File Management: Concept of File, Access methods, File types, File operation, Directory structure, File System structure, Allocation methods (contiguous, linked, indexed), Free-space management (bit vector, linked list, grouping), directory implementation (linear list, hash table), efficiency and performance.
I/O Hardware: I/O devices, Device controllers, Direct memory access Principles of I/O Software: Goals of Interrupt handlers, Device drivers, Device independent I/O software.

File System

File Concept:

Computers can store information on various storage media such as, magnetic disks, magnetic tapes, optical disks. The physical storage is converted into a logical storage unit by operating system. The logical storage unit is called FILE. A file is a collection of similar records. A record is a collection of related fields that can be treated as a unit by some application program. A field is some basic element of data. Any individual field contains a single value. A data base is collection of related data.

Student	Marks	Marks	Fail/Pas
KUMA	85	86	P
LAKSH	93	92	P

DATA FILE

Student name, Marks in sub1, sub2, Fail/Pass is fields. The collection of fields is called a RECORD. **RECORD:**

LAKSH	93	92	P

Collection of these records is called a data file.

FILE ATTRIBUTES :

1. Name : A file is named for the convenience of the user and is referred by its name. A name is usually a string of characters.
2. Identifier : This unique tag, usually a number ,identifies the file within the file system.
3. Type : Files are of so many types. The type depends on the extension of the file.

Example:
.exe Executable file
.obj Object file
.src Source file

4. Location : This information is a pointer to a device and to the location of the file on that device.

5. Size : The current size of the file (in bytes, words,blocks).
6. Protection : Access control information determines who can do reading, writing, executing and so on.
7. Time, Date, User identification : This information may be kept for creation, last modification,last use.

FILE OPERATIONS

1. Creating a file : Two steps are needed to create a file. They are:
- *Check whether the space is available ornot.*
- *If the space is available then made an entry for the new file in the directory. The entry includes name of the file, path of the file,etc…*

2. Writing a file : To write a file, we have to know 2 things. One is name of the file and second is the information or data to be written on the file, the system searches the entired given location for the file. If the file is found, the system must keep a write pointer to the location in the file where the next write is to take place.
3. Reading a file : To read a file, first of all we search the directories for the file, if the file is found, the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated.
4. Repositioning within a file : The directory is searched for the appropriate entry and the current file position pointer is repositioned to a given value. This operation is also called file seek.
5. Deleting a file : To delete a file, first of all search the directory for named file, then released the file space and erase the directoryentry.
6. Truncating a file : To truncate a file, remove the file contents only but, the attributes are as itis.

FILE TYPES: The name of the file split into 2 parts. One is name and second is Extension. The file type is depending on extension of the file.

File Type	Extension	Purpose
Executable	.exe .com .bin	Ready to run (or) ready to run machine
Source code	.c .cpp .asm	Source code in various languages.
Object	.obj .o	Compiled, machine
Batch	.bat .sh	Commands to the command

Text	.txt .doc	Textual data, documents
Word processor	.doc .wp .rtf	Various word processor formats
Library	.lib .dll	Libraries of routines for
Print or View	.pdf .jpg	Binary file in a format for
Archive	.arc .zip	Related files grouped into a
Multimedia	.mpeg .mp3 .avi	Binary file containing audio or audio/video

FILE STRUCTURE

File types also can be used to indicate the internal structure of the file. The operating system requires that an executable file have a specific structure so that it can determine where in memory to load the file and what the location of the first instruction is. If OS supports multiple file structures, the resulting size of OS is large. If the OS defines 5 different file structures, it needs to contain the code to support these file structures. All OS must support at least one structure that of an executable file so that the system is able to load and run programs.

INTERNAL FILE STRUCTURE

In UNIX OS, defines all files to be simply stream of bytes. Each byte is individually addressable by its offset from the beginning or end of the file. In this case, the logical record size is 1 byte. The file system automatically packs and unpacks bytes into physical disk blocks, say 512 bytes per block.

The logical record size, physical block size, packing determines how many logical records are in each physical block. The packing can be done by the user's application program or OS. A file may be considered a sequence of blocks. If each block were 512 bytes, a file of 1949 bytes would be allocated 4 blocks (2048 bytes). The last 99 bytes

would be wasted. It is called internal fragmentation all file systems suffer from internal fragmentation, the larger the block size, the greater the internal fragmentation.

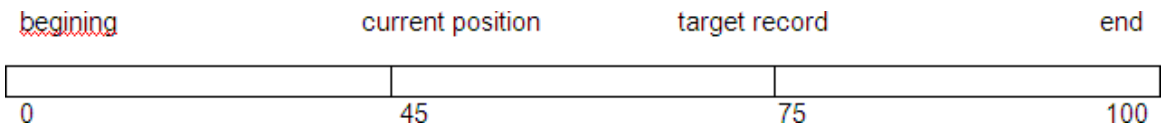
FILE ACCESS METHODS

Files stores information, this information must be accessed and read into computer memory. There are so many ways that the information in the file can be accessed.

1. Sequential file access:

Information in the file is processed in order i.e. one record after the other.
Magnetic tapes are supporting this type of file accessing.

Eg : A file consisting of 100 records, the current position of read/write head is 45th record, suppose we want to read the 75th record then, it access sequentially from 45, 46, 47
…….. 74, 75. So the read/write head traverse all the records between 45 to 75.



2. Direct access:

Direct access is also called relative access. Here records can read/write randomly without any order. The direct access method is based on a disk model of a file, because disks allow random access to any file block.

Eg : A disk containing of 256 blocks, the position of read/write head is at 95th block. The block is to be read or write is 250th block. Then we can access the 250th block directly without any restrictions.

Eg : CD consists of 10 songs, at present we are listening song 3, If we want to listen song 10, we can shift to 10.

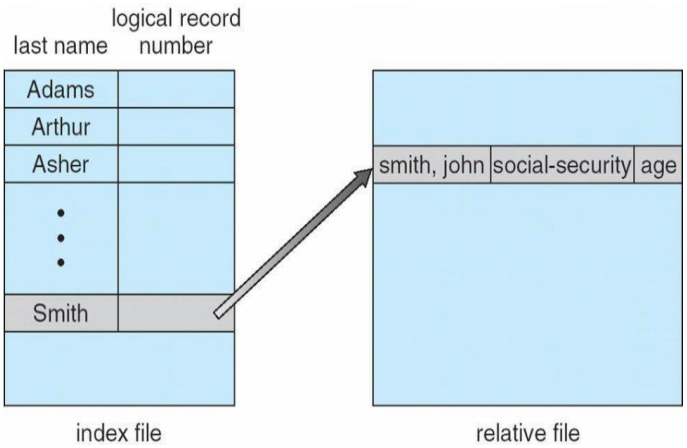
3. Indexed Sequential File access

The main disadvantage in the sequential file is, it takes more time to access a Record .Records are organized in sequence based on a key field.

Eg :

A file consisting of 60000 records,the master index divide the total records into 6 blocks, each block consisiting of a pointer to secondary index.The secondary index divide the 10,000 records into 10 indexes.Each index consisting of a pointer to its orginal

location.Each record in the index file consisting of 2 field, A key field and a pointer field.



DIRECTORY STRUCTURE

Sometimes the file system consisting of millions of files,at that situation it is very hard to manage the files. To manage these files grouped these files and load one group into one partition.

Each partition is called a directory .a directory structure provides a mechanism for organizing many files in the file system.

OPERATION ON THE DIRECTORIES :

1. Search for a file : Search a directory structure for requiredfile.

2. createafile : New files need to be created, added to thedirectory.

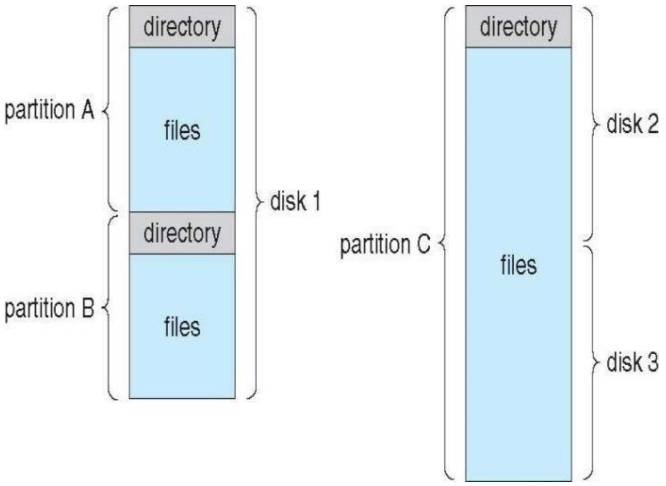
3. Deleteafile : When a file is no longer needed,we want to remove it fromthe

directory.

4. List adirectory : We can know the list of files in thedirectory.

5. Renameafile : When ever we need to change the name of the file,wecanchange thename.

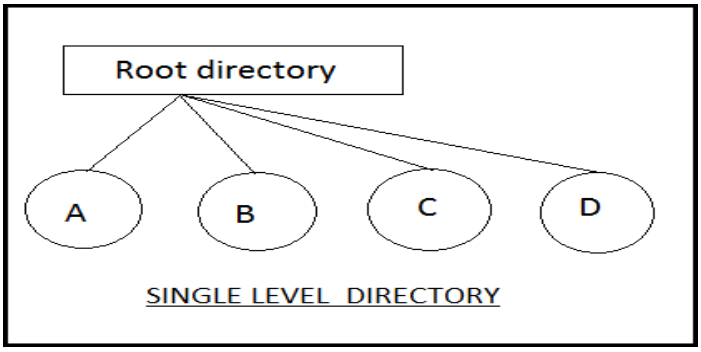
6. Traverse the file system : We need to access every directory and every file with in a directory structure we can traverse the file system



The various directory structures

1. Single level directory:

The directory system having only one directory,it consisting of all files some times it is said to be root directory.



E.g :- Here directory containing 4 files (A,B.C,D).the advantage of the scheme is its simplicity and the ability to locate files quickly.The problem is different users may accidentally use the same names for their files.

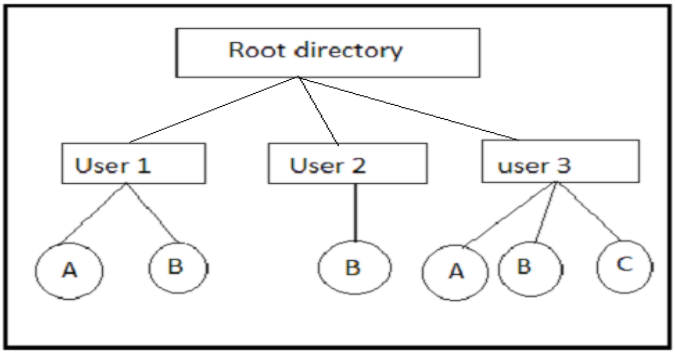
E.g :- If user 1 creates a files caled sample and then later user 2 to creates a file called sample,then user2's file will overwrite user 1 file.Thats why it is not used in the multi user system.

2. Two level directory:

The problem in single level directory is different user may be accidentally use

the same name for their files. To avoid this problem each user need a private directory,

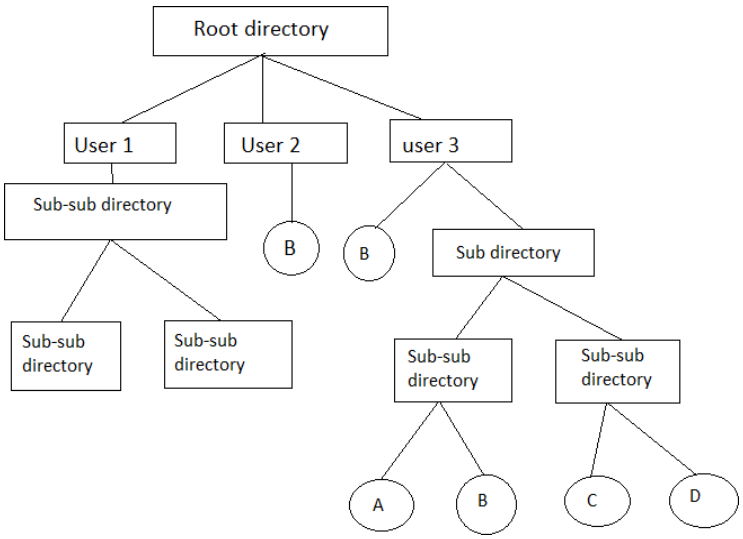
Names chosen by one user don't interfere with names chosen by a different user.



Root directory is the first level directory.user 1,user2,user3 are user level of directory A,B,C are files.

3. Tree structured directory:

Two level directory eliminates name conflicts among users but it is not satisfactory for users with a large number of files.To avoid this create the sub- directory and load the same type of files into the sub-directory.so, here each can have as many directories are needed.



There are 2 types of path

1. Absoulte path
2. Relative path

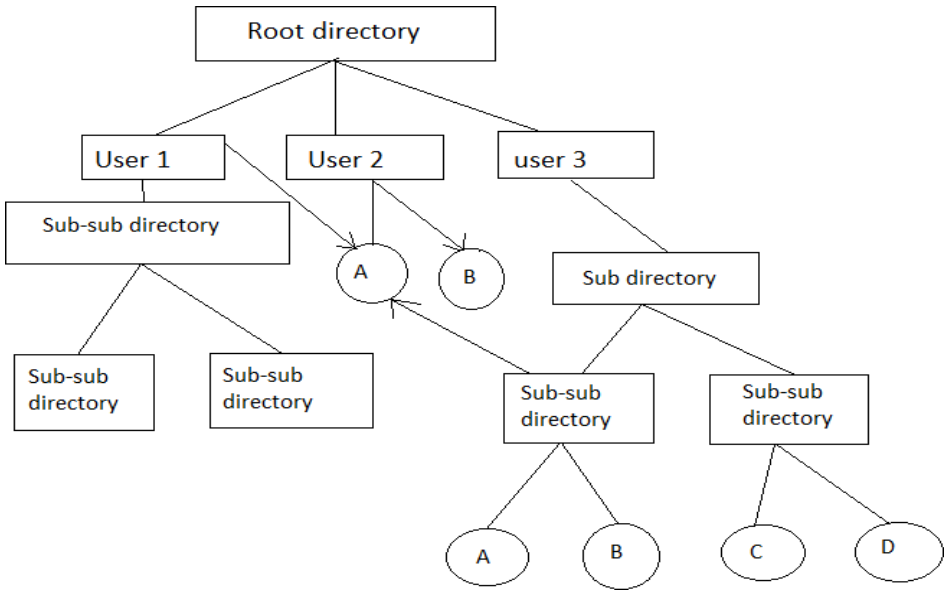
Absoulte path : Begging with root and follows a path down to specified files giving directory, directory name on the path.
Relative path : A path from current directory.

4. Acyclic graphdirectory

Multiple users are working on a project, the project files can be stored in a comman sub-directory of the multiple users. This type of directory is called acyclic graph directory .The common directory will be declared a shared directory. The graph contain no cycles with shared files, changes made by one user are made visible to other users.A file may now have multiple absolute paths. when shared directory/file is deleted, all pointers to the directory/ files also to be removed.

5. General graph directory:

When we add links to an existing tree structured directory, the tree structure is destroyed, resulting is a simple graph structure.

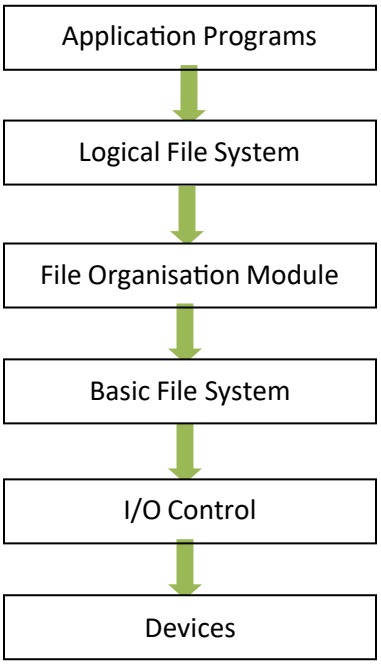


Advantages :- Traversing is easy. Easy sharing is possible.

File system structure:

Disk provides the bulk of secondary storage on which a file system is maintained. They have 2 characteristics that make them a convenient medium for storing multiple files.

1. A disk can be rewritten in place. It is possible to read a block from the disk, modify the block, and write it back into same place.
2. A disk can access directly any block of information it contains.



I/O Control: consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. The device driver writes specific bit patterns to special locations in the I/O controller's memory to tell the controller which device location to act on and what actions to take.
The Basic File System needs only to issue commands to the appropriate device driver to read and write physical blocks on the disk. Each physical block is identified by its numeric disk address (Eg. Drive 1, cylinder 73, track2, sector 10).

The File Organization Module knows about files and their logical blocks and physical blocks. By knowing the type of file allocation used and the location of the file, file organization module can translate logical block address to physical addresses for the basic file system to transfer. Each file's logical blocks are numbered from 0 to n. so, physical blocks containing the data usually do not match

the logical numbers. A translation is needed to locate each block.

The Logical File System manages all file system structure except the actual data (contents of file). It maintains file structure via file control blocks. A file control block (inode in Unix file systems) contains information about the file, ownership, permissions, location of the file contents.

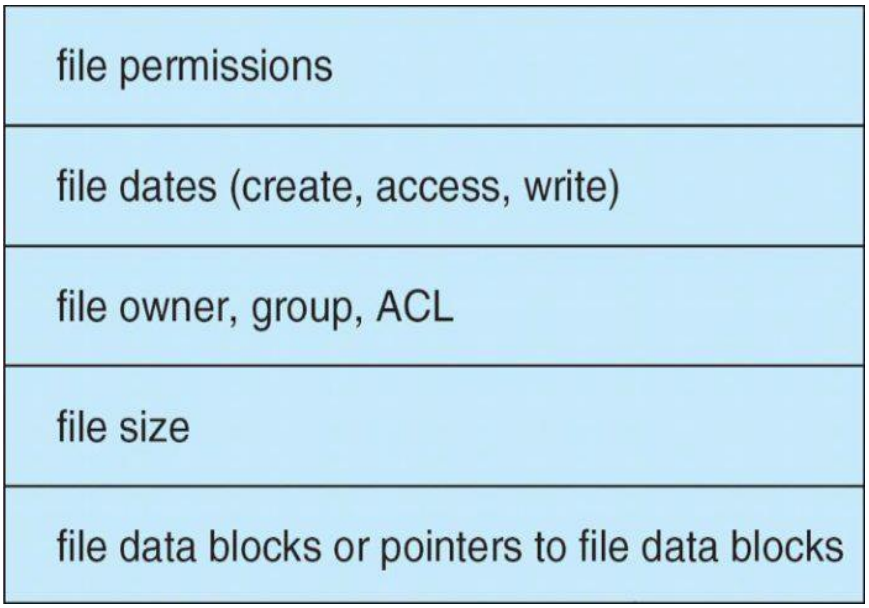
File System Implementation:

Overview:

A Boot Control Block (per volume) can contain information needed by the system to boot an OS from that volume. If the disk does not contain an OS, this block can be empty.

A Volume Control Block (per volume) contains volume (or partition) details, such as number of blocks in the partition, size of the blocks, a free block, count and free block pointers, free FCB count, FCB pointers.

A Typical File Control Block



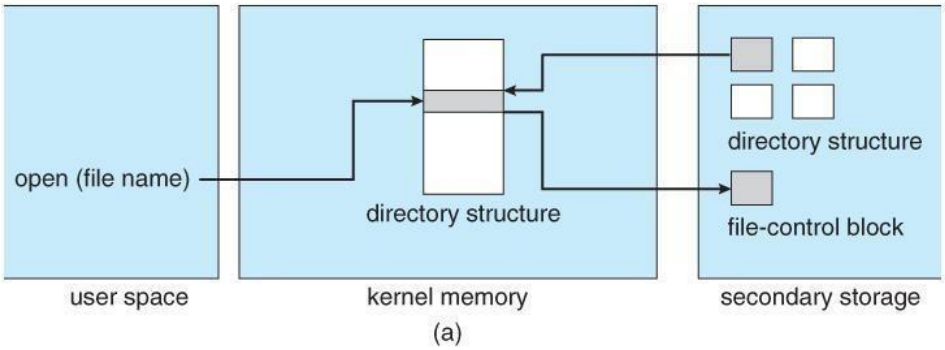
A Directory Structure (per file system) is used to organize the files. A PER-FILE FCB contains many details about the file.
A file has been created; it can be used for I/O. First, it must be opened. The open( ) call passes a file name to the logical file system. The open( ) system call First searches

the system wide open file table to see if the file is already in use by anotherprocess. If it is ,a *per process open file table* entry is created pointing to the existing*system wide open file table.* If the file is not already open, the directory structure is searched for the given file name. Once the file is found, FCB is copied into a system

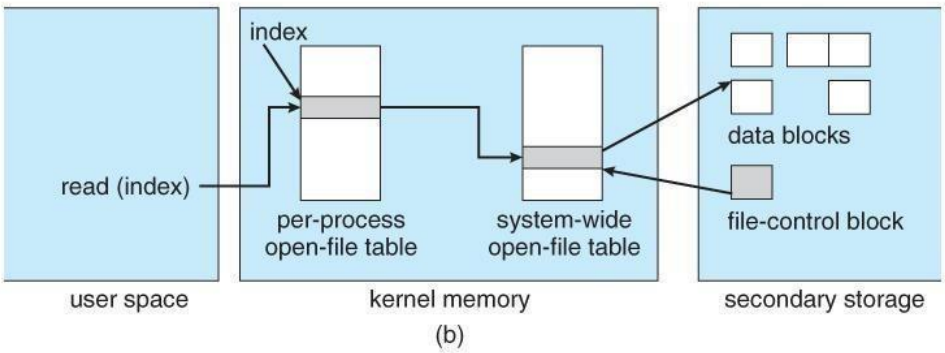
wide open file table in memory. This table not only stores the FCB but also tracks the number of processes that have the file open.

Next, an entry is made in the per – process open file table, with the pointer to the entry in the system wide open file table and some other fields. These are the fields include a pointer to the current location in the file (for the next read/write operation) and the access mode in which the file is open. The open () call returns a pointer to the appropriate entry in the per-process file system table. All file operations are preformed via this pointer. When a process closes the file the per- process table entry is removed. And the system wide entry open count is decremented. When all users that have opened the file close it, any updated metadata is copied back to the disk base directory structure. System wide open file table entry is removed.

System wide open file table contains a copy of the FCB of each open file, other information. Per process open file table, contains a pointerto the appropriate entry in the system wide open file table, other information.



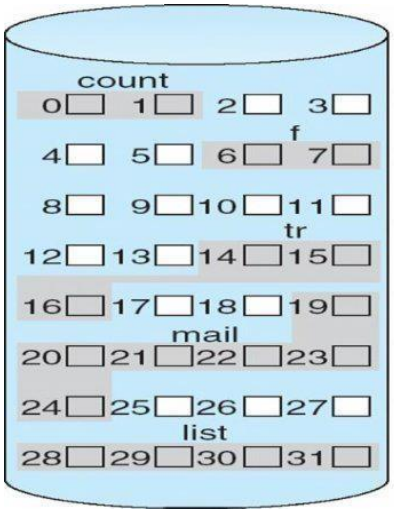
(a)



(b)

Allocation Methods – Contiguous

An allocation method refers to how disk blocks are allocated for files:
Contiguous allocation – each file occupies set of contiguous blocks o Best performance in most cases
- o Simple – only starting location (block #) and length (number ofblocks) are required
- o Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line** (**downtime**) or **on-line**

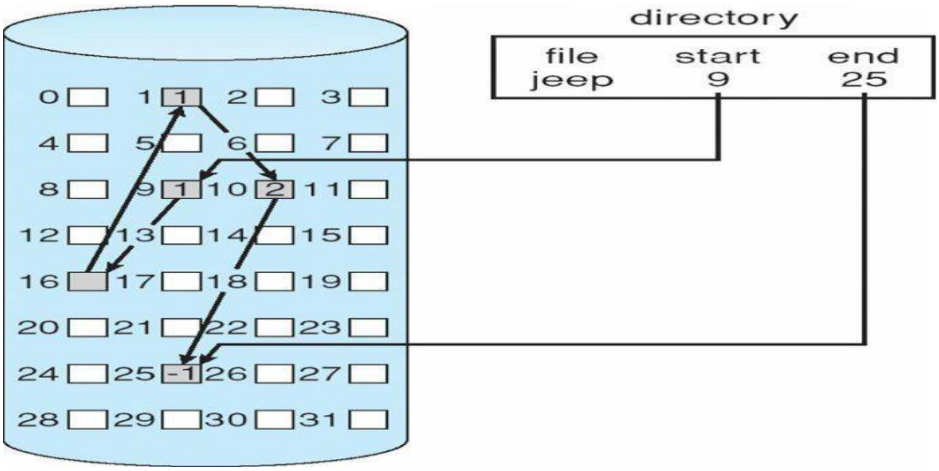


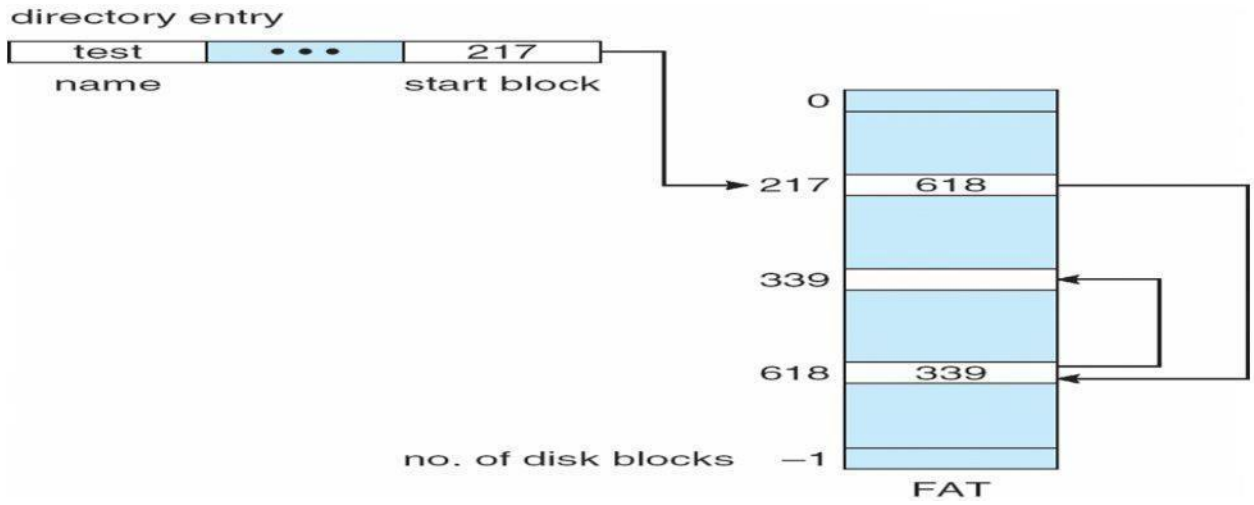
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked

Linked allocation – each file a linked list
of blocks o File ends at nil pointer
- o No external fragmentation
- o Each block contains pointer to next block
- o No compaction, external fragmentation
- o Free space management system called when new block needed
- o Improve efficiency by clustering blocks into groups but increases internal fragmentation
- o Reliability can be a problem
- o Locating a block can take many I/Os and disk seeks FAT (File Allocation Table) variation
- o Beginning of volume has table, indexed by block number
- o Much like a linked list, but faster on disk and cacheable

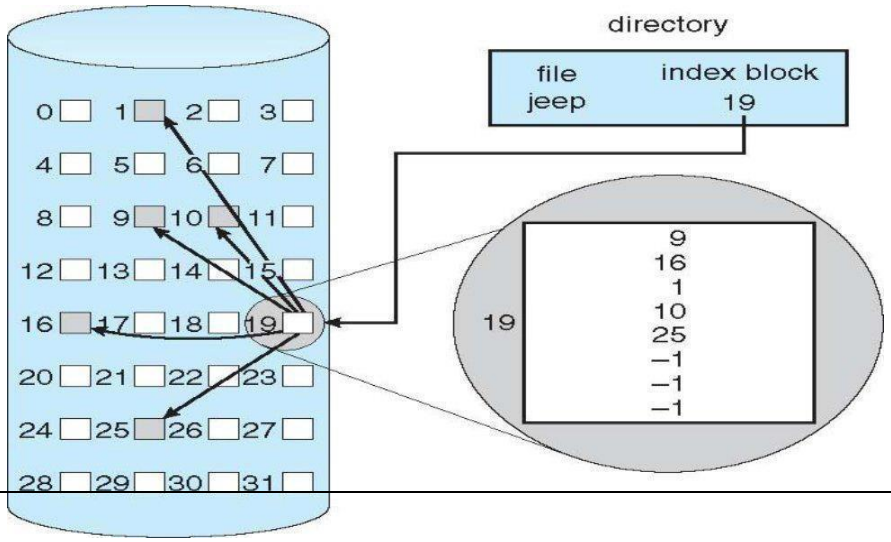




File-Allocation Table

Indexed allocation

- Each file has its own **index block**(s) of pointers to its data blocks



Free-Space Management

File system maintains **free-space list** to track available blocks/clusters Linked list (free list)

- Cannot get contiguous space easily
- No waste of space
- No need to traverse the entire list

1. **Bitmap or Bit vector –**
A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: *0 indicates that the block is allocated* and 1 indicates a free block. The given instance of disk blocks on the disk in *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

Advantages –

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

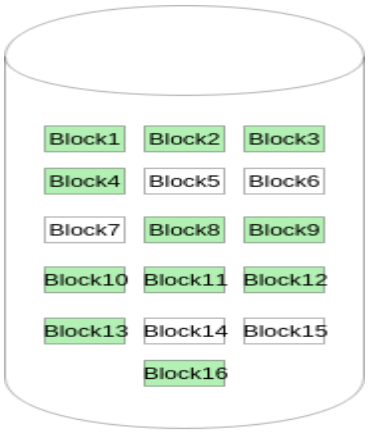
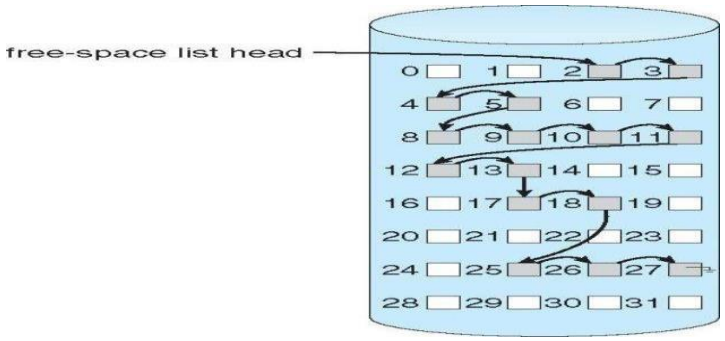


Figure - 1

Linked Free Space List on Disk



In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

Grouping

Modify linked list to store address of next *n-1* free blocks in first free block, plus a pointer to next block that contains free-block-pointers (like this one).
An **advantage** of this approach is that the addresses of a group of free disk blocks can be found easily

Counting

Because space is frequently contiguously used and freed, with contiguous- allocation allocation, extents, or clustering.
Keep address of first free block and count of following free blocks. Free space list then has entries containing addresses and counts.

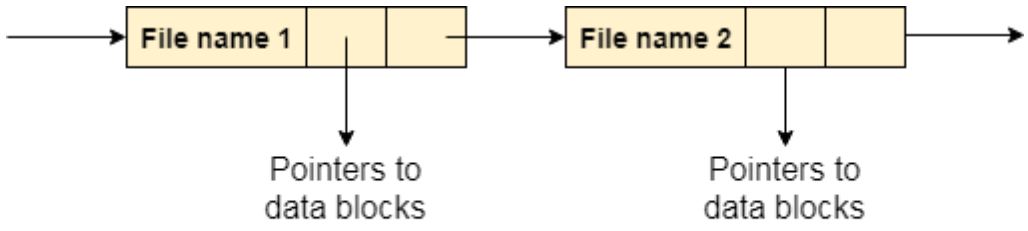
Directory Implementation

1. Linear List

In this algorithm, all the files in a directory are maintained as singly lined list. Each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.

Characteristics

1. When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not. In case, it doesn't exist, the file can be created at the beginning or at the end. Therefore, searching for a unique name is a big concern because traversing the whole list takes time.
2. The list needs to be traversed in case of every operation (creation, deletion, updating, etc) on the files therefore the systems become inefficient.



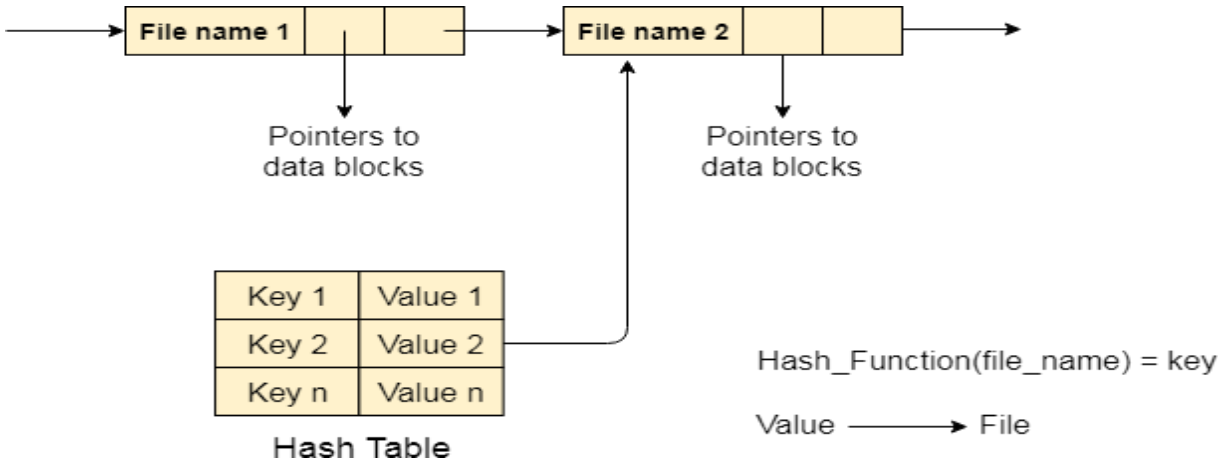
Linear List

2. Hash Table

To overcome the drawbacks of singly linked list implementation of directories, there is an alternative approach that is hash table. This approach suggests to use hash table along with the linked lists.
A key-value pair for each file in the directory gets generated and stored in the hash table. The key can be

determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

Now, searching becomes efficient due to the fact that now, entire list will not be searched on every operating. Only hash table entries are checked using the key and if an entry found then the corresponding file will be fetched using the value.



Efficiency and Performance

Efficiency dependent on:

- Disk allocation and directory algorithms
- Types of data kept in file's directory entry

Performance

- Disk cache – separate section of main memory for frequently used blocks
- free-behind and read-ahead – techniques to optimize sequential access
- improve PC performance by dedicating section of memory as virtual disk, or RAM disk

I/O Hardware: I/O devices

Input/output devices are the devices that are responsible for the input/output operations in a computer system.

Basically there are following two types of input/output devices:

- Block devices
- Character devices

Block Devices

A block device stores information in block with fixed-size and own-address.

It is possible to read/write each and every block independently in case of block device.

In case of disk, it is always possible to seek another cylinder and then wait for required block to rotate under head without mattering where the arm currently is. Therefore, disk is a block addressable device.

Character Devices

A character device accepts/delivers a stream of characters without regarding to any block structure.

Character device isn't addressable.

Character device doesn't have any seek operation.

There are too many character devices present in a computer system such as printer, mice, rats, network interfaces etc. These four are the common character devices.

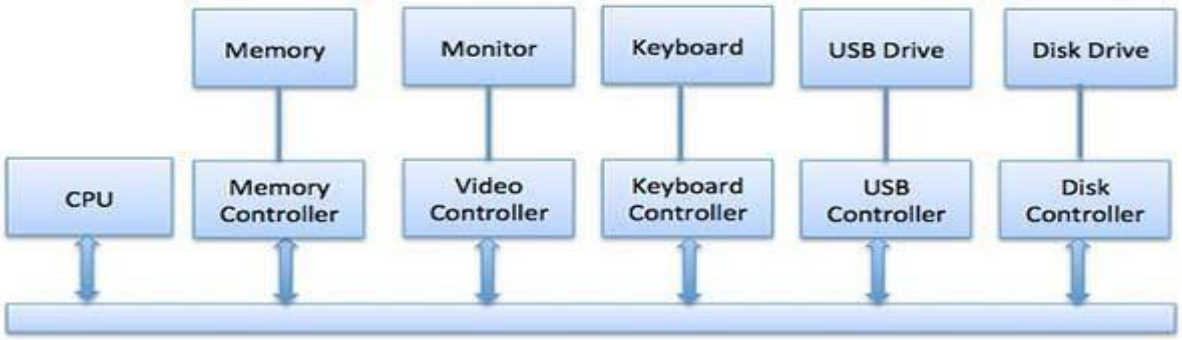
Device Controllers

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.

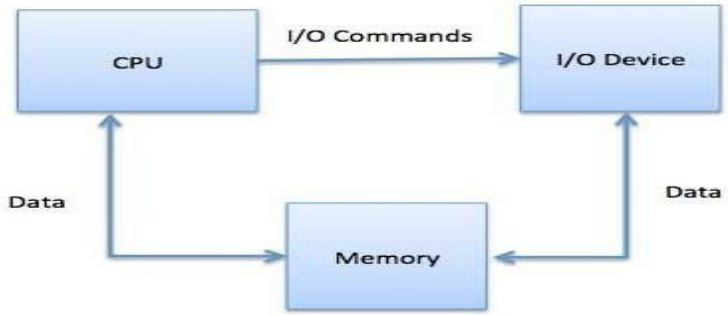
- Special Instruction I/O
- Memory-mapped I/O
- Direct memory access (DMA)

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.

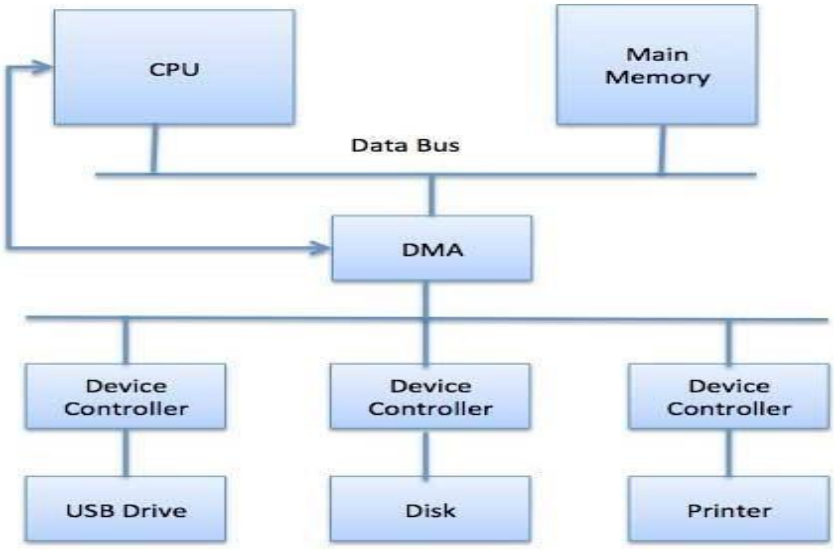
The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped IO is used for most high-speed I/O devices like disks, communication interfaces.

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.



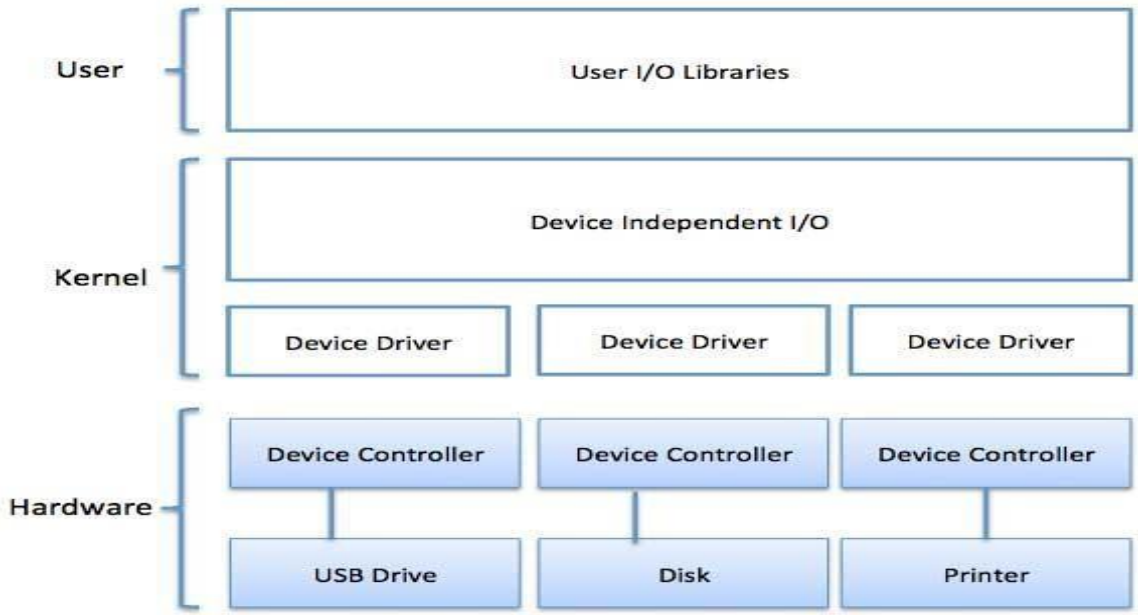
The operating system uses the DMA hardware as follows −

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.

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.



DeviceDrivers

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.

Interrupthandlers

An interrupt handler, also known as an interrupt service routine or ISR, is a piece of software or more specifically a callback functions 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 architecture, 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-IndependentI/OSoftware

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 commonamong 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-SpaceI/OSoftware

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.

KernelI/OSubsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of theservices 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 whilethey are transferred between two devices or between a device with an application operation. Bufferingis 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.