**AIMAN COLLEGE OF ARTS & SCIENCE FOR WOMEN,TIRUCHIRAPPALLI**

**Class : IIi B.Sc Computer Science – Vi Semester**

**Subject Title : Operating Systems**

**Subject Code : 16SCCCS8**

**Staff Name : A. Gurshith Begum**

**Designation : Asst. Professor**

IO SUBSYSTEM

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories −

* **Block devices** − A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
* **Character devices** − A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sounds cards etc

Device Controllers

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.

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.



Synchronous vs asynchronous I/O

* **Synchronous I/O** − In this scheme CPU execution waits while I/O proceeds
* **Asynchronous I/O** − I/O proceeds concurrently with CPU execution

Communication to I/O Devices

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

* Special Instruction I/O
* Memory-mapped I/O
* Direct memory access (DMA)

Special Instruction I/O

This uses CPU instructions that are specifically made for controlling I/O devices. These instructions typically allow data to be sent to an I/O device or read from an I/O device.

Memory-mapped I/O

When using memory-mapped I/O, the same address space is shared by memory and I/O devices. The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.



While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished.

The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped IO is used for most high-speed I/O devices like disks, communication interfaces.

Direct Memory Access (DMA)

Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses direct memory access (DMA) hardware to reduce this overhead.

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.



The operating system uses the DMA hardware as follows −

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | Device driver is instructed to transfer disk data to a buffer address X. |
| 2 | Device driver then instruct disk controller to transfer data to buffer. |
| 3 | Disk controller starts DMA transfer. |
| 4 | Disk controller sends each byte to DMA controller. |
| 5 | DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero. |
| 6 | When C becomes zero, DMA interrupts CPU to signal transfer completion. |

Polling vs Interrupts I/O

A computer must have a way of detecting the arrival of any type of input. There are two ways that this can happen, known as **polling** and **interrupts**. Both of these techniques allow the processor to deal with events that can happen at any time and that are not related to the process it is currently running.

Polling I/O

Polling is the simplest way for an I/O device to communicate with the processor. The process of periodically checking status of the device to see if it is time for the next I/O operation, is called polling. The I/O device simply puts the information in a Status register, and the processor must come and get the information.

Most of the time, devices will not require attention and when one does it will have to wait until it is next interrogated by the polling program. This is an inefficient method and much of the processors time is wasted on unnecessary polls.

Compare this method to a teacher continually asking every student in a class, one after another, if they need help. Obviously the more efficient method would be for a student to inform the teacher whenever they require assistance.

Interrupts I/O

An alternative scheme for dealing with I/O is the interrupt-driven method. An interrupt is a signal to the microprocessor from a device that requires attention.

A device controller puts an interrupt signal on the bus when it needs CPU’s attention when CPU receives an interrupt, It saves its current state and invokes the appropriate interrupt handler using the interrupt vector (addresses of OS routines to handle various events). When the interrupting device has been dealt with, the CPU continues with its original task as if it had never been interrupted.

I/O software is often organized in the following layers −

* **User Level Libraries** − This provides simple interface to the user program to perform input and output. For example, **stdio** is a library provided by C and C++ programming languages.
* **Kernel Level Modules** − This provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
* **Hardware** − This layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.

A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on a floppy disk, on a hard disk, or on a CD-ROM, without having to modify the program for each different device.



Device Drivers

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 drivers encapsulate device-dependent code and implement a standard interface in such a way that code contains device-specific register reads/writes. Device driver, is generally written by the device's manufacturer and delivered along with the device on a CD-ROM.

A device driver performs the following jobs −

* To accept request from the device independent software above to it.
* Interact with the device controller to take and give I/O and perform required error handling
* Making sure that the request is executed successfully

How a device driver handles a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.

Interrupt handlers

An interrupt handler, also known as an interrupt service routine or ISR, is a piece of software or more specifically a callback function in an operating system or more specifically in a device driver, whose execution is triggered by the reception of an interrupt.

When the interrupt happens, the interrupt procedure does whatever it has to in order to handle the interrupt, updates data structures and wakes up process that was waiting for an interrupt to happen.

The interrupt mechanism accepts an address ─ a number that selects a specific interrupt handling routine/function from a small set. In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Device-Independent I/O Software

The basic function of the device-independent software is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software. Though it is difficult to write completely device independent software but we can write some modules which are common among all the devices. Following is a list of functions of device-independent I/O Software −

* Uniform interfacing for device drivers
* Device naming - Mnemonic names mapped to Major and Minor device numbers
* Device protection
* Providing a device-independent block size
* Buffering because data coming off a device cannot be stored in final destination.
* Storage allocation on block devices
* Allocation and releasing dedicated devices
* Error Reporting

User-Space I/O Software

These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers. Most of the user-level I/O software consists of library procedures with some exception like spooling system which is a way of dealing with dedicated I/O devices in a multiprogramming system.

I/O Libraries (e.g., stdio) are in user-space to provide an interface to the OS resident device-independent I/O SW. For example putchar(), getchar(), printf() and scanf() are example of user level I/O library stdio available in C programming.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

* **Scheduling** − Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
* **Buffering** − Kernel I/O Subsystem maintains a memory area known as **buffer** that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.
* **Caching** − Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
* **Spooling and Device Reservation** − A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
* **Error Handling** − An operating system that uses protected memory can guard against many kinds of hardware and application errors.

**Magnetic Disk in Computer Architecture-**

In computer architecture,

* Magnetic disk is a storage device that is used to write, rewrite and access data.
* It uses a magnetization process.

**Architecture-**

* The entire disk is divided into **platters**.
* Each platter consists of concentric circles called as **tracks**.
* These tracks are further divided into **sectors** which are the smallest divisions in the disk.



* A **cylinder** is formed by combining the tracks at a given radius of a disk pack.



* There exists a mechanical arm called as **Read / Write head**.
* It is used to read from and write to the disk.
* Head has to reach at a particular track and then wait for the rotation of the platter.
* The rotation causes the required sector of the track to come under the head.
* Each platter has 2 surfaces- top and bottom and both the surfaces are used to store the data.
* Each surface has its own read / write head.



**Disk Performance Parameters-**

The time taken by the disk to complete an I/O request is called as **disk service time** or **disk access time**.

Components that contribute to the service time are-



1. Seek time
2. Rotational latency
3. Data transfer rate
4. Controller overhead
5. Queuing delay

**1. Seek Time-**

* The time taken by the read / write head to reach the desired track is called as **seek time**.
* It is the component which contributes the largest percentage of the disk service time.
* The lower the seek time, the faster the I/O operation.

|  |  |
| --- | --- |
| **Specifications**Seek time specifications include-1. Full stroke
2. Average
3. Track to Track

 **1. Full Stroke-** * It is the time taken by the read / write head to move across the entire width of the disk from the innermost track to the outermost track

 **2. Average-** * It is the average time taken by the read / write head to move from one random track to another.

|  |
| --- |
| Average seek time = 1 / 3 x Full stroke |

 **3. Track to Track-** * It is the time taken by the read-write head to move between the adjacent tracks.
 |

**2. Rotational Latency-**

* The time taken by the desired sector to come under the read / write head is called as **rotational latency**.
* It depends on the rotation speed of the spindle.

|  |
| --- |
| Average rotational latency = 1 / 2 x Time taken for full rotation |

**3. Data Transfer Rate-**

* The amount of data that passes under the read / write head in a given amount of time is called as **data transfer rate**.
* The time taken to transfer the data is called as **transfer time**.

It depends on the following factors-

1. Number of bytes to be transferred
2. Rotation speed of the disk
3. Density of the track
4. Speed of the electronics that connects the disk to the computer

**4. Controller Overhead-**

* The overhead imposed by the disk controller is called as **controller overhead**.
* Disk controller is a device that manages the disk.

**5. Queuing Delay-**

* The time spent waiting for the disk to become free is called as **queuing delay**.

**NOTE-**

|  |
| --- |
| All the tracks of a disk have the same storage capacity. |

**Storage Density-**

* All the tracks of a disk have the same storage capacity.
* This is because each track has different storage density.
* Storage density decreases as we from one track to another track away from the center.

Thus,

* Innermost track has maximum storage density.
* Outermost track has minimum storage density.

**Important Formulas-**

**1. Disk Access Time-**

Disk access time is calculated as-

|  |
| --- |
| Disk access time= Seek time + Rotational delay + Transfer time + Controller overhead + Queuing delay |

**2. Average Disk Access Time-**

Average disk access time is calculated as-

|  |
| --- |
| Average disk access time= Average seek time + Average rotational delay + Transfer time + Controller overhead + Queuing delay |

**3. Average Seek Time-**

Average seek time is calculated as-

|  |
| --- |
| Average seek time= 1 / 3 x Time taken for one full stroke |

**Alternatively**,

If time taken by the head to move from one track to adjacent track = t units and there are total k tracks, then-

Average seek time

= { Time taken to move from track 1 to track 1 + Time taken to move from track 1 to last track } / 2

= { 0 + (k-1)t } / 2

= (k-1)t / 2

**4. Average Rotational Latency-**

Average rotational latency is calculated as-

|  |
| --- |
| Average rotational latency= 1 / 2 x Time taken for one full rotation |

Average rotational latency may also be referred as-

* Average rotational delay
* Average latency
* Average delay

**5. Capacity Of Disk Pack-**

Capacity of a disk pack is calculated as-

|  |
| --- |
| Capacity of a disk pack= Total number of surfaces x Number of tracks per surface x Number of sectors per track x Storage capacity of one sector |

**6. Formatting Overhead-**

Formatting overhead is calculated as-

|  |
| --- |
| Formatting overhead= Number of sectors x Overhead per sector |

**7. Formatted Disk Space-**

Formatted disk space also called as usable disk space is the disk space excluding formatting overhead.

It is calculated as-

|  |
| --- |
| Formatted disk space= Total disk space or capacity – Formatting overhead |

**8. Recording Density Or Storage Density-**

Recording density or Storage density is calculated as-

|  |
| --- |
| Storage density of a track= Capacity of the track / Circumference of the track |

From here, we can infer-

Storage density of a track ∝ 1 / Circumference of the track

**9. Track Capacity-**

Capacity of a track is calculated as-

|  |
| --- |
| Capacity of a track= Recording density of the track x Circumference of the track |

**10. Data Transfer Rate-**

Data transfer rate is calculated as-

|  |
| --- |
| Data transfer rate= Number of heads x Bytes that can be read in one full rotation x Number of rotations in one second |

**OR**

|  |
| --- |
| Data transfer rate= Number of heads x Capacity of one track x Number of rotations in one second |

**11. Tracks Per Surface-**

Total number of tracks per surface is calculated as-

|  |
| --- |
| Total number of tracks per surface= (Outer radius – Inner radius) / Inter track gap |

**Points to Remember-**

* The entire disk space is not usable for storage because some space is wasted in formatting.
* When rotational latency is not given, use average rotational latency for solving numerical problems.
* When seek time is not given, use average seek time for solving numerical problems.
* It is wrong to say that as we move from one track to another away from the center, the capacity increases.
* All the tracks have same storage capacity.

To gain better understanding about magnetic disk-

**[Watch this Video Lecture](https://youtu.be/ZjMwUhapSEM%22%20%5Ct%20%22_blank)**

**Next Article-** **[Practice Problems On Magnetic Disk](https://www.gatevidyalay.com/magnetic-disk-practice-problems-coa/)**

Get more notes and other study material of **[Computer Organization and Architecture](https://www.gatevidyalay.com/computer-organization-architecture/)**.

Watch video lectures by visiting our YouTube channel **[LearnVidFun](https://www.youtube.com/learnvidfun%22%20%5Ct%20%22_blank)**.

**Summary**



**Article Name**

Magnetic Disk in Computer Architecture

**Description**

Magnetic Disk is a storage device. Disk performance parameters- Seek time, Rotational Latency, Data Transfer Rate. Disk Formulas- Seek time Formula, Rotational Latency Formula etc.

**Author**

**What is File System?**

A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program.

In general, a file is a sequence of bits, bytes, or records whose meaning is defined by the file creator and user. Every File has a logical location where they are located for storage and retrieval.

## Objective of File management System

Here are the main objectives of the file management system:

* It provides I/O support for a variety of storage device types.
* Minimizes the chances of lost or destroyed data
* Helps OS to standardized I/O interface routines for user processes.
* It provides I/O support for multiple users in a multiuser systems environment.

## Properties of a File System

Here, are important properties of a file system:

* Files are stored on disk or other storage and do not disappear when a user logs off.
* Files have names and are associated with access permission that permits controlled sharing.
* Files could be arranged or more complex structures to reflect the relationship between them.

## File structure

A File Structure needs to be predefined format in such a way that an operating system understands . It has an exclusively defined structure, which is based on its type.

Three types of files structure in OS:

* A text file: It is a series of characters that is organized in lines.
* An object file: It is a series of bytes that is organized into blocks.
* A source file: It is a series of functions and processes.

## File Attributes

A file has a name and data. Moreover, it also stores meta information like file creation date and time, current size, last modified date, etc. All this information is called the attributes of a file system.

Here, are some important File attributes used in OS:

* **Name:** It is the only information stored in a human-readable form.
* **Identifier**: Every file is identified by a unique tag number within a file system known as an identifier.
* **Location:** Points to file location on device.
* **Type:** This attribute is required for systems that support various types of files.
* **Size**. Attribute used to display the current file size.
* **Protection**. This attribute assigns and controls the access rights of reading, writing, and executing the file.
* **Time, date and security:** It is used for protection, security, and also used for monitoring

## File Type

It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS\_DOS and UNIX has the following type of files:

### Character Special File

It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

### Ordinary files

* These types of files stores user information.
* It may be text, executable programs, and databases.
* It allows the user to perform operations like add, delete, and modify.

### Directory Files

* Directory contains files and other related information about those files. Its basically a folder to hold and organize multiple files.

### Special Files

* These files are also called device files. It represents physical devices like printers, disks, networks, flash drive, etc.

## Functions of File

* Create file, find space on disk, and make an entry in the directory.
* Write to file, requires positioning within the file
* Read from file involves positioning within the file
* Delete directory entry, regain disk space.
* Reposition: move read/write position.

Commonly used terms in File systems

Field:

This element stores a single value, which can be static or variable length.

DATABASE:

Collection of related data is called a database. Relationships among elements of data are explicit.

FILES:

Files is the collection of similar record which is treated as a single entity.

RECORD:

A Record type is a complex data type that allows the programmer to create a new data type with the desired column structure. Its groups one or more columns to form a new data type. These columns will have their own names and data type.

**Operation performed on 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 are:**

* **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 same name for two files.
* **Grouping problem:** Users cannot group files according to their need.



**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 same file name for different user.
* Searching is efficient in this method.



**TREE-STRUCTURED DIRECTORY :**
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 Access Methods

File access is a process that determines the way that files are accessed and read into memory. Generally, a single access method is always supported by operating systems. Though there are some operating system which also supports multiple access methods.

Three file access methods are:

* Sequential access
* Direct random access
* Index sequential access

**FILE ALLOCATION METHODS**
**1. 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.


**Disadvantage**

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

**2. 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. Increase in file size is always possible if 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 file.

**Disadvantage:**

* Internal fragmentation exists in last disk block of 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 sequencial access of files.

**3. 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. 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.


**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 correspond 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.

1. **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.



## Space Allocation

In the Operating system, files are always allocated disk spaces.

Three types of space allocation methods are:

* Linked Allocation
* Indexed Allocation
* Contiguous Allocation

### Contiguous Allocation

In this method,

* Every file users a contiguous address space on memory.
* Here, the OS assigns disk address is in linear order.
* In the contiguous allocation method, external fragmentation is the biggest issue.

### Linked Allocation

In this method,

* Every file includes a list of links.
* The directory contains a link or pointer in the first block of a file.
* With this method, there is no external fragmentation
* This File allocation method is used for sequential access files.
* This method is not ideal for a direct access file.

### Indexed Allocation

In this method,

* Directory comprises the addresses of index blocks of the specific files.
* An index block is created, having all the pointers for specific files.
* All files should have individual index blocks to store the addresses for disk space.

## File Directories

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

Following is the information which is maintained in a directory:

* **Name** The name which is displayed to the user.
* **Type**: Type of the directory.
* **Position**: Current next-read/write pointers.
* **Location**: Location on the device where the file header is stored.
* **Size** : Number of bytes, block, and words in the file.
* **Protection**: Access control on read/write/execute/delete.
* **Usage**: Time of creation, access, modification

## File types- name, extension

|  |  |  |
| --- | --- | --- |
| **File Type** | **Usual extension** | **Function** |
| Executable | exe, com, bin or none | ready-to-run machine- language program |
| Object | obj, o | complied, machine language, not linked |
| Source code | c. p, pas, 177, asm, a | source code in various languages |
| Batch | bat, sh | Series of commands to be executed |
| Text | txt, doc | textual data documents |
| Word processor | doc,docs, tex, rrf, etc. | various word-processor formats |
| Library | lib, h | libraries of routines |
| Archive | arc, zip, tar | related files grouped into one file, sometimes compressed. |

### Summary:

* A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes.
* It provides I/O support for a variety of storage device types.
* Files are stored on disk or other storage and do not disappear when a user logs off.
* A File Structure needs to be predefined format in such a way that an operating system understands it.
* File type refers to the ability of the operating system to differentiate different types of files like text files, binary, and source files.
* Create find space on disk and make an entry in the directory.
* Indexed Sequential Access method is based on simple sequential access
* In Sequential Access method records are accessed in a certain pre-defined sequence
* The random access method is also called direct random access
* Three types of space allocation methods are:
	+ Linked Allocation
	+ Indexed Allocation
	+ Contiguous Allocation
* Information about files is maintained by Directories