UNIT - IV

File & I/O Systems

File concept:-

Def:-

File is defined as a collection of related info usually stored on a Secondary Storage.

1) **File naming:-**

A File can be named just for identifi--cation where we name a file, its very easy to access.

File name is a group of characters. Some languages treat lower case letters & upper case letters as different.

2) **File Attributes:-**

It is nothing but the properties of a file. The various attributes of a file are:-

a) Type:- This give the Type of file.

b) Creator:- The person who creates the file.

c) owner:- The person who uses the file.

d) Size:- About size of the given file.

e) Time of Creation:- when the file was created time.

f) Time of modification:- when the file was modified time.

g) Location:- where the file is located spot?

h) Identifier:- An unique tag no. assigned by operating system O.S.

i) Date of creation:- when the file was exactly created.

j) Status:- Status of a line.

c) File operations:-

The different operations performed on a file are called file operations.

The various operations are:-

a) Creating a file:- A file is created.
b) opening a file:- A file is opened.
c) Closing a file:- A file is closed.
d) Reading a file:- A file is read.
e) writing a file:- (A) writing content into the file.
f) Deleting a file:- A file is deleted.
g) Copying a file:- Copying the data of a file.
h) updating:- modifying the file.
i) Appending:- Data is added at the last of file.
j) Repositioning a file:- Getting a file back to it's original position.
k) Truncating a file:- Cutting a file.

Regular File:-

d) File Types:

→ Regular

→ Direc

→ Char

Ex:

→ Block

e) File

→ By

→ P

→ T

1) File Types:-

Into 4 types:-

-) Regular file:- User Info exists

-) Directory:- It contains System files.

-) character Special file:- These files contain Info related to I/o devices.

Ex:- Keyboard, printer.

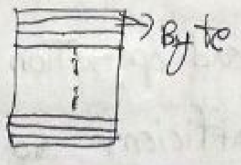
-) Block Special file:-
It contains info regarding block devices.

Ex:- magnetic disk.

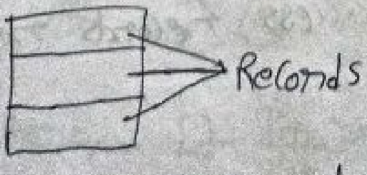
2) File structures:-

In 3 different ways.

-) Byte Sequence:- A Small Size file can be Stored.

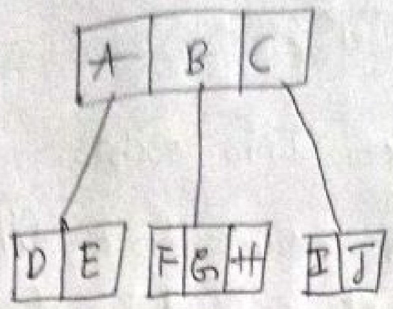


-) Record Sequence:- A medium Size file can be Stored interms of records.



-) Tree:- A large & very large Sized file

Can be stored.



=) File access methods:-

of 2 types:-

a) Sequential Access method:-

Records of file can be accessed (Read/write) in sequential order only.

Ex:- 5 records are there, the 5th record can be read only after reading 4 records.

Ex:- magnetic Tape

R_0	R_1	R_2	R_3	R_4	R_5	R_6

Advantages:-

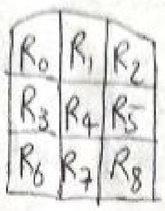
=) 1 read operation, 1 write operation is enough/sufficient to read, write all the records respectively, very simple method.

Disadvantages:-

=) Time consuming process, we can't access records randomly.

Random Access method:-

Any record can be accessed, when it is available in any part of the memory.



magnetic disk/drum.

Advantages:-

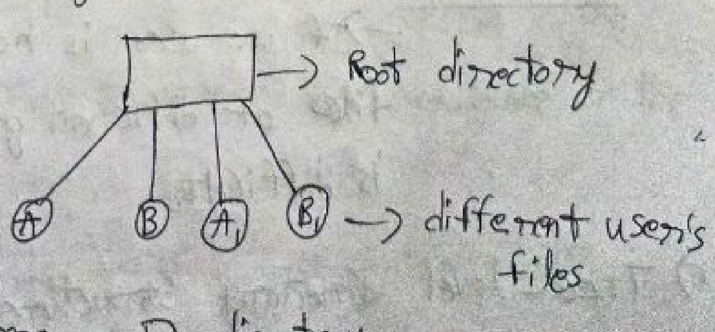
=> It takes less time to write/read any record. Records can be accessed randomly.

=> Directory Structure:-

A directory can be structured in 3 different ways.

a) one-level directory structure:-

In this structure, only one directory is available which is called "root directory". If ~~a~~ contains different users ~~have~~ with different files, all the files will be stored in the single directory only.

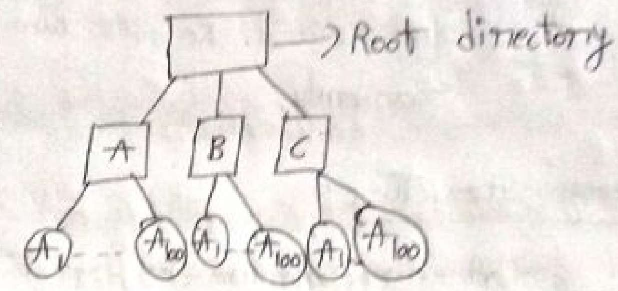


where, □ - directory
○ - file

Advantages:-
→ Easy to implement, All files are available in one place,

Disadvantages:-
→ Identifying different users's with different files is difficult, Security is less.

b) Two-level directory Structure:-



⇒ For each user, a separate directory can be created, So that different user files are Stored under different directories which leads to easy identification.

Advantages:-
→ Easy identification of data

Disadvantages:-
→ If one user is having different files of different groups. Identification is difficult.

c) Tree-level directory Structure:-

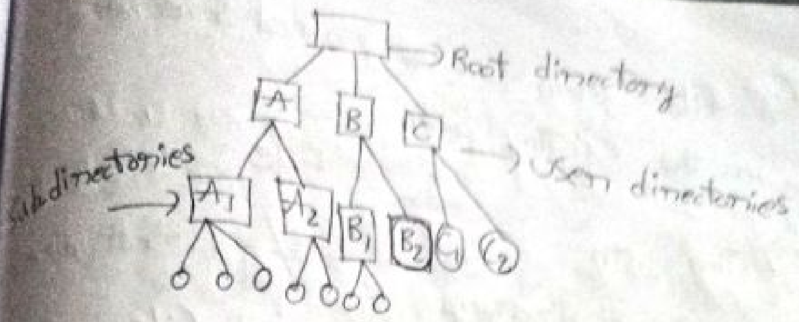
files are
e.

users with the
, Security is less

ry

y can be
are
hich leads

ferent
ntification



Advantages:-

→ For each user, one Separate directory.
→ For each directory, there is a (periodically) possibility to create Sub-directories, so that group of files can be Segregated & Stored in Separate Sub-directories.

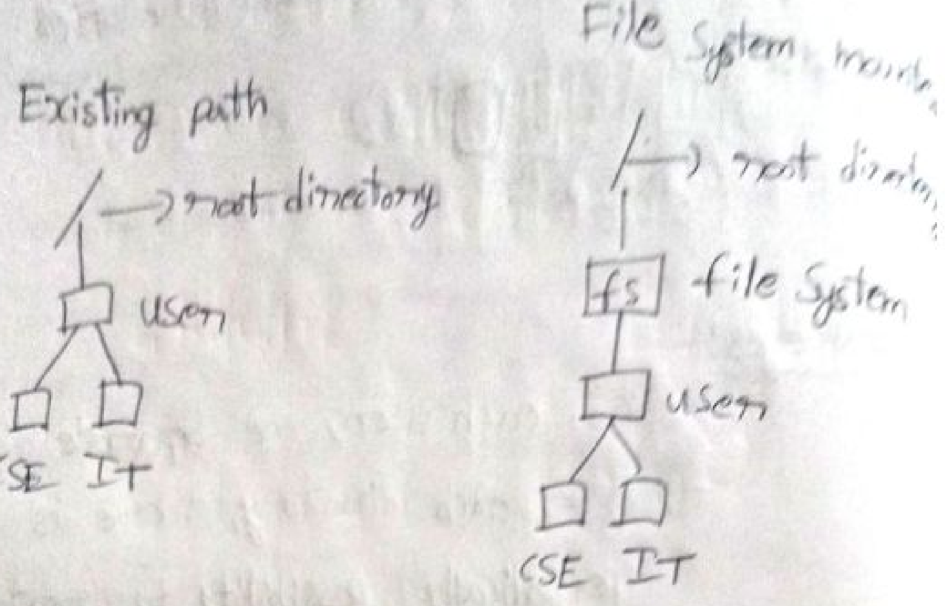
File System mounting:- High no. of directories.

It is a part of os which takes care of all files available in the System.

A file must be opened before it is being accessed. In the similar way, a file System must be loaded some where in the memory so that the processes available in the System can make use of it.

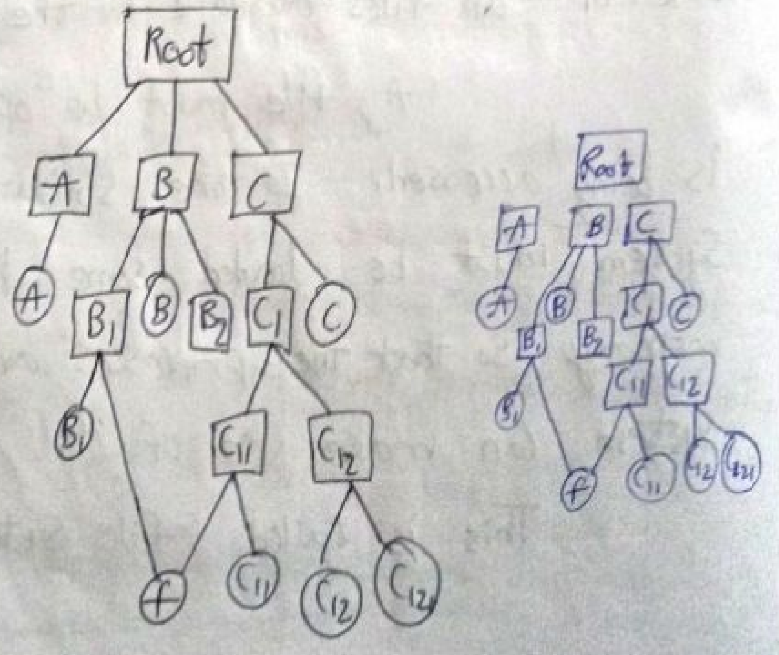
This is called 'file System mounting'.

A file system can be mounted somewhere in the existing path.



=) <u>File sharing:-</u>

A file which is created in 1 particular directory can be shared by many other directories available in the system. This concept is called "file sharing".



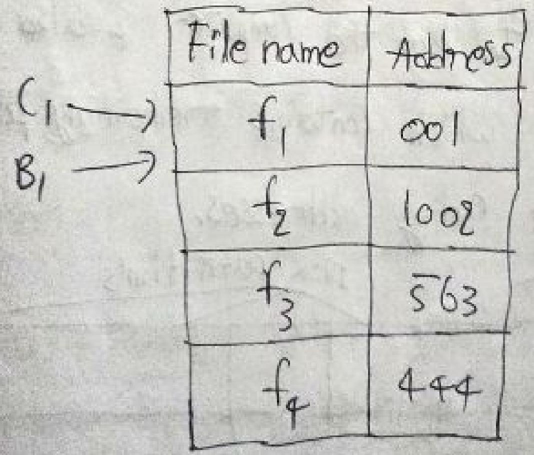
problem of file sharing:-

when a particular file 'f' is shared by B & C, directories. If any modification is done on the file 'f' by B, then the modification is not available in C,.

Solutions:-

Solution ①:-

Instead of making shared file into copies, a data structure called "I-node", is created & this "i-node" is given to all directories. This i-node contains file names & addresses. If any one directory modifies the file, then modifications can be accessed by all the directories.



$C_1 \rightarrow$
$B_1 \rightarrow$

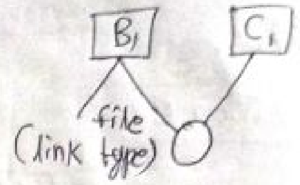
File name	Address
f_1	001
f_2	1002
f_3	563
f_4	444

i(index)-node

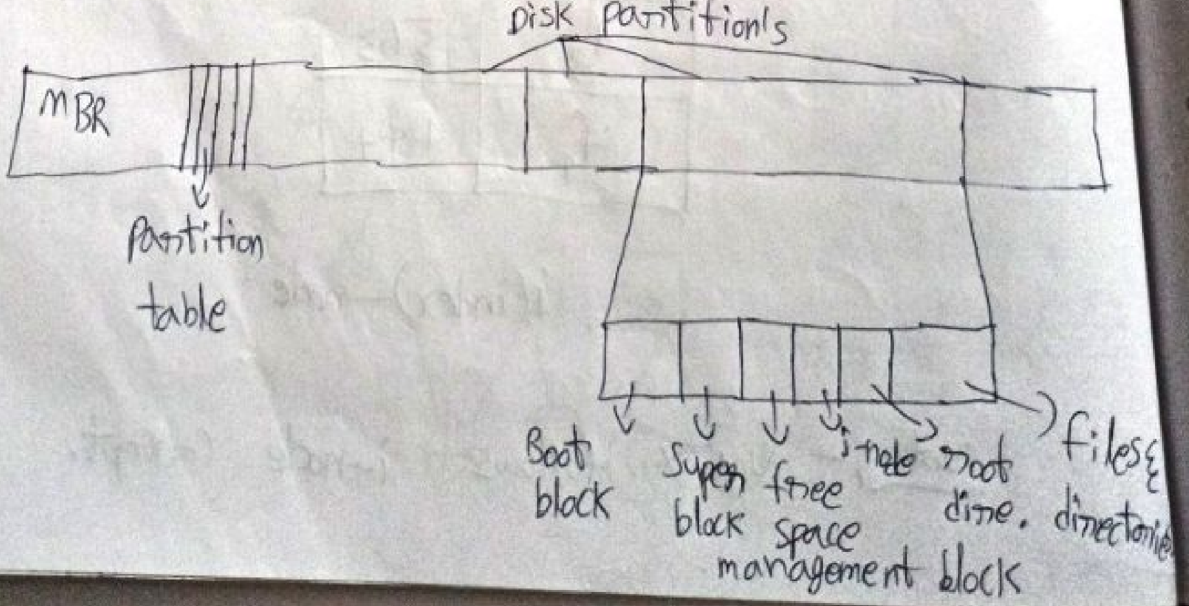
Note:- UNIX follows i-node concept.

=> Solution ②:-
when a particular file 'f' is shared by B_1 & C_1 directories, the directory B_1 creates a file, which is of type "LINK". This file consists of only path name, as given below.

$B_1/C_1/C_{11}/f_1$



=> File System Structure:-
Although, File System is stored in magnetic Disk. Disk is divided into no. of partitions. Each partition is of different size. The 1st pattern is called 'master Boot Record (MBR)'. The purpose of MBR is to boot the computer & also it contains a table which contains remaining partitions starting & ending addresses.



=> parti
a) no.
b) par
c) st
d)
E
a)
b)
c)
=>
a)

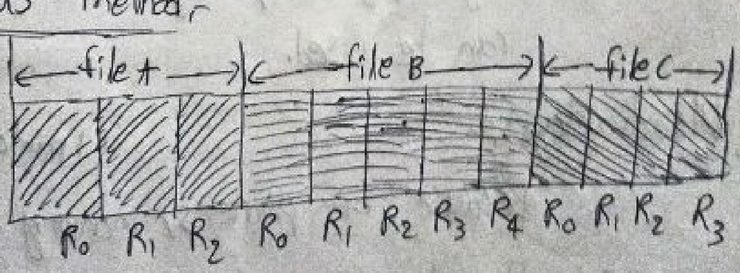
=) partition table contains the following Info:-

a) no. of partitions
b) partition no.
c) starting address of partition,
d) ending address of partition.

Each disk partition is made into 6 Blocks:-
a) Boot Block:- which boots os.
b) Super Block:- contains key parameters of file systems.
c) Free Space management Block:- How much memory is still left over.
d) i-nodes :- contains filenames & addresses.
e) Root directory:- Base directory
f) Files & directories:- Available files & directories.

=) Implementation of File Systems:-

i) Contiguous method:-

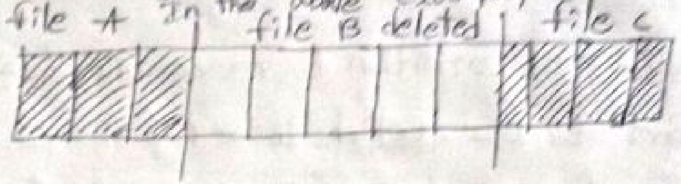


Advantages:-

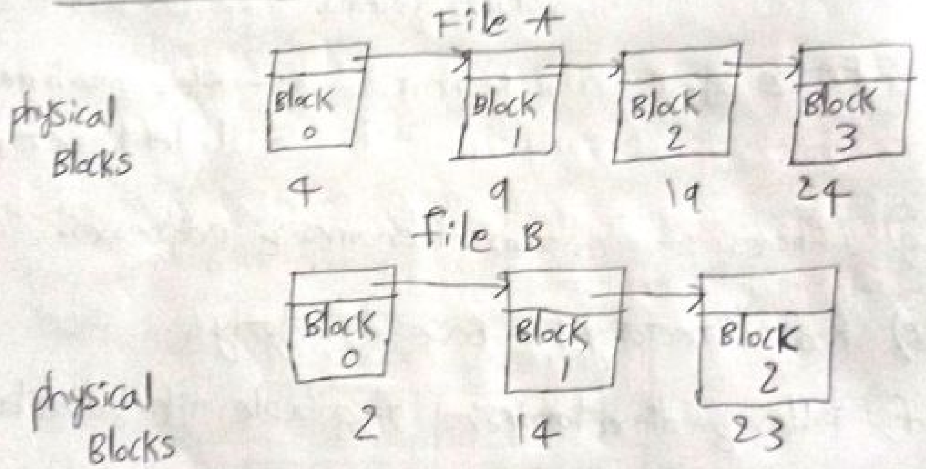
Simplest method to implement, Accessing (Reading & writing) is easy.

Disadvantages:-

It's Time consuming because of sequential Searching, memory is wasted. In the above example,



b) Linked List Allocation method:-



Advantages:-

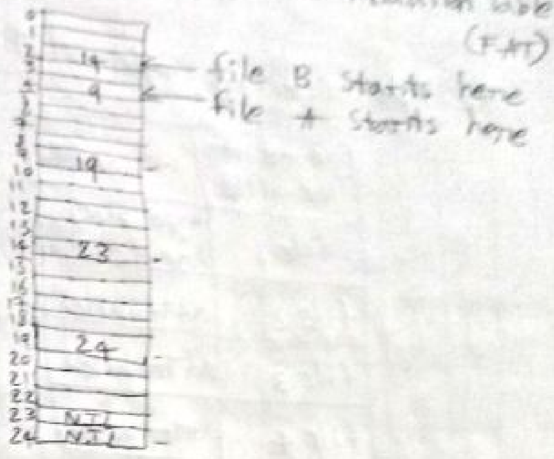
Any empty block, anywhere in the disk can be used.

Disadvantages:-

(1 pointer = 2 bytes)

Because of pointers, memory is wasted, But, Even though random access is possible the process is very slow.

The Disadvantages are rectified by introducing a data structure called "File Allocation Table". (FAT)



i) i-node method:-

The first entry of an i-node table contains file attributes like name, creator, owner, size etc. Then each entry contains the disk address of one particular block.

when file is extended in the future, to contain the disk addresses of the new blocks, the last entry of the i-node table is reserved.

Advantages:- No memory is wasted, Fast accessing.

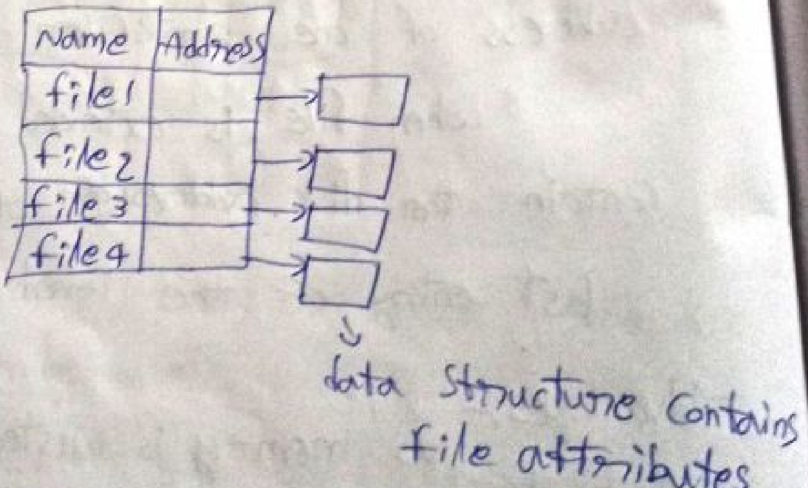
=> Directory Implementation:-

3 different ways,

a) Simple directory implementation:-

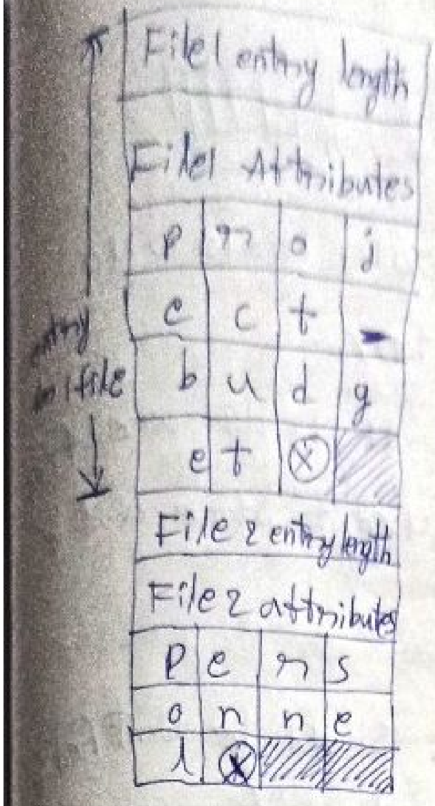
Name of the file	Attributes
file 1	type, size, date of creation
file 2	attributes
file 3	attributes
file 4	attributes

b) A small modification with 1st method, that attributes are stored in a separate data structure, which addresses will be stored in directory entries.

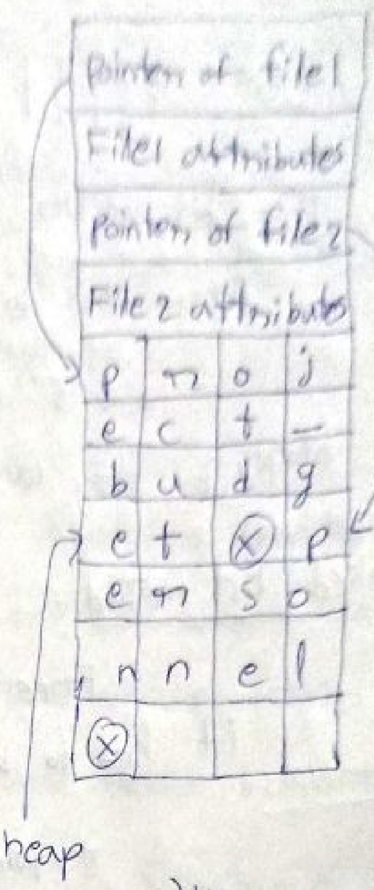


Note:-
method ① & method ② both supports only when the file name is fixed size(1-8 characters)

c) Direct Implementation using Heap & Inline:-



a) In-line



b) Heap

Free space management:-

whenever a file is stored & after execution the file will be deleted. The blocks will become empty. In that empty blocks, a new file can be stored. Handling these free blocks to store a new file is called "free space management".

1) Bit vector method:-

By this method,

0 – block is allocated,

1 – block is free.

Now, we write a bit vector as a combination of 0's & 1's, so that, free blocks can be identified to store a new file.

Ex:-

The bit vector is; 00111 1001010110

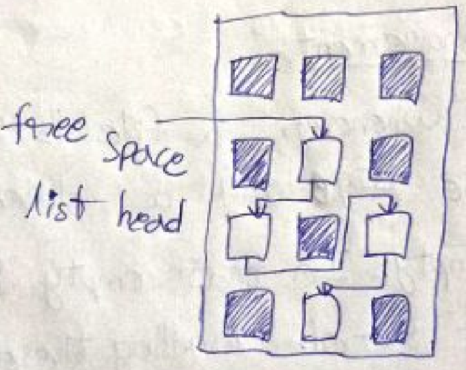
From this bit vector,

3 4 5 6 9 11 13 14 are free blocks & they can be used to store a file.

b) <u>Linked list method:-</u>

Free block should be identified & it has to be named as "free space list head.

A pointer is used to point the next free blocks & so on.



c) <u>Grouping:-</u>

According to this, if a countable no. of blocks are continuously free, then the free blocks are grouped in such a way that the first free block contains disk addresses of next free blocks.

I]

lo110

are free
a file.

entified &
ace list head!
the next

f blocks
are
ee block
ks.

Ex:-

If there are 1 to n blocks are free, then in the 1st Block 2 to n block address will be available.

⇒ **I/o Systems:-**

⇒ Introduction:-

The control of devices connected to the computers is a major concern of OS design because I/o devices vary so widely in their fn & speed, a variety (methods) are needed to control them. These methods form the I/o subsystem of the kernel, which separates the kernel from the complexity of managing I/o devices.

⇒ **I/o Hardware:-**

Computers operate $\overset{\text{many}}{\underset{\&}{?}}$ kinds(many) of devices.

They are fit into categories:-

⇒ Storage Devices → disks, tapes.

⇒ Transformation Devices → modems, h/w l s/w cards.

⇒ Human-Interface Devices → keyboard, mouse.

The one thing we have to know is how the devices are attached & how the s/w can control h/w.

=) **Bus:-**

A device communication with the machin via a connection point(or port). If one on more devices use a common set of wires, the connection is bus.

=) **Controller:-**

It is a collection of electronics that can operate a port a bus/device.

=) **Interrupt:-**

H/w mechanism, that enables a device to notify the cpu.

It is as follows:-

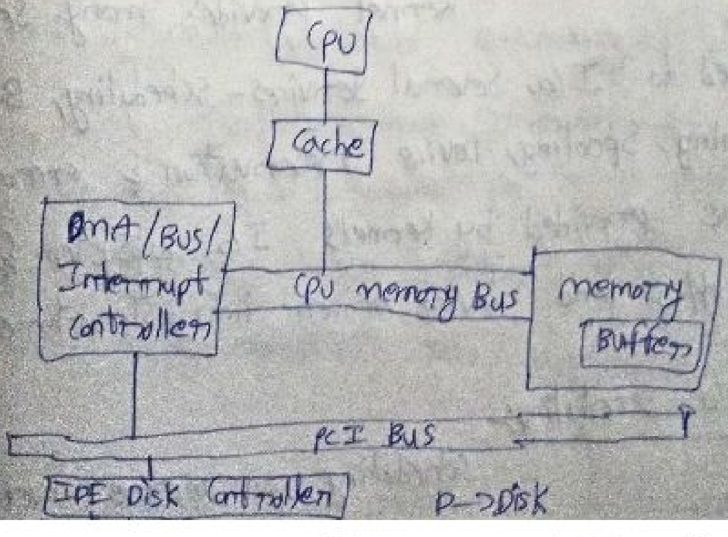
cpu H/w has a wire interrupt request line, that cpu senses after executing every instruction. when cpu detect that, controller has asserted a signal on interrupt request line, the cpu saves a small amount of state, such as current value of the instruction pointer, & Jumps to the interrupt-handler routine at a fixed address in memory. Interrupt handler determines the cause of interrupt, performs the necessary processing, & executes a return from instruction to return the cpu to the execution state prior to interrupt.

→ Direct memory Access:-

when device does large Transfers, w/o (informing) burdening cpu, the reading can be done In memory directly.

To initiate DMA transfer, the host writes a DMA command block into memory. This block contains a pointer to the source of a transfer, a pointer to the destination of a the transfer, & a count of no. of bytes to be transfered. cpu writes the address of this command block to DMA controller, then goes on with other work.

The DMA controller proceeds to operate the memory bus directly, placing addresses on bus to perform transfers without help of the main cpu.



=) Application I/o Interface:-

An app. can open a file on a disk w/o knowing what kind of disk it is, & how new disks & other devices can be added to a computer w/o os being disrupted.

A given device may ship with multiple device drivers - for ex:- devices for ms-DOS, windows 95/98, windows NT/2000, and Solaries. The following I/o devices may vary from many dimensions:-

a) Character stream/ block, b) sequential/random Access,

c) Synchronous/ Asynchronous.

d) Shoretable/ dedicated.

e) Speed (operation)

f) Read-write, Read_only/write only.

=) Kernel I/o Subsystem:-

Kernel provides many services related to I/o, Several Services - scheduling, Buffering, caching, Spooling, Device reservation & error handling are provided by kernel's I/o Subsystem & build on H/w and device driver infrastructure.

a) I/o Scheduling:-

Scheduling can improve overall

system performance. Can share device access fairly among processes, & can reduce the average waiting time for I/o to complete. OS developers implement scheduling by maintaining a queue of requests for each device. When an app. issues a blocking I/o system call, the request is placed on the queue for that device. The I/o scheduler rearranges the order of queue to improve overall system efficiency & avg. response time experienced by applications.

Buffering:-

A Buffer is a memory area that stores data while they're transferred b/w 2 devices / b/w a device & an App. . Buffering is done for 3 reasons.

a) One is to cope with a speed mindset b/w producer & consumer of data stream.
Ex:- Double Buffering.

b) Another use of Buffering is to adapt b/w devices that have different data transfer rates.

c) Another use of buffering is to support copy semantics for application I/o.

Caching:-
A cache is a region of fast memory

that holds copies of data. Access to the cached copy is more efficient than access to the original. The difference b/w buffer & cache is that buffer may hold the only existing copy of a data item, whereas a cache, by definition, just holds a copy on faster storage of an item that resides elsewhere.

d) Spooling & Device Reservation:-

Simultaneous peripheral operational online, (online) onlining,

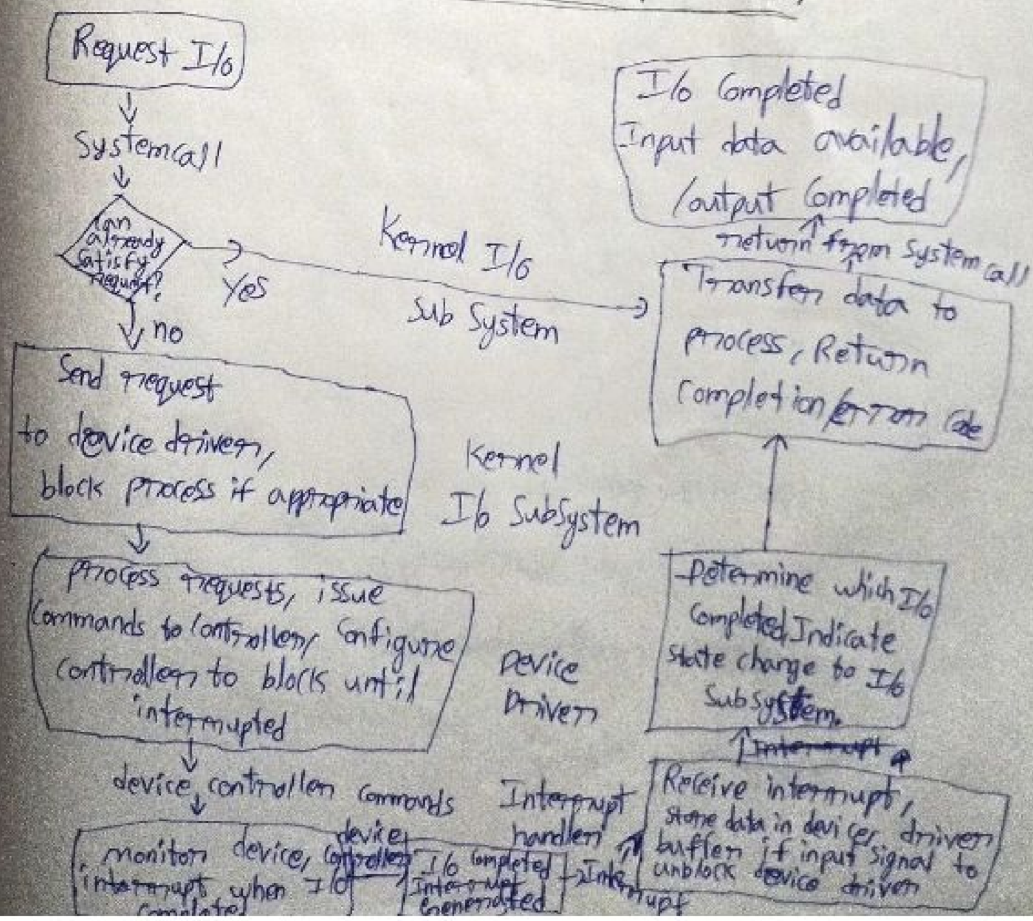
A Spool is a buffer that holds output for a device, printer like, that can't accept interleaved data streams. Although a printer can serve only one Job at a time, several apps may wish to print their output concurrently, w/o having their output mixed together. OS solves this problem by intercepting all output's to the printer. Each app's output is spooled to a separate disk file. when an app finishes printing, the spooling system queues the corresponding spool files to the printer one at a time.

e) Error handling:-

An OS that uses protected memory can

guard against many kinds of h/w & appl. errors, so that a complete-system is not the usual result of each minor mechanical glitch. Devices & I/o transfers can fail in many ways, either for transient reasons, such as h/w becoming overloaded, or for 'permanent' reasons, such as a disk controller becoming effective. As a rule, an I/o System call will return 1bit of Info about status of call signifying either success /failure. In UNIX OS, an additional Integer variable named errno is used to return an error code.

→ Transforming I/o to hardware operations:-



Request I/o
Systemcall
Can already satisfy request?
yes
no
Kernel I/o Sub System
I/o Completed Input data available, /output Completed
return from system call
Transfer data to process, Return completion/error code
Send request to device driver, block process if appropriate
Kernel I/o SubSystem
Process requests, issue commands to controller, configure controller to blocks until interrupted
Device Driver
Determine which I/o completed, Indicate state change to I/o Subsystem.
~~Interrupt~~
device controller commands
Interrupt handler
Receive interrupt, store data in device driver buffer if input signal to unblock device driver
monitor device, interrupt when I/o completed
device controller
I/o completed Interrupt Generated
Interrupt