File Systems in Operating System

A computer file is defined as a medium used for saving and managing data in the computer system. The data stored in the computer system is completely in digital format, although there can be various types of files that help us to store the data.

What is a File System?

A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device. Some common types of file systems include:

1. **FAT (File Allocation Table):** An older file system used by older versions of Windows and other operating systems.
2. **NTFS (New Technology File System):** A modern file system used by Windows. It supports features such as file and folder permissions, compression, and encryption.
3. **Ext (Extended File System):** A file system commonly used on Linux and Unix-based operating systems.
4. **HFS (Hierarchical File System):** A file system used by macOS.
5. **APFS (Apple File System):** A new file system introduced by Apple for their Macs and iOS devices.

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From the user's perspective, a file is the smallest allotment of logical secondary storage.

The name of the file is divided into two parts as shown below:

- name
- extension, separated by a period.

Issues Handled by File System

We've seen a variety of data structures where the file could be kept. The file system's job is to keep the files organized in the best way possible. A free space is created on the hard drive whenever a file is deleted from it. To reallocate them to other files, many of these spaces may need to be recovered. Choosing where to store the files on the hard disc is the main issue with files one block may or may not be used to store a file. It may be kept in the disk's non-contiguous blocks. We must keep track of all the blocks where the files are partially located.

Files Attributes And Their Operations

Attributes	**Types**	**Operations**
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation Data	Xis	Write
Author	C	Append
Last Modified	Java	Truncate
protection	class	Delete
		Close

File type	Usual extension	Function
Executable	exe, com, bin	Read to run machine language program
Object	obj, o	Compiled, machine language not linked
Source Code	C, java, pas, asm, a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, tex, rrf, doc	Various word processor formats
Archive	arc, zip, tar	Related files grouped into one compressed file
Multimedia	mpeg, mov, rm	For containing audio/video information
Markup	xml, html, tex	It is the textual data and documents

File type	Usual extension	Function
Library	lib, a ,so, dll	It contains libraries of routines for programmers
Print or View	gif, pdf, jpg	It is a format for printing or viewing an ASCII or binary file.

File Directories

The collection of files is a file directory. The directory contains information about the files, including attributes, location, and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Below are information contained in a device directory.

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

The operation performed on the directory are:

- Search for a file
- Create a file

- Delete a file
- List a directory
- Rename a file
- Traverse the file system

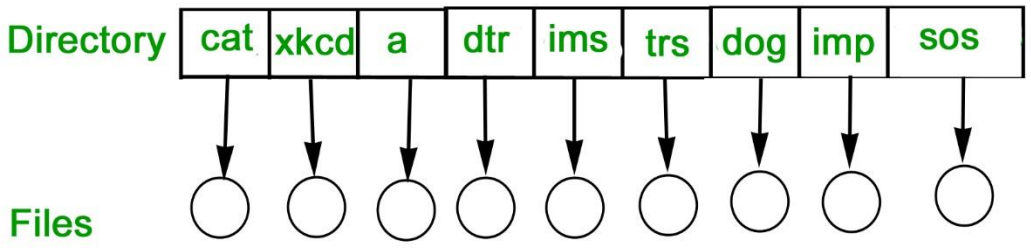
Advantages of Maintaining Directories

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

Single-Level Directory

In this, a single directory is maintained for all the users.

- **Naming problem:** Users cannot have the same name for two files.
- **Grouping problem:** Users cannot group files according to their needs.

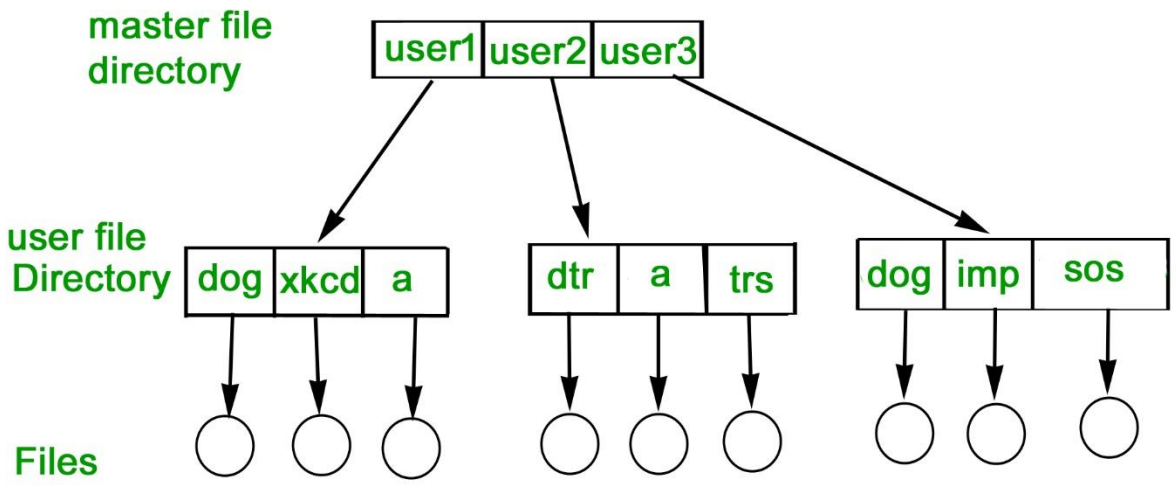


Two-Level Directory

In this separate directories for each user is maintained.

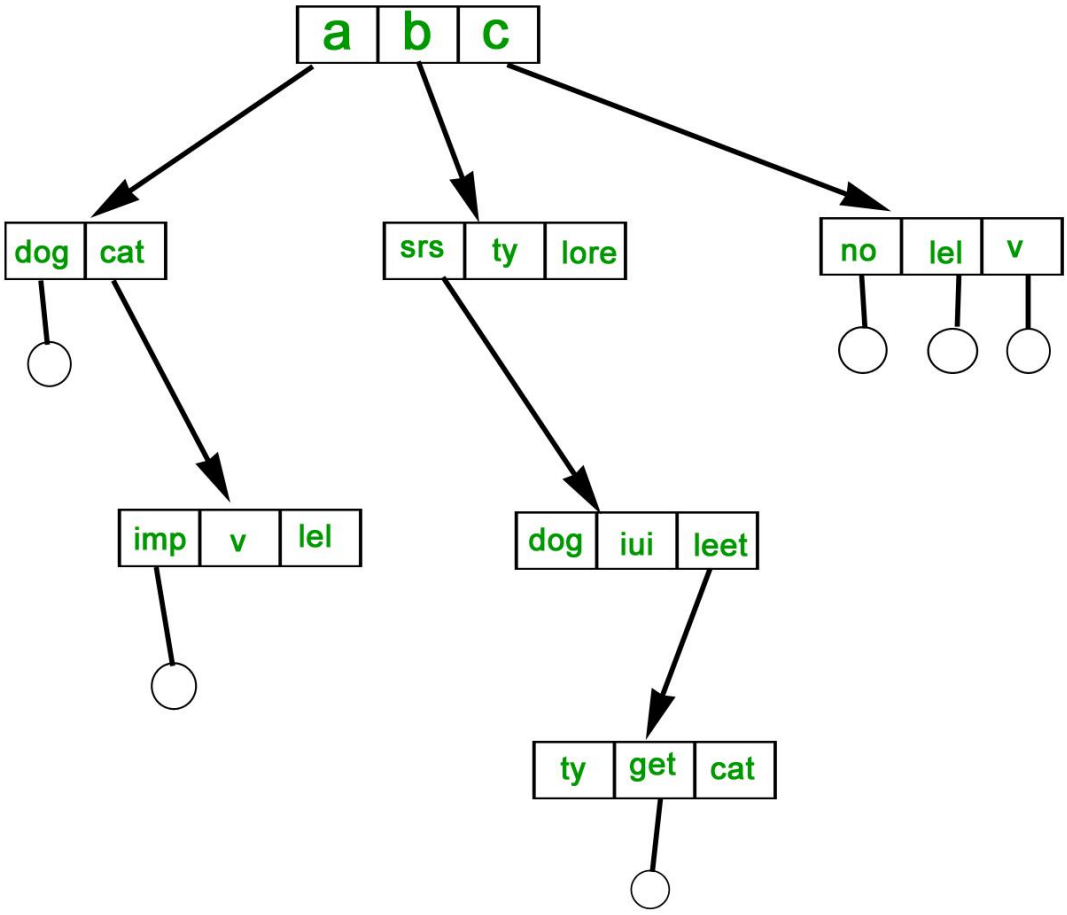
- **Path name**: Due to two levels there is a path name for every file to locate that file.
- Now, we can have the same file name for different users.

- Searching is efficient in this method.



Tree-Structured Directory

The directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



File Allocation Methods

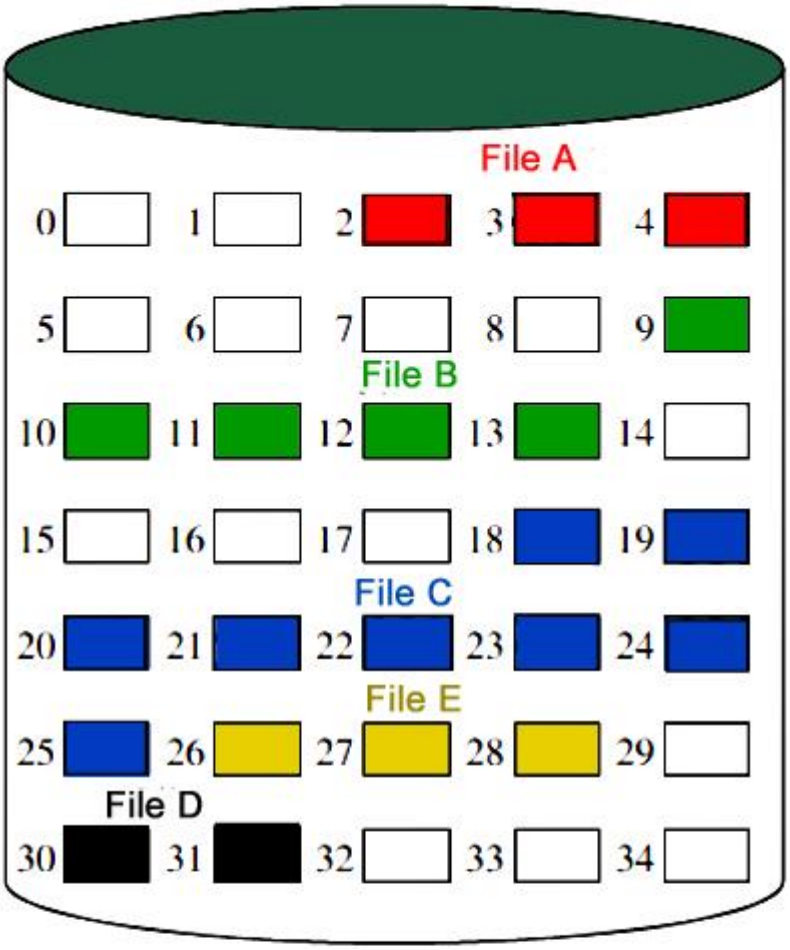
There are several types of file allocation methods. These are mentioned below.

- Continuous Allocation
- Linked Allocation(Non-contiguous allocation)
- Indexed Allocation

Continuous Allocation

A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block.

For example, if a file starts at block b, and the ith block of the file is wanted, its location on secondary storage is simply b+i-1.



File allocation table

File name	Start block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Disadvantages of Continuous Allocation

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. A compaction algorithm will be necessary to free up additional space on the disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

Linked Allocation(Non-Contiguous Allocation)

Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. An increase in file size is always possible if a free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of the file.

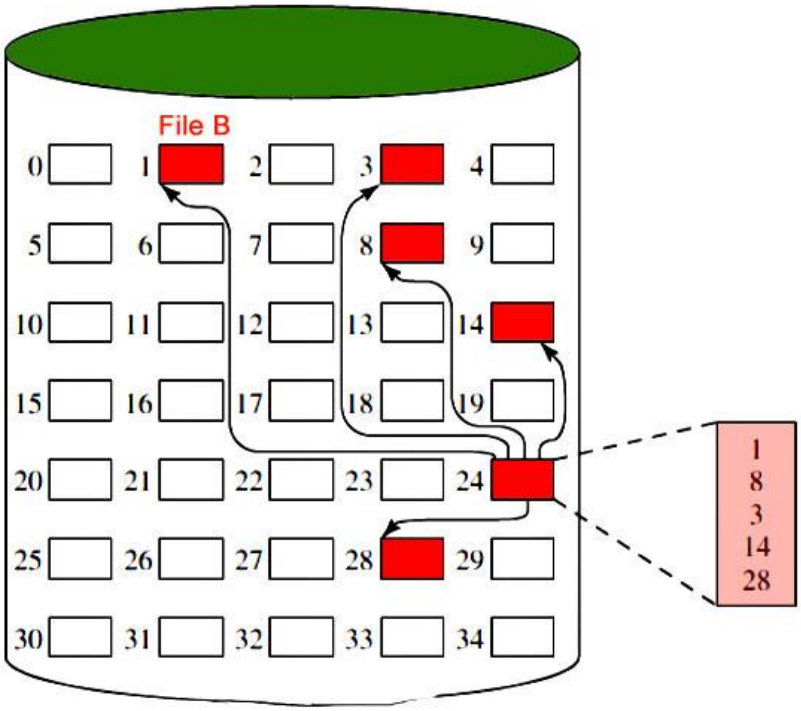
Disadvantage Linked Allocation(Non-contiguous allocation)

- Internal fragmentation exists in the last disk block of the file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.
- It supports only the sequential access of files.

Indexed Allocation

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The

index has one entry for each block allocated to the file. The allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.



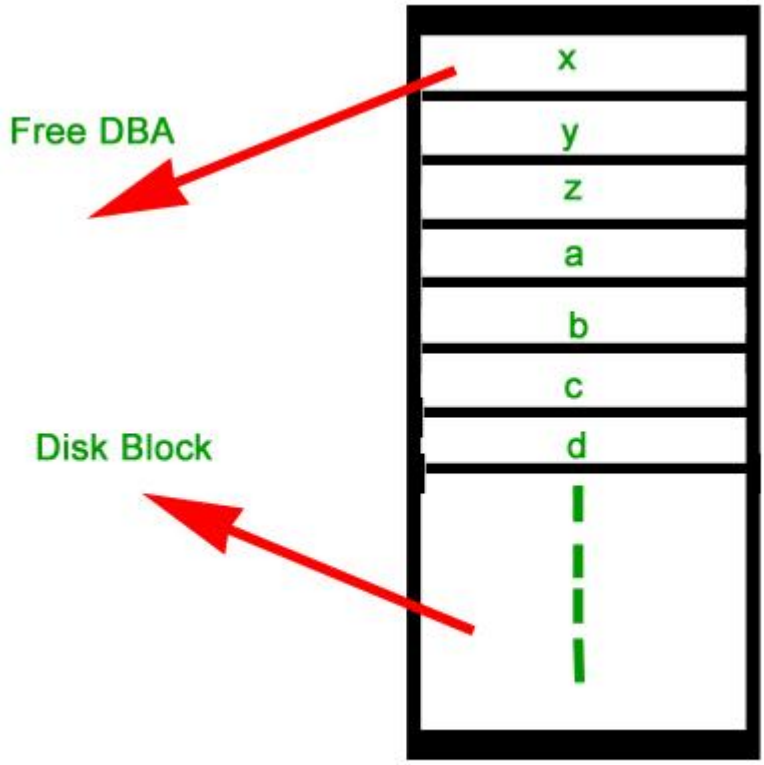
File allocation table

File name	Index block
•••	•••
File B	24
•••	•••

Disk Free Space Management

Just as the space that is allocated to files must be managed, so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques, it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table. The following are the approaches used for free space management.

1. **Bit Tables**: This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.
   For example 00011010111100110001
   In this vector every bit corresponds to a particular block and 0 implies that that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.
2. **Free Block List**: In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.



Advantages of File System

- **Organization:** A file system allows files to be organized into directories and subdirectories, making it easier to manage and locate files.
- **Data protection:** File systems often include features such as file and folder permissions, backup and restore, and error detection and correction, to protect data from loss or corruption.
- **Improved performance:** A well-designed file system can improve the performance of reading and writing data by organizing it efficiently on disk.

Disadvantages of File System

- **Compatibility issues:** Different file systems may not be compatible with each other, making it difficult to transfer data between different operating systems.

- **Disk space overhead:** File systems may use some disk space to store metadata and other overhead information, reducing the amount of space available for user data.
- **Vulnerability:** File systems can be vulnerable to data corruption, malware, and other security threats, which can compromise the stability and security of the system.

FAQs on File System

Q.1: How does a file system organize data?

Answer:

A file system organizes data by using a hierarchical structure consisting of directories (also called folders) and files. Directories can contain both files and subdirectories, forming a tree-like structure. This allows users to organize their files into meaningful groups and navigate through the file system using paths or directory structures.

Q.2: What is a file allocation table (FAT)?

Answer:

The File Allocation Table (FAT) is a file system structure used by some operating systems, including older versions of Windows. It uses a table to keep track of the allocation status of each cluster (a fixed-size block of storage) on the disk. The FAT file system has evolved over time, with variations such as FAT12, FAT16, and FAT32, supporting different disk sizes and features.

Q.3: What is NTFS (New Technology File System)?

Answer:

NTFS (New Technology File System) is a file system introduced by Microsoft with Windows NT. It is the default file system used by modern versions of Windows, including Windows XP, Windows 7, Windows 10, and Windows Server editions. NTFS offers features such as improved performance, reliability, security, support for large file sizes and volumes, file compression, encryption, and access control.

Transforming of I/O Requests to Hardware Operations

We know that there is handshaking between device driver and device controller but here question is that how operating system connects application request or we can say I/O request to set of network wires or to specific disk sector or we can say to hardware -operations.

To understand concept let us consider example which is as follows.

Example –

We are reading file from disk. The application we request for will refers to data by file name. Within disk, file system maps from file name through file-system directories to obtain space allocation for file. In MS-DOS, name of file maps to number that indicates as entry in file-access table, and that entry to table tells us that which disk blocks are allocated to file. In UNIX, name maps to inode number, and inode number contains information about space-allocation. But here question arises that how connection is made from file name to disk controller?

The method that is used by MS-DOS, is relatively simple operating system. The first part of MS-DOS file name, is preceding with colon, is string that identifies that there is specific hardware device.

The UNIX uses different method from MS-DOS. It represents device names in regular file-system name space. Unlike MS-DOS file name, which has colon separator, but UNIX path name has no clear separation of device portion. In fact, no part of path name is name of device. Unix has mount table that associates with prefixes of path names with specific hardware device names.

Modern operating systems gain significant flexibility from multiple stages of lookup tables in path between request and physical device stages controller. There is general mechanisms is which is used to pass request between

application and drivers. Thus, without recompiling kernel, we can introduce new devices and drivers into computer. In fact, some operating system have the ability to load device drivers on demand. At the time of booting, system firstly probes hardware buses to determine what devices are present. It is then loaded to necessary drivers, accordingly I/O request.

The typical life cycle of blocking read request, is shown in the following figure. From figure, we can suggest that I/O operation requires many steps that together consume large number of CPU cycles.

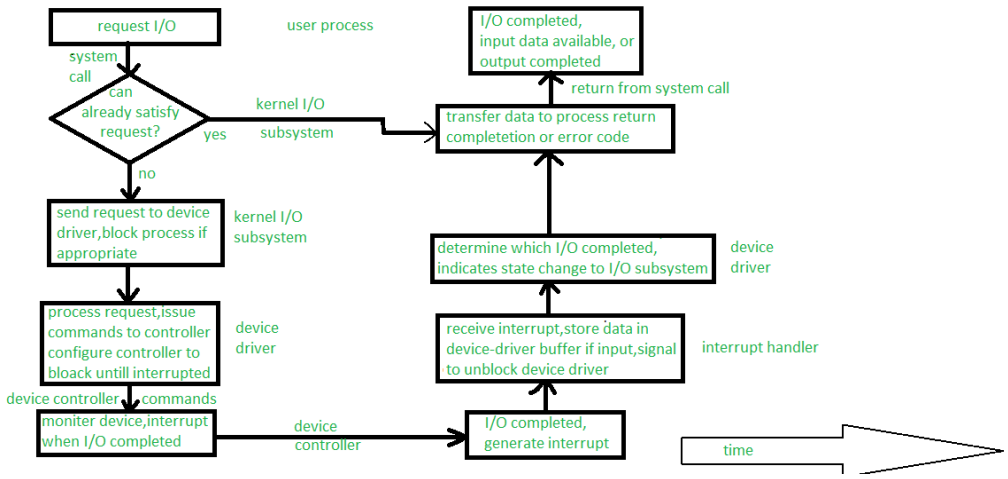


Figure – The life cycle of I/O request

1. Systemcall –

   Whenever, any I/O request comes, process issues blocking read() system call to previously opened file descriptor of file. Basically, role of system-call code is to check parameters for correctness in kernel. If data we put in form of input is already available in buffer cache, data is going to returned to process, and in that case I/O request is completed.

2. **Alternative approach if input is not available –**
   If the data is not available in buffer cache then physical I/O must be performed. The process is removes from run queue and is placed on wait queue for device, and I/O request is scheduled. After scheduling, I/O subsystem sends request to device driver via subroutine call or in-kernel message but it depends upon operating system by which mode request will send.
3. **Role of Device driver –**
   After receiving the request, device driver have to receive data and it will receive data by allocating kernel buffer space and after receiving data it will schedules I/O. After all this, command will be given to device controller by writing into device-control registers.
4. **Role of Device Controller –**
   Now, device controller operates device hardware. Actually, data transfer is done by device hardware.
5. **Role of DMA controller –**
   After data transfer, driver may poll for status and data, or it may have set up DMA transfer into kernel memory. The transfer is managed by DMA controller. At last when transfers complete, it will generates interrupt.
6. **Role of interrupt handler –**
   The interrupt is send to correct interrupt handler through interrupt-vector table. It store any necessary data, signals device driver, and returns from interrupt.
7. **Completion of I/O request –**
   When, device driver receives signal. This signal determines that I/O request has completed and also determines request's status, signals kernel I/O subsystem that request has been completed. After

transferring data or return codes to address space kernel moves process from wait queue back to ready queue.

8. **Completion of System call –** When process moves to ready queue it means process is unblocked. When the process is assigned to CPU, it means process resumes execution at completion of system call.

I/O Hardware in Operating System

I/O Hardware is a set of specialized hardware devices that help the operating system access disk drives, printers, and other peripherals. These devices are located inside the motherboard and connected to the processor using a bus. They often have specialized controllers that allow them to quickly respond to requests from software running on top of them or even respond directly to commands from an application program. This post will discuss in detail I/O Hardware basics such as daisy chain expansion bus controller memory-mapped I/O Direct Memory Access (DMA)

The Daisy chain, expansion bus, controller, and host adapter are used to access the I/O hardware.

The **daisy chain** is a method of connecting multiple I/O devices with each other through a single connection point (pin). Each device can be accessed by plugging into any of the pins on this connection point. The **expansion bus** connects devices together in parallel with each other so that they can be accessed simultaneously by using only one cable instead of several cables (one per device). This design allows you to connect more than one peripheral device while maintaining compatibility with older systems that may not support additional peripherals or features such as **memory-mapped I/O** (MMIO).

A **controller** manages all incoming data from its associated port and sends outgoing commands from its associated port; it's like an interface between an application software program and hardware components such as disk drives or network adapters

A **host adapter** is a bridge between the system bus and the expansion bus. It allows you to connect multiple expansion cards at the same time and provides additional interfaces for those cards such as DMA or MMIO. A controller manages all incoming data from its associated port and sends outgoing commands from its associated port; it's like an interface between an application software program and hardware components such as disk drives or network adapters.

1. Polling

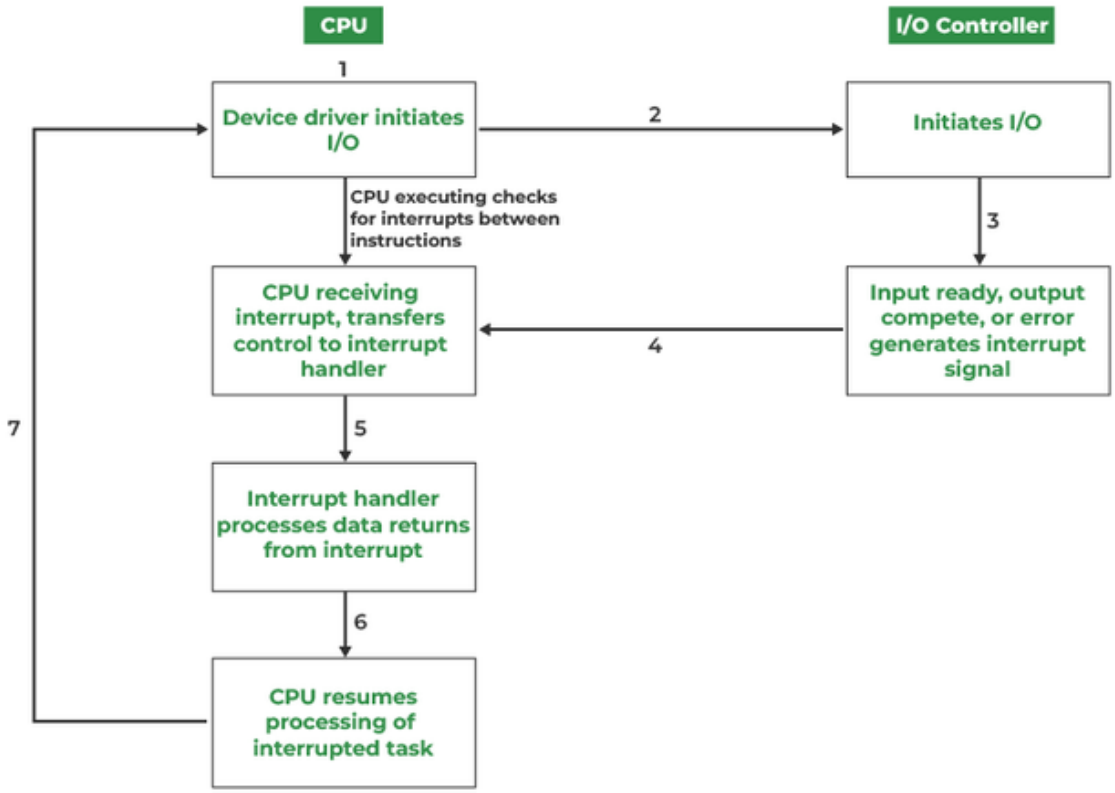
Polling is a software technique that uses a program to check the status of devices. The device can be a disk drive or any other peripheral device in the computer. The program polls the device for information, such as if it has data available or not. Polling is a slow way to get data from a device because it has to wait until another function occurs before being able to get information about its state.

In some cases polling may be desirable; for example, when there are several items being polled simultaneously and only one item updates its state at any time — the rest continue waiting until they receive an acknowledgment from one item that it's done updating their states (which could take seconds).

Polling can also be used to check if a device is online or not. If the device is offline, then this information can be used to take some appropriate action such as pausing or suspending tasks that depend on the device (e.g., stopping a backup in progress).

2. Interrupts

The CPU is interrupted by several different devices, such as I/O hardware and peripheral devices. These interrupts are used by the I/O device to notify the CPU when it needs attention. The CPU can be interrupted by several different devices at any given time, but only one interrupt will be delivered to it at any given time. This can happen because of a hardware or software error occurring on an I/O bus—for example, if a disk drive has failed then all other data transfers will pause until it's repaired; or perhaps another device wants access to memory (or vice versa). In either case, there won't be any way for your application program not to be written specifically for this particular machine!



Interrupt-driven I/O cycle

The operating system can also be interrupted by other processes that need CPU time. For example, if a user application program writes to a file and then exits, the operating system will write that file on disk before returning control back

to the user. This is called multitasking and allows many different programs to run at once.

The operating system is responsible for scheduling the CPU's time among multiple programs. This is done by using a program called an interrupt handler or interrupt service routine (ISR). An ISR is simply a piece of code that runs when an interrupt occurs and typically ends with returning control back to the original program.
It's important to note that the operating system is never interrupted by an ISR.

3. Direct Memory Access:

DMA is a way to move data between the CPU and I/O devices. When a user wants to access memory from an I/O device (such as a disk drive), the user must first inform the operating system that they are going to perform the operation. This is done by attaching a special command called an interrupt request (IRQ) handler routine for each device that needs access. The interrupt handler routine then tells the user program when it should be ready for another task—usually using interrupts generated by hardware components like joysticks or network cards—and may even tell other processes running on other CPUs not only about what's happening at that moment but also when something has happened before so they can take appropriate action as well!
The first step in using interrupts is to set up the interrupt controller. This is done by calling the BIOS INT 10h service with several parameters, including the number of IRQ lines available (16 is a common value), the maximum number of devices that can be connected to each line, and so on.

The next step is to assign an INT 10h handler for each device that needs access. This is done by calling the BIOS INT 15h service with several parameters, including the number of IRQ lines available (16 is a common value), the maximum number of devices that can be connected to each line, and so on

I/O Hardware Summary:

I/O hardware is a collection of devices that are used to make it easier for the CPU and other processors on your computer to communicate with the outside world. These devices may include:

Peripheral controller cards (also called I/O adapters) connect to the expansion bus (sometimes called PCI or ISA), and control how data is transferred between different parts of a computer system.

Interrupt request lines (IRQs) and serial ports, which allow multiple peripherals such as keyboards, mice, and printers to communicate with one another without having their own dedicated channel in memory.

A device driver software program that tells Windows what kind of hardware the user has installed so it can work properly; this may include information about how much RAM or hard drive space each device has available at any given moment in time because those things change over time depending on how much activity occurs within them throughout their life cycle.

Conclusion:

I/O Hardware is a key component of operating systems. It enables communication between the computer and other devices such as keyboards, mouse, printers, etc. I/O hardware has evolved over time and today there are many different types of I/O interfaces available in modern computers that can be used depending on the specific needs of your project or application

Access matrix in Operating System

Access Matrix is a security model of protection state in computer system. It is represented as a matrix. Access matrix is used to define the rights of each process executing in the domain with respect to each object. The rows of matrix represent domains and columns represent objects. Each cell of matrix represents set of access rights which are given to the processes of domain means each entry(i, j) defines the set of operations that a process executing in domain Di can invoke on object Oj.

Different types of rights:

There are different types of rights the files can have. The most common ones are:

1. Read- This is a right given to a process in a domain, which allows it to read the file.
2. Write- Process in domain can write into the file.
3. Execute- Process in domain can execute the file.
4. Print- Process in domain only has access to printer.

Sometimes, domains can have more than one right, i.e. combination of rights mentioned above.

Let us now understand how an access matrix works from the example given below.

	F1	**F2**	**F3**	**Printer**
D1	read		read	
D2				print
D3		read	execute	

	F1	F2	F3	Printer
D4	read write		read write	

Observations of above matrix:

- There are **four domains** and **four objects**– three files(F1, F2, F3) and one printer.
- A process executing in D1 can read files F1 and F3.
- A process executing in domain D4 has same rights as D1 but it can also write on files.
- Printer can be accessed by only one process executing in domain D2.
- A process executing in domain D3 has the right to read file F2 and execute file F3.

Mechanism of access matrix:

The mechanism of access matrix consists of many policies and semantic properties. Specifically, we must ensure that a process executing in domain Di can access only those objects that are specified in row i. Policies of access matrix concerning protection involve which rights should be included in the **(i, j)th entry**. We must also decide the domain in which each process executes. This policy is usually decided by the operating system. The users decide the contents of the access-matrix entries. Association between the domain and processes can be either static or dynamic. Access matrix provides a mechanism for defining the control for this association between domain and processes.

Switch operation: When we switch a process from one domain to another, we execute a switch operation on an object(the domain). We can control domain switching by including domains among the objects of the access matrix. Processes should be able to switch from one domain (Di) to another domain

(Dj) if and only if a switch right is given to access(i, j). This is explained using an example below:

	F1	F2	F3	Printer	D1	D2	D3	D4
D1	read		read			switch		
D2				print			switch	switch
D3		read	execute					
D4	read write		read write		switch			

According to the above matrix, a process executing in domain D2 can switch to domain D3 and D4. A process executing in domain D4 can switch to domain D1 and process executing in domain D1 can switch to domain D2.