known as direct memory access (DMA). DMA transfers can be performed between two devices, between a device and memory, or between memory and memory. This chapter explains transfers between a device and memory only.

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations. The process is managed by a chip known as a DMA controller (DMAC). A defined portion of memory is used to send data directly from a peripheral to the motherboard without involving the microprocessor, so that the process does not interfere with overall computer operation.

In older computers, four DMA channels were numbered 0, 1, 2 and 3. When the 16-bit industry standard architecture (ISA) expansion bus was introduced, channels 5, 6 and 7 were added.

ISA was a computer bus standard for IBM-compatible computers, allowing a device to initiate transactions (bus mastering) at a quicker speed. The ISA DMA controller has 8 DMA channels, each one of which associated with a 16-bit address and count registers. ISA has since been replaced by accelerated graphics port (AGP) and peripheral component interconnect (PCI) expansion cards, which are much faster. Each DMA transfers approximately 2 MB of data per second. A computer's system resource tools are used for communication between hardware and software.

The four types of system resources are:

- I/O addresses.
- Memory addresses.
- Interrupt request numbers (IRQ).
- Direct memory access (DMA) channels.

DMA channels are used to communicate data between the peripheral device and the system memory. All four system resources rely on certain lines on a bus. Some lines on the bus are used for IRQs, some for addresses (the I/O addresses and the memory address) and some for DMA channels.

A DMA channel enables a device to transfer data without exposing the CPU to a work overload. Without the DMA channels, the CPU copies every piece of data using a peripheral bus from the I/O device. Using a peripheral bus occupies the CPU during the read/write process and does not allow other work to be performed until the operation is completed.

With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. The data block can be transferred to and from memory by the DMAC in three ways.

In burst mode, the system bus is released only after the data transfer is completed. In cycle stealing mode, during the transfer of data between the DMA channel and I/O device, the system bus is relinquished for a few clock cycles so that the CPU can perform other tasks. When the data transfer

is complete, the CPU receives an interrupt request from the DMA controller. In transparent mode, the DMAC can take charge of the system bus only when it is not required by the processor.
However, using a DMA controller might cause cache coherency problems. The data stored in RAM accessed by the DMA controller may not be updated with the correct cache data if the CPU is using external memory. Solutions include flushing cache lines before starting outgoing DMA transfers, or performing cache invalidation on incoming DMA transfers when external writes are signaled to the cache controller.

Buffering strategies

Buffering

In computer system when the speed in which data is received and the speed in which data is processed are different, then there we use the buffer. Buffer is a memory space which stores the input data and passes it on to the system according to this speed in this way there is no need to hold the input device until it is processed. Simply the data will be stored in buffer and the used by the system. The buffer can be of any type, hardware or software, but in general software buffer are used widely.

Example – In printer's spoolers, we can pass a large no of pages to print as input, but the processing/printing is slow. Here buffering is used.

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.

I/O buffering and its Various Techniques

A buffer is a memory area that stores data being transferred between two devices or between a device and an application.

Uses of I/O Buffering

- Buffering is done to deal effectively with a speed mismatch between the producer and consumer of the data stream.
- A buffer is produced in main memory to heap up the bytes received from modem.
- After receiving the data in the buffer, the data get transferred to disk from buffer in a single operation.
- This process of data transfer is not instantaneous; therefore, the modem needs another buffer in order to store additional incoming data.
- When the first buffer got filled, then it is requested to transfer the data to disk.
- The modem then starts filling the additional incoming data in the second buffer while the data in the first buffer getting transferred to disk.
- When both the buffers completed their tasks, then the modem switches back to the first buffer while the data from the second buffer get transferred to the disk.
- The use of two buffers disintegrates the producer and the consumer of the data, thus minimizes the time requirements between them.
- Buffering also provides variations for devices that have different data transfer sizes.

Types of various I/O buffering techniques

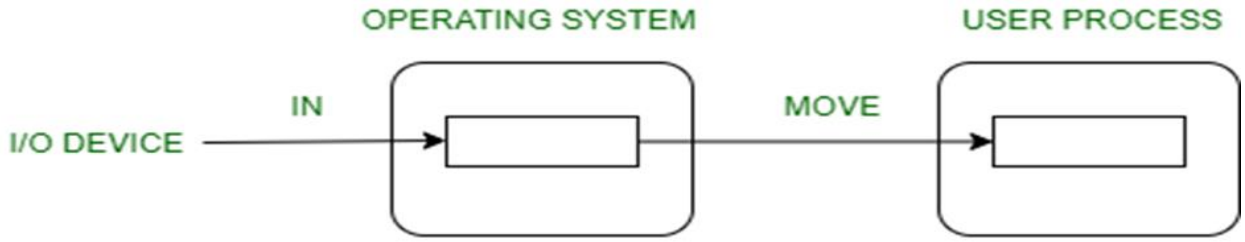
1. Single buffer: buffer is provided by the operating system to the system portion of the main memory.

Block oriented device

- System buffer takes the input.
- After taking the input, the block gets transferred to the user space by the process and then the process requests for another block.
- Two blocks work simultaneously, when one block of data is processed by the user process, the next block is being read in.
- OS can swap the processes.
- OS can record the data of system buffer to user processes.

Stream oriented device

- Line- at a time operation is used for scroll made terminals. User inputs one line at a time, with a carriage return signaling at the end of a line.
- Byte-at a time operation is used on forms mode, terminals when each keystroke is significant.



SINGLE I/O BUFFERING

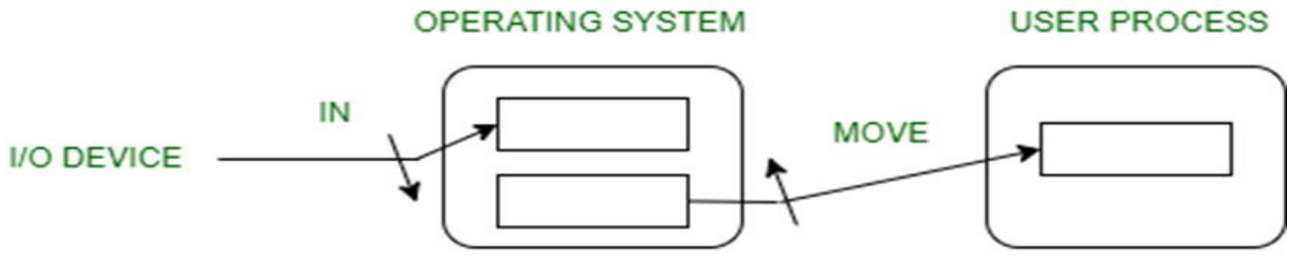
2. Double buffer: two schemes or two buffers are used in the place of one. In this buffering, the producer produces one buffer while the consumer consumes another buffer simultaneously. So, the producer not needs to wait for filling the buffer. Double buffering is also known as buffer swapping.

Block oriented

- There are two buffers in the system.
- One buffer is used by the driver or controller to store data while waiting for it to be taken by higher level of the hierarchy.
- Other buffer is used to store data from the lower-level module.
- Double buffering is also known as buffer swapping.
- A major disadvantage of double buffering is that the complexity of the process get increased.
- If the process performs rapid bursts of I/O, then using double buffering may be deficient.

Stream oriented

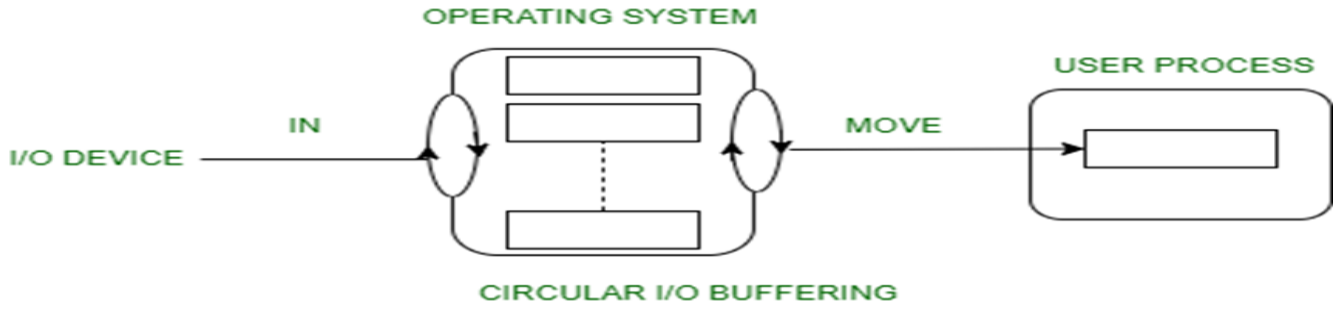
- Line- at a time I/O, the user process need not be suspended for input or output, unless process runs ahead of the double buffer.
- Byte- at a time operation, double buffer offers no advantage over a single buffer of twice the length.



DOUBLE I/O BUFFERING

3. Circular buffer: When more than two buffers are used, the buffers' collection is called a **circular buffer**. Each buffer is being one unit in the circular buffer. The data transfer rate will increase using the circular buffer rather than the double buffering.

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer.
- In this, the data do not directly pass from the producer to the consumer because the data would change due to overwriting of buffers before they had been consumed.
- The producer can only fill up to buffer i-1 while data in buffer i is waiting to be consumed.



Recovery from failures

Recovery from Failure is a phrase used to describe a need in aviation to continue real-time operations to a safe conclusion despite a critical part of a system (technical, procedural, or human) failing, sometimes at the most crucial time.

To maintain high availability of its stored data, ScaleOut StateServer creates up to two replicas of every stored data object (up to three copies total as determined by the max_replicas parameter). These replicas are apportioned to other hosts so as to maintain the overall load-balance. The number of replicas per object depends on the number of available hosts:

# hosts	**# replicas per object**
1	0
2	1
>2	1 or 2, as specified

All updates to an object are transactionally applied to its replicas using a patented, quorum-based updating algorithm. When a host fails or suffers a network outage, ScaleOut StateServer uses the replicas on other hosts to preserve stored data and to provide continued access to the affected objects. If a host suffers only a brief network outage, it resumes management of its objects once the outage has been corrected. Otherwise, in a process called self-healing, one of the replica's hosts takes over management of each affected object, and an additional replica is created on another surviving host. (The failed host leaves the store.) Note that a host's failure only affects the objects managed by that host; other objects usually remain continuously available during the recovery process.

Detecting Failures

Scale Out State Server uses periodic heartbeat packet exchanges between pairs of hosts to detect a network outage or host failure. To minimize overall networking overhead, ScaleOut StateServer uses a highly efficient heartbeat algorithm that limits the number of heartbeat exchanges per second so that overhead grows slowly with the number of hosts. Under normal conditions, the maximum network bandwidth consumed by a host to send heartbeat packets is less than 1K bytes/second.

Based on extensive evaluation of IP load balancing on server farms, ScaleOut StateServer uses a default heartbeat exchange rate of one heartbeat packet per 300 milliseconds. A host reports a possible outage if twenty consecutive heartbeat packets are not received from a remote host. This indicates that either one of the hosts has a network outage or the remote host has failed. Once an outage has been detected, the host begins a recovery process to isolate the problem and recover from a failure.

The Recovery Process

After a heartbeat failure occurs, all hosts engage in a recovery protocol to establish the currently responding membership. Hosts that do not respond because of server failures or network outages are dropped from the membership of the distributed store. This process takes approximately five to ten seconds after the heartbeat failure occurs. After the new membership is established, the hosts perform a self-healing process in which:

- the hosts that are determined to have failed are removed from the membership,

- lost copies of stored objects are replicated on other hosts to restore full redundancy (i.e., one or two replicas per object for farms with three or more hosts),
- heartbeat exchanges are initiated between appropriate surviving hosts to replace the terminated exchanges, and
- all hosts rebalance the storage load across the new membership. The self-healing and dynamic rebalancing process may require a minute or more to replicate the affected objects on new hosts, and this will temporarily slow the rate at which access requests to the affected objects can be handled.

Scale Out State Server heals its distributed store by creating new replicas for stored objects that were lost due to a server failure or network outage. This restores the full redundancy of the distributed store. If two or more servers are removed from the membership during the recovery process, some objects may be permanently lost. In this case, the store will return object_not_found exceptions for access requests to missing objects. To maintain service whenever possible, ScaleOut StateServer attempts to self-heal and restore service even if no copies of an object still exist.

A host which suffers a network outage and cannot reach the network displays the isolated status and assigns all remote hosts an unknown status. The host suspends all handling of access requests until the network outage is corrected. This helps to maintain the consistency of stored data; see Split-Brain Failures below.

A host becomes isolated when its connection to the network switch is unplugged, as shown in the following diagram. This will cause the host to suspend its handling of access requests, and other hosts may remove it from the membership. It may not be possible for the ScaleOut service to detect this condition under all situations, in which case the host will form an independent membership.

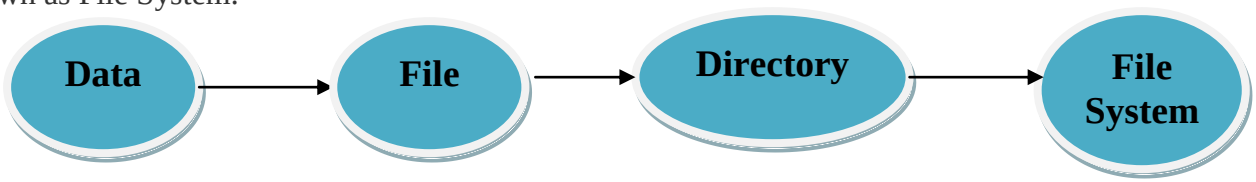
Once an active host can again reach the network, it determines if it is in the current membership. Remote hosts may have formed a new membership that excludes the host so that they can recover from the network outage. In this case, the host automatically restarts the StateServer service so that the host can rejoin the distributed store. This behavior ensures that an isolated host does not interfere with other hosts once a new membership has been formed. To enable recovery from switch failures, ScaleOut StateServer does not automatically restart an isolated hostafter it waits for a fixed recovery interval. Instead, it waits indefinitely for the network outage to be corrected and then determines whether to restart the StateServer service.

Chapter Four
File and Secondary Storage Management

What is a File ?

A file can be defined as a data structure which stores the sequence of records. Files are stored in a file system, which may exist on a disk or in the main memory. Files can be simple (plain text) or complex (specially-formatted).

The collection of files is known as Directory. The collection of directories at the different levels, is known as File System.



Attributes of the File

Name: Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
Identifier: Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension **.txt;** a video file can have the extension **.mp4.**
Type: In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.
Location: In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.
Size: The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
Protection: The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.
Time and Date: Every file carries a time stamp which contains the time and date on which the file is last modified.

Operations on the File

The various operations which can be implemented on a file such as read, write, open and close etc. are called file operations. These operations are performed by the user by using the commands provided by the operating system. Some common operations are as follows:

Create operation:
This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.

Open operation:
This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

Write operation:
This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.

Read operation:
This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

Re-position or Seek operation:
The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

Delete operation:
Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.

Truncate operation:
Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.

Close operation:
When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.
Append operation: This operation adds data to the end of the file.

Rename operation: This operation is used to change name (give new name) for the existing file.

Secondary Storage

Secondary storage is non-volatile, persistent computer memory that is not directly accessed by a computer or processor, and it processes data at a slower rate. It allows a user to store data that can be accessed, transmitted, and used by applications and services in real time. Secondary storage is less crucial than primary memory because it is essentially more storage for more data. The secondary storage is also termed **auxiliary memory or external memory.**
One of the reasons we have secondary storage is, it is essential to have another form of memory that has a high storage capacity and from which data and programs are not lost when the computer turns off. It can be:

Fixed Storage

A fixed storage device is an internal disc or media device in our computer system that is used to store data. Usually, the data of the computer system is saved in a fixed storage device incorporated into the system. Fixed storage is often referred to as hard drives or fixed disc drives. In general, we don't remove fixed devices, but sometimes we remove them in case of maintenance or replacement or any kind of repairing services required.

Removable Storage

Unlike Fixed Storage, removable media is an external disc in our computer system in the secondary memory. Disk drives or external drives are popular words for removable storage. It's a storage device that may be installed or withdrawn from a computer based on our needs. We can effortlessly uninstall them from the computer system while it is still functioning. We can simply transfer data from one computer to another using removable storage device since they are portable.

Classification of Secondary Storage Devices

The following image shows the classification of commonly used secondary storage devices.

Sequential Access Storage Device

It is a class of data storage devices that read stored data in a sequence. This is in contrast to random access memory (RAM), where data can access in any order, and magnetic tape is the common sequential access storage device.

- **Magnetic tape:** It is a medium for magnetic recording, made of a thin, magnetizable coating on a long, narrow strip of plastic film. Devices that record and play audio and video using magnetic tape are tape recorders and videotape recorders. A device that stores computer data on magnetic tape is known as a ***tape drive***.
  It was a key technology in early computer development, allowing unparalleled amounts of data to be mechanically created, stored for long periods, and rapidly accessed.

Direct Access Storage Devices

A direct-access storage device (DASD) is another name for secondary storage devices that store data in discrete locations with a unique address, such as hard disk drives, optical drives and most magnetic storage devices.

Magnetic disks: A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data. It is covered with a magnetic coating and stores data in the form of tracks, spots and sectors. Hard disks, zip disks and floppy disks are common examples of magnetic disks.

i. **Floppy Disk:** A floppy disk is a flexible disk with a magnetic coating on it, and it is packaged inside a protective plastic envelope. These are among the oldest portable storage devices that could store up to 1.44 MB of data, but now they are not used due to very little memory storage.

ii. **Hard Disk Drive (HDD):** Hard disk drive comprises a series of circular disks called ***platters*** arranged one over the other almost ½ inches apart around a ***spindle***. Disks are made of non-magnetic material like aluminium alloy and coated with 10-20 nm magnetic material. The standard diameter of these disks is 14 inches, and they rotate with speeds varying from 4200 rpm (rotations per minute) for personal computers to 15000 rpm for servers.

    Data is stored by magnetizing or demagnetizing the magnetic coating. A magnetic reader arm is used to read data from and write data to the disks. A typical modern HDD has a capacity in terabytes (TB).

Optical Disk: An optical disk is any computer disk that uses optical storage techniques and technology to read and write data. It is a computer storage disk that stores data digitally and uses laser beams to read and write data.

i. **CD Drive:** CD stands for Compact Disk. CDs are circular disks that use optical rays, usually lasers, to read and write data. They are very cheap as you can get 700 MB of storage space for less than a dollar. CDs are inserted in CD drives built into the CPU cabinet. They

are portable as you can eject the drive, remove the CD and carry it with you. There are three types of CDs:

- **CD-ROM (Compact Disk - Read Only Memory):** The manufacturer recorded the data on these CDs. Proprietary Software, audio or video are released on CD-ROMs.
- **CD-R (Compact Disk - Recordable):** The user can write data once on the CD-R. It cannot be deleted or modified later.
- **CD-RW (Compact Disk - Rewritable):** Data can repeatedly be written and deleted on these optical disks.

ii. **DVD Drive:** DVD stands for digital video display. DVD is an optical device that can store 15 times the data held by CDs. They are usually used to store rich multimedia files that need high storage capacity. DVDs also come in three varieties - read-only, recordable and rewritable.

iii. **Blu Ray Disk:** Blu Ray Disk (BD) is an optical storage media that stores high definition (HD) video and other multimedia files. BD uses a shorter wavelength laser than CD/DVD, enabling the writing arm to focus more tightly on the disk and pack in more data. BDs can store up to 128 GB of data.

Memory Storage Devices: A memory device contains trillions of interconnected memory cells that store data. When switched on or off, these cells hold millions of transistors representing 1s and 0s in binary code, allowing a computer to read and write information. It includes USB drives, flash memory devices, SD and memory cards, which you'll recognize as the storage medium used in digital cameras.

i. **Flash Drive:** A flash drive is a small, ultra-portable storage device. USB flash drives were essential for easily moving files from one device to another. Flash drives connect to computers and other devices via a built-in USB Type-A or USB-C plug, making one a USB device and cable combination.
Flash drives are often referred to as pen drives, thumb drives, or jump drives. The terms *USB drive* and *solid-state drive (SSD)* are also sometimes used, but most of the time, those refer to larger, not-so-mobile USB-based storage devices like external hard drives. These days, a USB flash drive can hold up to 2 TB of storage. They're more expensive per gigabyte than an external hard drive, but they have prevailed as a simple, convenient solution for storing and transferring smaller files.
Pen drive has the following advantages in computer organization, such as:

- **Transfer Files:** A pen drive is a device plugged into a USB port of the system that is used to transfer files, documents, and photos to a PC and vice versa.
- **Portability:** The lightweight nature and smaller size of a pen drive make it possible to carry it from place to place, making data transportation an easier task.
- **Backup Storage:** Most of the pen drives now come with the feature of having password encryption, important information related to family, medical records, and photos can be stored on them as a backup.

- **Transport Data:** Professionals or Students can now easily transport large data files and video, audio lectures on a pen drive and access them from anywhere. Independent PC technicians can store work-related utility tools, various programs, and files on a high-speed 64 GB pen drive and move from one site to another.

ii. **Memory card:** A memory card or memory cartridge is an electronic data storage device used for storing digital information, typically using flash memory. These are commonly used in portable electronic devices, such as digital cameras, mobile phones, laptop computers, tablets, PDAs, portable media players, video game consoles, synthesizers, electronic keyboards and digital pianos, and allow adding memory to such devices without compromising ergonomy, as the card is usually contained within the device rather than protruding like USB flash drives.

Disk Management: is a feature of the operating system that allows you to create, delete, and format disc partitions, among other things. Users can inspect, create, remove, and reduce partitions connected with disc drives using Disk Management. The operating system's disc management includes:

1. Disk Formatting

A new magnetic disk is mainly a blank slate. It is platters of the magnetic recording material. Before a disk may hold data, it must be partitioned into sectors that may be read and written by the disk controller. It is known as **physical formatting** and **low-level formatting.**

Low-level formatting creates a unique data structure for every sector on the drive. A data structure for a sector is made up of a header, a data region, and a trailer. The disk controller uses the header and trailer to store information like an error-correcting code (ECC) and a sector number.

The OS must require recording its own data structures on the disk drive to utilize it as a storage medium for files. It accomplishes this in two phases. The initial step is to divide the disk drive into one or more cylinder groups. The OS may treat every partition as it were a separate disk. For example, one partition could contain a copy of the OS executable code, while another could contain user files. The second stage after partitioning is **logical formatting.** The operating store stores the initial file system data structure on the disk drive in this second stage.

2. Boot Block

When a system is turned on or restarted, it must execute an initial program. The start program of the system is called the bootstrap program. It starts the OS after initializing all components of the system. The bootstrap program works by looking for the OS kernel on disk, loading it into memory, and jumping to an initial address to start the OS execution.

The bootstrap is usually kept in read-only memory on most computer systems. It is useful since read-only memory does not require initialization and is at a fixed location where the CPU may begin executing whether powered on or reset. Furthermore, it may not be affected by a computer system virus because ROM is read-only. The issue is that updating this bootstrap code needs replacing the ROM hardware chips.

As a result, most computer systems include small bootstrap loader software in the boot ROM, whose primary function is to load a full bootstrap program from a disk drive. The entire bootstrap program can be modified easily, and the disk is rewritten with a fresh version. The bootstrap program is stored in a partition and is referred to as the **boot block.** A **boot disk** or **system disk** is a type of disk that contains a boot partition.

3. Bad Blocks

Disks are prone to failure due to their moving parts and tight tolerances. When a disk drive fails, it must be replaced and the contents transferred to the replacement disk using backup media. For some time, one or more sectors become faulty. Most disks also come from the company with bad blocks. These blocks are handled in various ways, depending on the use of disk and controller.

On the disk, the controller keeps a list of bad blocks. The list is initialized during the factory's low-level format and updated during the disk's life. Each bad sector may be replaced with one of the spare sectors by directing the controller. This process is referred to as sector sparing.

What is a directory?

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

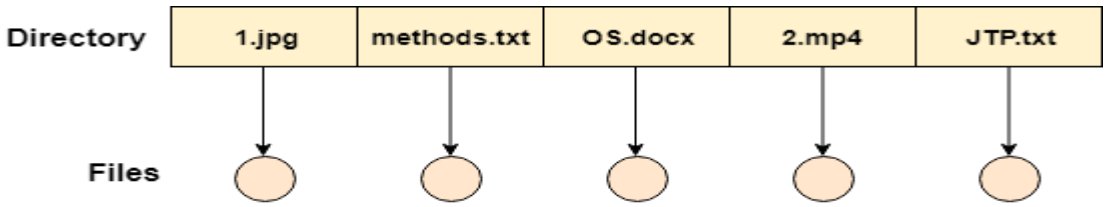
To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.
Each partition must have at least one directory in which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the information related to that file.

Every Directory supports a number of common operations on the file:

1. File Creation
2. Search for the file
3. File deletion
4. Renaming the file
5. Traversing Files
6. Listing of files

Single Level Directory

The simplest method is to have one big list of all the files on the disk. The entire system will contain only one directory which is supposed to mention all the files present in the file system. The directory contains one entry per each file present on the file system.



Single Level Directory

This type of directories can be used for a simple system.

Advantages

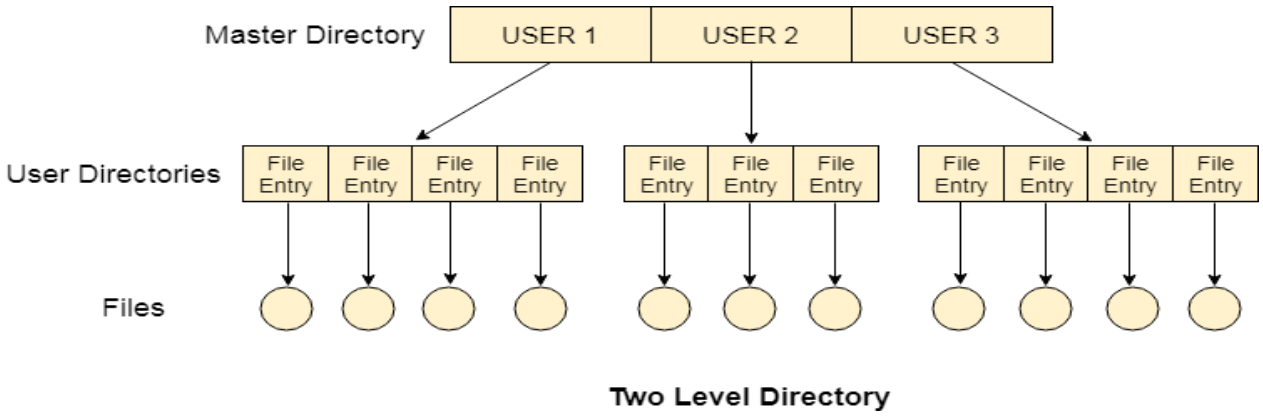
1. Implementation is very simple.
2. If the sizes of the files are very small then the searching becomes faster.
3. File creation, searching, deletion is very simple since we have only one directory.

Disadvantages

1. We cannot have two files with the same name.
2. The directory may be very big therefore searching for a file may take so much time.
3. Protection cannot be implemented for multiple users.
4. There are no ways to group same kind of files.
5. Choosing the unique name for every file is a bit complex and limits the number of files in the system because most of the Operating System limits the number of characters used to construct the file name.

Two Level Directory

In two level directory systems, we can create a separate directory for each user. There is one master directory which contains separate directories dedicated to each user. For each user, there is a different directory present at the second level, containing group of user's file. The system doesn't let a user to enter in the other user's directory without permission.



Two Level Directory

Characteristics of two level directory system

1. Each files has a path name as ***/User-name/directory-name/***
2. Different users can have the same file name.
3. Searching becomes more efficient as only one user's list needs to be traversed.
4. The same kind of files cannot be grouped into a single directory for a particular user.

Every Operating System maintains a variable as **PWD** which contains the present directory name (present user name) so that the searching can be done appropriately.

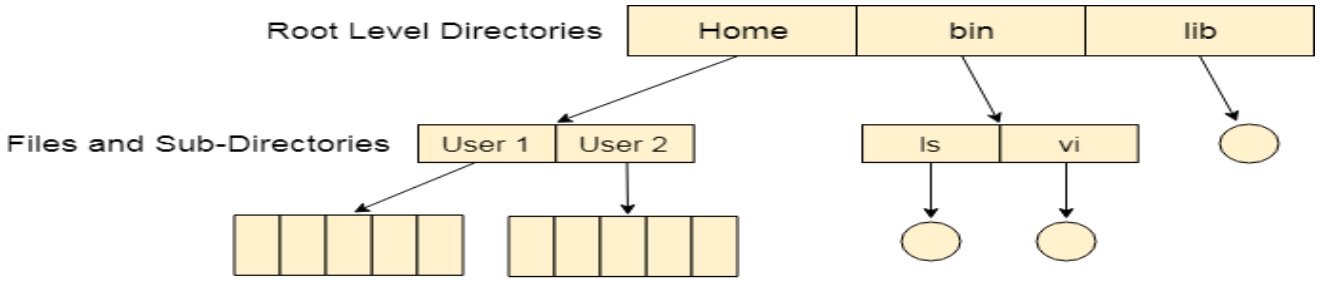
Tree Structured Directory

In Tree structured directory system, any directory entry can either be a file or sub directory. Tree structured directory system overcomes the drawbacks of two level directory system. The similar kind of files can now be grouped in one directory.

Each user has its own directory and it cannot enter in the other user's directory. However, the user has the permission to read the root's data but he cannot write or modify this. Only administrator of the system has the complete access of root directory.

Searching is more efficient in this directory structure. The concept of current working directory is used. A file can be accessed by two types of path, either relative or absolute.

Absolute path is the path of the file with respect to the root directory of the system while relative path is the path with respect to the current working directory of the system. In tree structured directory systems, the user is given the privilege to create the files as well as directories.



The Structured Directory System

File Systems

File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs. Some Operating systems treats everything as a file for example Ubuntu.
The File system takes care of the following issues

- **File Structure**
  We have seen various data structures in which the file can be stored. The task of the file system is to maintain an optimal file structure.
- **Recovering Free space**
  Whenever a file gets deleted from the hard disk, there is a free space created in the disk. There can be many such spaces which need to be recovered in order to reallocate them to other files.
- **disk space assignment to the files**
  The major concern about the file is deciding where to store the files on the hard disk. There are various disks scheduling algorithm which will be covered later in this tutorial.
- **tracking data location**
  A File may or may not be stored within only one block. It can be stored in the non - contiguous blocks on the disk. We need to keep track of all the blocks on which the part of the files reside.

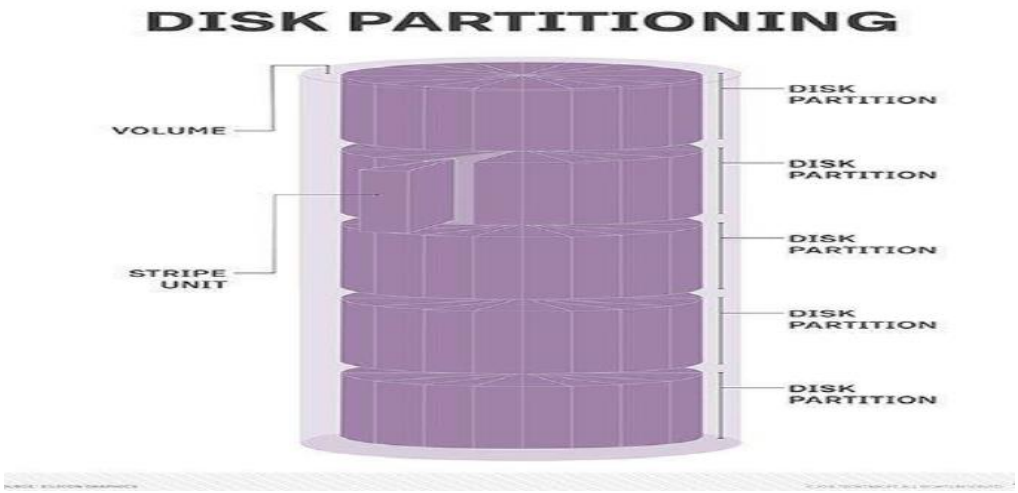
Disk Partition

A partition is a logical division of a hard disk that is treated as a separate unit by operating systems (OSes) and file systems. The OSes and file systems can manage information on each partition as if it were a distinct hard drive. This allows the drive to operate as several smaller sections to improve efficiency, although it reduces usable space on the hard disk because of additional overhead from multiple OSes.
A disk partition manager allows system administrators to create, resize, delete and manipulate partitions, while a partition table logs the location and size of the partition. Each partition appears to the OS as a distinct logical disk, and the OS reads the partition table before any other part of the disk. Once a partition is created, it is formatted with a file system such as:

- NTFS on Windows drives;
- FAT32 and exFAT for removable drives;
- HFS Plus (HFS+) on Mac computers; or
- Ext4 on Linux.

Data and files are then written to the file system on the partition. When users boot the OS in a computer, a critical part of the process is to give control to the first sector on the hard disk. This includes the partition table that defines how many partitions will be formatted on the hard disk, the size of each partition and the address where each disk partition begins. The sector also contains a program that reads the boot sector for the OS and gives it control so that the rest of the OS can be loaded into random access memory.

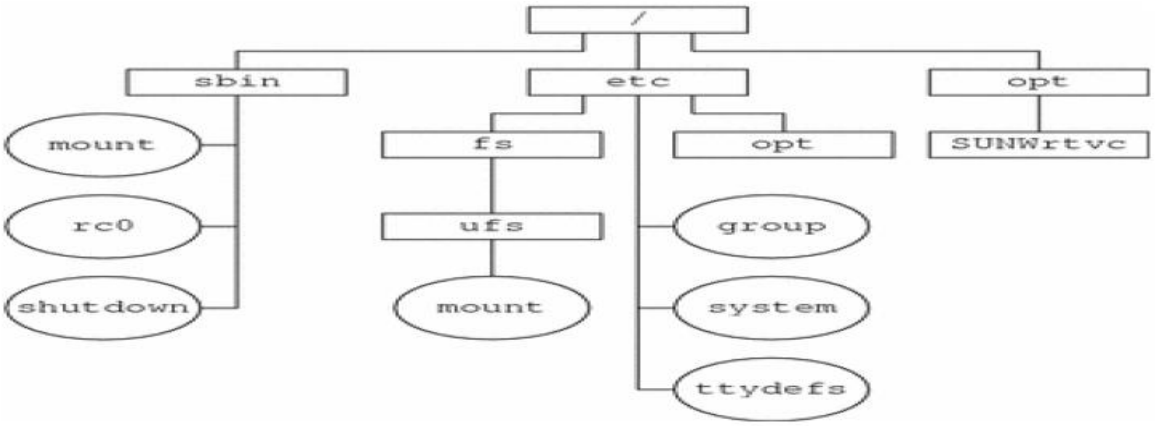
A key aspect of partitioning is the active or bootable partition, which is the designated partition on the hard drive that contains the OS. Only the partition on each drive that contains the boot loader for the OS can be designated as the active partition. The active partition also holds the boot sector and must be marked as active. A recovery partition restores the computer to its original shipping condition. In enterprise storage, partitioning helps enable short stroking, a practice of formatting a hard drive to speed performance through data placement.



There are three types of partitions: primary partitions, extended partitions and logical drives. A disk may contain up to four primary partitions (only one of which can be active), or three primary partitions and one extended partition.

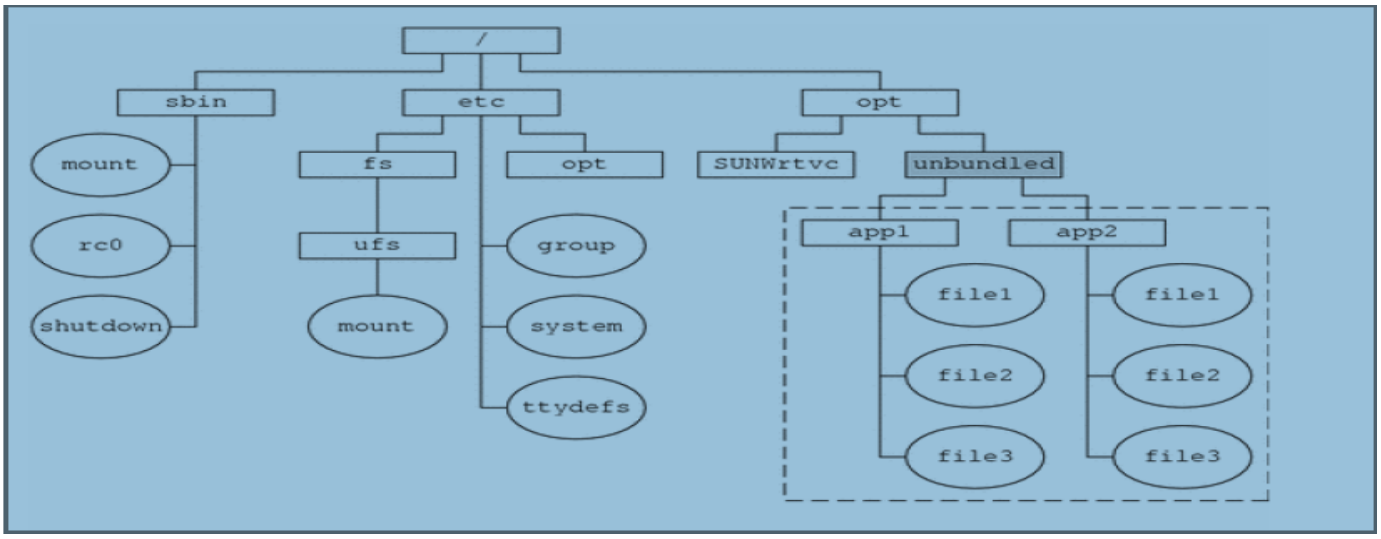
Mounting and Unmounting File Systems

Before you can access the files on a file system, you need to mount the file system. When you mount a file system, you attach that file system to a directory (mount point) and make it available to the system. The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system. When you mount a file system, any files or directories in the underlying mount point directory are unavailable as long as the file system is mounted. These files are not permanently affected by the mounting process, and they become available again when the file system is unmounted. However, mount directories are typically empty, because you usually do not want to obscure existing files.
For example, the following figure shows a local file system, starting with a root (/) file system and the sbin, etc, and opt subdirectories.



To access a local file system from the /opt file system that contains a set of unbundled products, you must do the following:

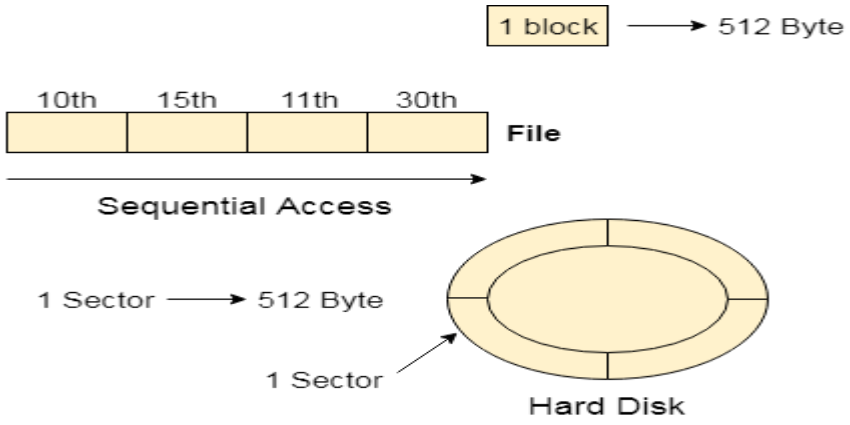
- First, you must create a directory to use as a mount point for the file system you want to mount, for example, /opt/unbundled.
- Once the mount point is created, you can mount the file system (by using the mount command), which makes all of the files and directories in /opt/unbundled available, as shown in the following figure.



File Access Methods

Let's look at various ways to access files stored in secondary memory.

Sequential Access



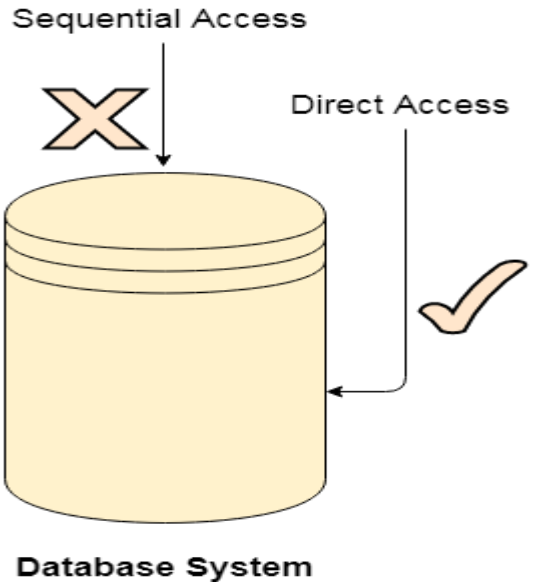
Most of the operating systems access the file sequentially. In other words, we can say that most of the files need to be accessed sequentially by the operating system.
In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file. If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word. This process continues till the end of the file.
Modern word systems do provide the concept of direct access and indexed access but the most used method is sequential access due to the fact that most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

Direct Access

The Direct Access is mostly required in the case of database systems. In most of the cases, we need filtered information from the database. The sequential access can be very slow and inefficient in such cases.

Suppose every block of the storage stores 4 records and we know that the record we needed is stored in 10th block. In that case, the sequential access will not be implemented because it will traverse all the blocks in order to access the needed record.

Direct access will give the required result despite of the fact that the operating system has to perform some complex tasks such as determining the desired block number. However, that is generally implemented in database applications.



Indexed Access

If a file can be sorted on any of the filed then an index can be assigned to a group of certain records. However, A particular record can be accessed by its index. The index is nothing but the address of a record in the file.

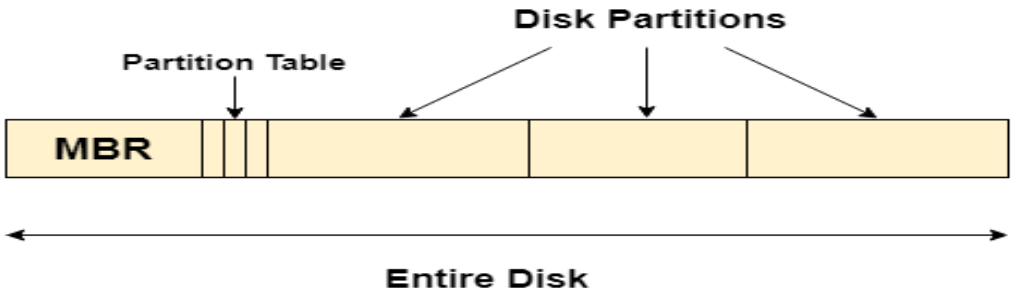
In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.

Master Boot Record (MBR)

Master boot record is the information present in the first sector of any hard disk. It contains the information regarding how and where the Operating system is located in the hard disk so that it can be booted in the RAM.

MBR is sometimes called master partition table because it includes a partition table which locates every partition in the hard disk.

Master boot record (MBR) also includes a program which reads the boot sector record of the partition that contains operating system.



What happens when you turn on your computer?

Due to the fact that the main memory is volatile, when we turn on our computer, CPU cannot access the main memory directly. However, there is a special program called as BIOS stored in ROM is accessed for the first time by the CPU.

BIOS contains the code, by executing which, the CPU access the very first partition of hard disk that is MBR. It contains a partition table for all the partitions of the hard disk.

Since, MBR contains the information about where the operating system is being stored and it also contains a program which can read the boot sector record of the partition, hence the CPU fetches all this information and load the operating system into the main memory.

Allocation Methods

There are various methods which can be used to allocate disk space to the files. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system. Allocation method provides a way in which the disk will be utilized and the files will be accessed.

There are following methods which can be used for allocation.

1. Contiguous Allocation.
2. Extents
3. Linked Allocation
4. Clustering
5. FAT
6. Indexed Allocation
7. Linked Indexed Allocation
8. Multilevel Indexed Allocation
9. Inode

We will discuss three of the most used methods in detail.

1. Contiguous Allocation

If the blocks are allocated to the file in such a way that all the logical blocks of the file get the contiguous physical block in the hard disk then such allocation scheme is known as contiguous allocation.

In the image shown below, there are three files in the directory. The starting block and the length of each file are mentioned in the table. We can check in the table that the contiguous blocks are assigned to each file as per its need.

Advantages

1. It is simple to implement.
2. We will get Excellent read performance.

3. Supports Random Access into files.

Disadvantages

1. The disk will become fragmented.
2. It may be difficult to have a file grow.

2. Linked List Allocation

Linked List allocation solves all problems of contiguous allocation. In linked list allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.

Advantages

1. There is no external fragmentation with linked allocation.
2. Any free block can be utilized in order to satisfy the file block requests.
3. File can continue to grow as long as the free blocks are available.
4. Directory entry will only contain the starting block address.

Disadvantages

1. Random Access is not provided.
2. Pointers require some space in the disk blocks.
3. Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.
4. Need to traverse each block.

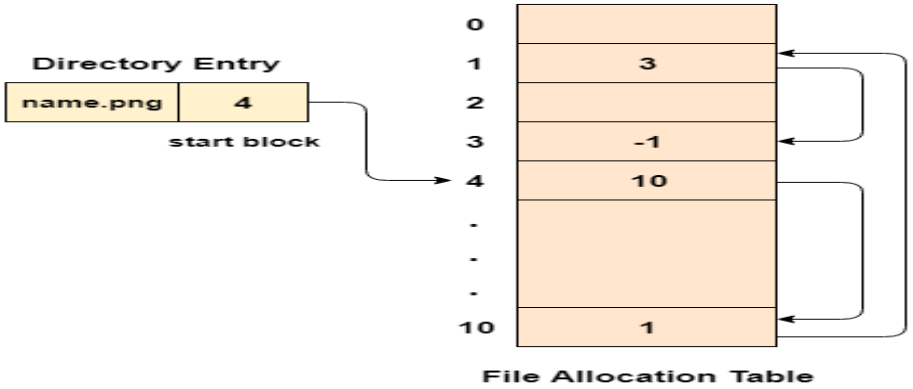
File Allocation Table

The main disadvantage of linked list allocation is that the Random access to a particular block is not provided. In order to access a block, we need to access all its previous blocks.

File Allocation Table overcomes this drawback of linked list allocation. In this scheme, a file allocation table is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number.

File allocation table needs to be cached in order to reduce the number of head seeks. Now the head doesn't need to traverse all the disk blocks in order to access one successive block.

It simply accesses the file allocation table, read the desired block entry from there and access that block. This is the way by which the random access is accomplished by using FAT. It is used by MS-DOS and pre-NT Windows versions.



Advantages

1. Uses the whole disk block for data.
2. A bad disk block doesn't cause all successive blocks lost.
3. Random access is provided although its not too fast.
4. Only FAT needs to be traversed in each file operation.

Disadvantages

1. Each Disk block needs a FAT entry.
2. FAT size may be very big depending upon the number of FAT entries.
3. Number of FAT entries can be reduced by increasing the block size but it will also increase Internal Fragmentation.

3. Indexed Allocation

Limitation of FAT

File allocation table tries to solve as many problems as possible but leads to a drawback. The more the number of blocks, the more will be the size of FAT.

Therefore, we need to allocate more space to a file allocation table. Since, file allocation table needs to be cached therefore it is impossible to have as many space in cache. Here we need a new technology which can solve such problems.

Instead of maintaining a file allocation table of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.

Advantages

1. Supports direct access
2. A bad data block causes the loss of only that block.

Disadvantages

1. A bad index block could cause the loss of entire file.
2. Size of a file depends upon the number of pointers, an index block can hold.
3. Having an index block for a small file is totally wastage.
4. More pointer overhead

Free Space Management

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

1. Bit Vector

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

LAs the space allocation proceeds, the file system starts allocating blocks to the files and setting the respective bit to 0.

2. Linked List

It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.
Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.

Disk Scheduling

As we know, a process needs two types of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.

However, the operating system must be fair enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.
The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.
Let's discuss some important terms related to disk scheduling.

Seek Time: Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

Rotational Latency: It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

Transfer Time: It is the time taken to transfer the data.

Disk Access Time: Disk access time is given as,

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

Disk Response Time: It is the average of time spent by each request waiting for the IO operation.

Purpose of Disk Scheduling

The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms

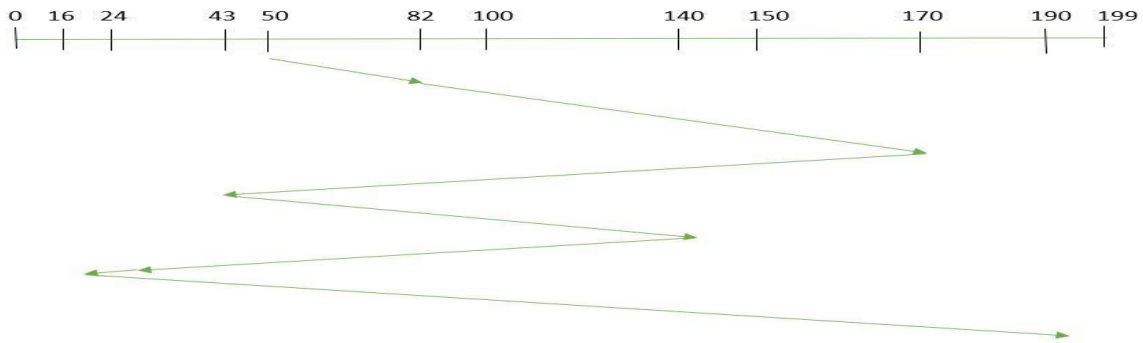
The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

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

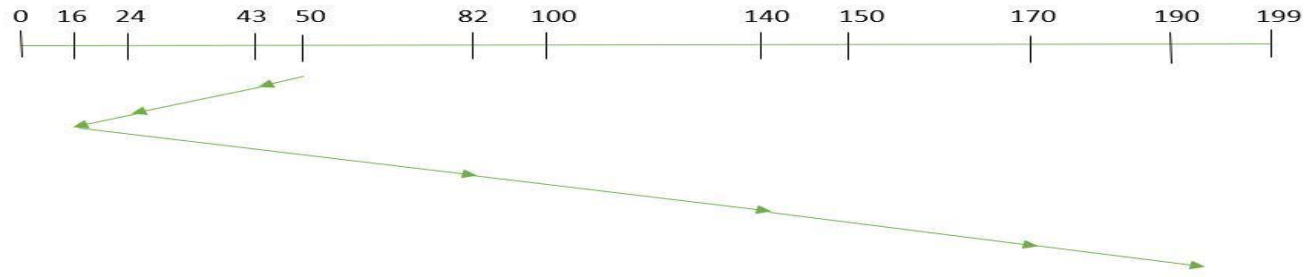
Disadvantages:

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

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.

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

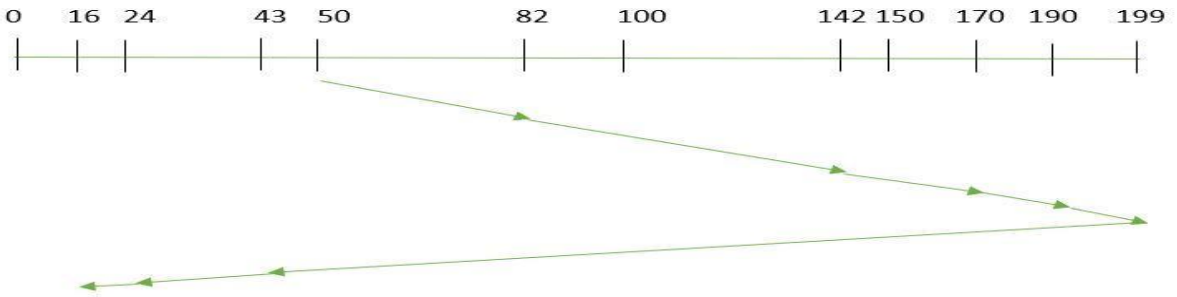
Disadvantages:

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

SCAN: In SCAN algorithm the disk arm moves in a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and is hence also known as an **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

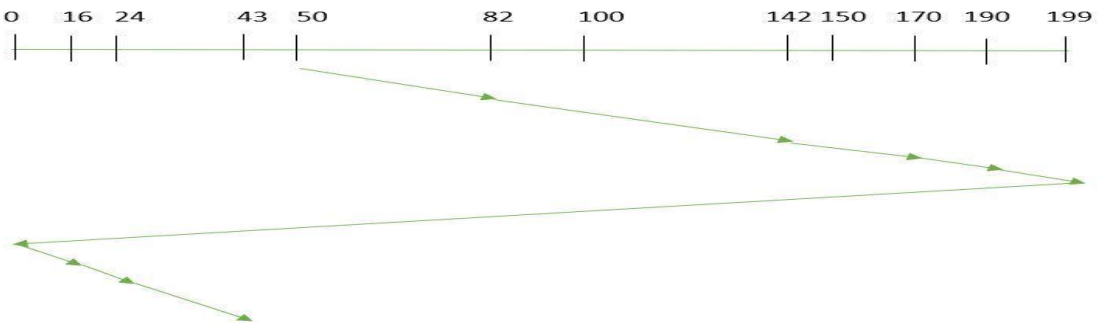
Disadvantages: Long waiting time for requests for locations just visited by disk arm

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"**



.

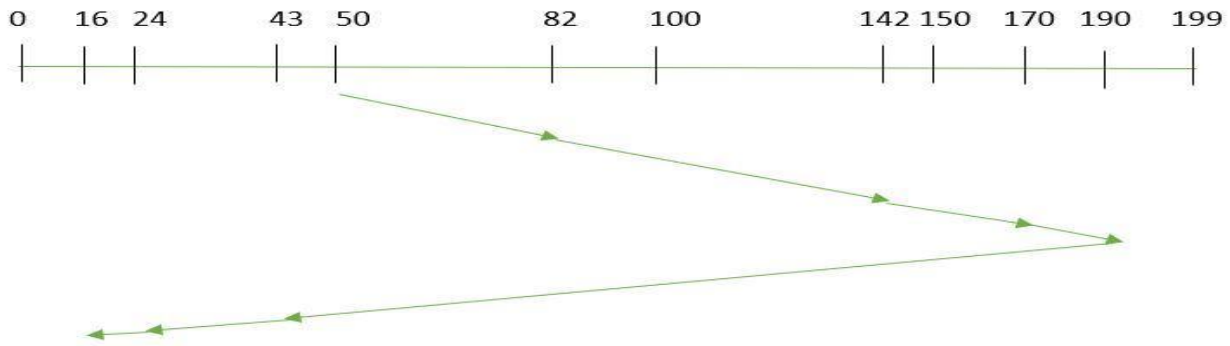
Seek time is calculated as: =(199-50)+(199-0)+(43-0) =391

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".**

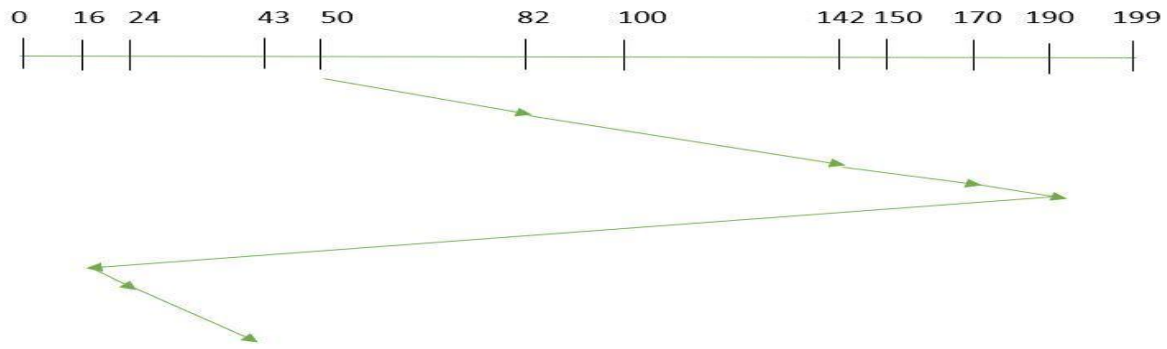


So, the seek time is calculated as: =(190-50)+(190-16) =314

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

Backup

A system backup is the process of backing up the operating system, files and system specific useful/essential data. Backup is a process in which the state, files and data of a computer system are duplicated to be used as a backup or data substitute when the primary system data is corrupted, deleted or lost. There are mainly three types of backups are there: Full backup, differential backup and incremental backup. Let's take a look at each type of backup and its respective pros and cons.

System backup primarily ensures that not only the user data in a system is saved, but also the system's state or operational condition. This helps in restoring the system to the last saved state along with all the selected backup data. Generally, the system backup is performed through backup software and the end file (system backup) generated through this process is known as the system snapshot/image. Moreover, in a networked/enterprise environment, the system backup file/snapshot/image is routinely uploaded and updated on an enterprise local/remote storage server.

Evolution of backup

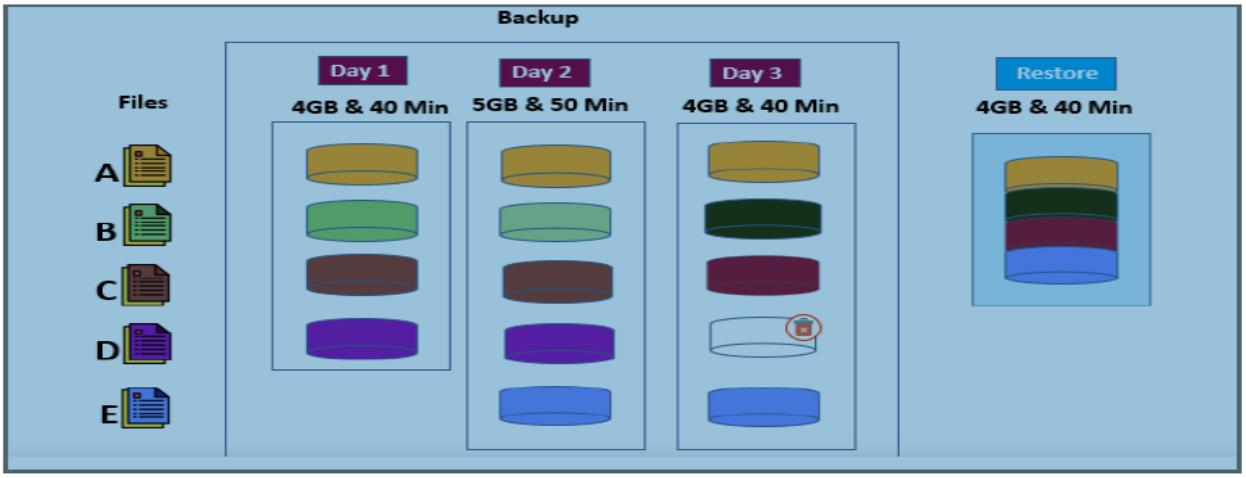
Backup techniques have evolved over time and become increasingly sophisticated (and perhaps complex as a result). Considerations such as time taken for backup, time taken for restores, storage costs, network bandwidth savings, etc. –have all, over time, driven innovations that have been designed to make backups better – but also increase complexity as a result.

Types of Backups

1. Full backup

They are the simplest form of backup and the easiest to understand. Full backup essentially makes a backup of everything you wish to protect every time. So, all files, objects, bytes however you wish to measure your data – every one of them is copied over to a secondary storage target each time. If you perform a full backup once a day – then everything is copied over once a day. Let's take an example – say you have 4 files A, B, C, and D. And let's say each of them is about 1GB each and each of them takes 10 mins to backup.

- On Day 1 – you'll backup 4GB and it'll take you 40 mins
- On Day 2, let's say File B changes to B1, and a new File called E is added. A-C & D remain the same.
- When you run the backup on Day 2, it'll backup all 5 files and it'll take you 50mins
- One Day 3, let's say File B changes again and becomes B2. File C also changes to C1, and File D gets deleted.
- When you run the backup on Day 3, it'll backup 4 files again (D is removed remember?) and it'll take you 40 mins.

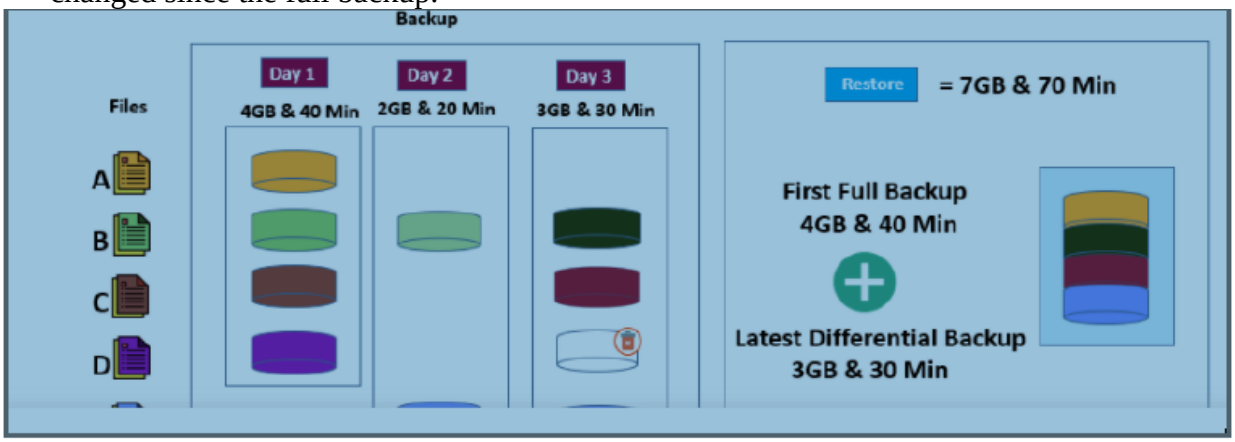


When you restore, you will most likely get data from the latest backup and it'll take you 40 mins to restore.

2. Differential backup

Full backups, as you can see take time. 40 minutes – 50 minutes each day as in our example. The next optimization the industry made was a differential backup. Differential backup makes a copy of files that have changed since the full backup. Let's take the same example – say you have 4 files A, B, C, and D. And let's say each of them is about 1GB each and each of them takes 10 mins to backup

- On Day 1 – you'll backup 4GB and it'll take you 40 mins
- On Day 2, let's say File B changes to B1, and a new File called E is added. Files A, C & D stay the same.
- When you run the backup on Day 2, it'll backup just the 2 changed files and the backup will take you 20mins
- One Day 3, let's say File B changes again and becomes B2. File C also changes to C1 and File D gets deleted.
- When you run the backup on Day 3, it'll backup 3 files (B2, C1, and E) and it'll take you 30mins. Why did we backup E? Remember a differential backup picks up everything that changed since the full backup.



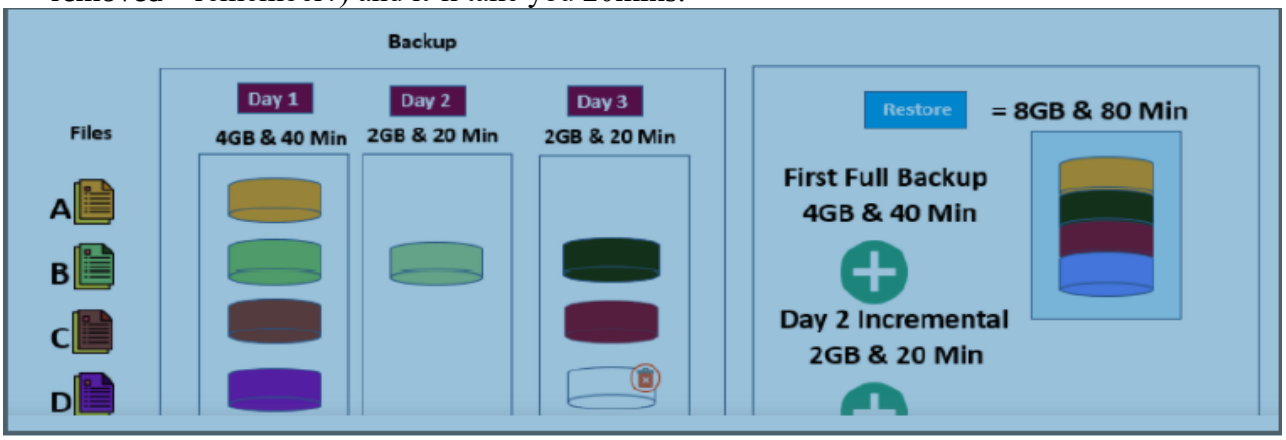
When you restore data, just like in the previous case you'll need to restore the Full backup first and then layer in each incremental backup on top of that – in order. So, while we have been able to improve backup speeds progressively, we have traded off restore times in each of the above cases. It is for this reason that traditional types of backup strategies recommend doing a full backup at frequent points in time – weekly, monthly, quarterly, yearly, etc. The idea is to ensure that you're able to keep restore times in check. If you're able to start the restore from a recent full backup, then the number of subsequent backups to restore and overlay on top of it are limited – thus saving time. But modern, enterprise-class backup technology has progressed further than this and will allow you to have the best of both worlds. Fast incremental backups and fast restores. The secret is something called cataloging.

3. Incremental backup

But if you think about it, differential backup has the potential to keep getting bigger and take longer and longer each day. After all, they're backing up all changes since the full backup. So, there could come a point where a daily differential backup is taking as much time as a full backup (or perhaps more). Enter the next innovation – incremental backup. Incremental backup only backup what was changed since the last backup. Sounds efficient right? Let's look at this with the same example:
So, you have 4 files A, B, C, and D. And let's say each of them is about 1GB each and each of them takes 10mins to backup.

- On Day 1 – you'll backup 4GB and it'll take you 40mins.
- On Day 2, let's say File B changes to B1, and a new File called E is added.

- When you run the backup on Day 2, it'll backup just the 2 changed files – and it'll take you 20mins.
- One Day 3, let's say File B changes again to B2. File C also changes to C1 and File D gets deleted.
- When you run the backup on Day 3, it'll backup just the 2 files again (B2 and C1) (D is removed – remember?) and it'll take you 20mins.



Let's see what happens when you restore. When you restore data, just like in the previous case – you'll need to restore the Full backup first – and then layer in each incremental backup on top of that – in order.

If you wish to get back the latest copy of data, it'll take us 80 mins to restore – that 40 + 20 + 20. So, not great from a restore standpoint.

So, while we have been able to improve backup speeds progressively, we have traded off restore times in each of the above cases.

It is for this reason that traditional types of backup strategies recommend doing a full backup at frequent points in time – weekly, monthly, quarterly, yearly, etc. The idea is to ensure that you're able to keep restore times in check. If you're able to start the restore from a recent full backup, then the number of subsequent backups to restore and overlay on top of it are limited – thus saving time.

But modern, enterprise-class backup technology has progressed further than this – and will allow you to have the best of both worlds.

Fast incremental backups and fast restores. The secret is something called cataloging.

Modern enterprise class backup technology has progressed further and now offers best of both the worlds – Fast incremental backups and fast restores. The secret is something called Cataloging.

Cataloging is a meta-data: i.e., data about your data. In each of the above cases, our backups were self-describing. There is no additional information required to restore from any of those backups.

But **meta-data describes data:** it can contain interesting information about file versions, about locations on media files, are kept, etc. – which can dramatically improve restore performance. Let's try the same example – this time with cataloging. So, you have 4 files A, B, C, and D. And let's say each of them is about 1GB each and each of them takes 10mins to backup

- On Day 1 – you'll backup 4GB and it'll take you 40mins.
- On Day 2, let's say File B changes to B1 and a new File called E is added.
- When you run the backup on Day 2, it'll backup just the 2 changed files – and it'll take you 20mins
- One Day 3, let's say File B changes again to B2. File C also changes to C1, and File D gets deleted.
- When you run the backup on Day 3, it'll backup just the 2 files again (B2 and C1) (D is removed – remember?) and it'll take you 20mins
- Now, when you restore, the catalog will supply you the latest version of each file automatically – So A, B2, C1 and E. In 40 minutes. And you'll not even bother bringing back D because the catalog knows it was deleted.