Unit VI

File Management

* File :

i) Concept :

- A file is a collection of related information that is recorded on secondary storage.
- The data cannot be written to secondary storage unless they are within a file.
- Files represents programs and data.
- Data files may be numeric, alpha numeric, alphabetic or binary.
- In general, a file is a sequence of bits, bytes, lines or records, the meaning of which is defined by the file's creator and user.
- The information in file is defined by its creator.
- A file has a certain defined structure, which depends on its types.

ii) File Attributes :

- Every file name has a ~~str~~i name which is a string of characters.
- Every file has its own name & data.
- Os append or associate some information with the file.
- Such extra information is called as Attributes.
- There attributes are vary from one system to another.

i) Name : The symbolic file name is the only information kept in human readable form.

ii) Identifier : The unique tag, usually a number, identifies the file within the File system. It is the non-readable name for the file.

iii) Type : This information is needed for systems that support different types of files.

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

v) Size : The current size of the file and possibly the maximum allowed size are included in this attribute.

vi) Protection : Access-control information determines who can ~~read~~ do reading, writing, executing and so on.

vii) Time, date and user identification :
- This information may be kept for creation, last modification and last use.
- These data can be useful for protection security & usage tracking.

iii) File Operations :

- A file is a abstract datatype.
- To define files we need to consider the operations performed on files.
- The OS can provide system calls to create, write, read, reposition, delete and truncate files.
- The six basic operations :

i) Creating a file : There are two steps to create a file :

i) Space in the file system must be found for the file.

ii) An entry for the new file must be made in the directory.

ii) Writing a file : To write a file we make a system call by specifying,

i) Name of the file.

ii) The information to be written to the file.

iii) Reading a file : To read the file we make a system call by specifying,

i) Name of the file.

ii) Where the next block of the file should be put.

iii) the system needs to keep pointer to the location in the file where the next read is to taken place. Once the read has taken place, the read pointer is updated.

iv) Repositioning within a file : The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value.

v) Deleting a file : File need to be deleted to free up disk when it is no longer required.

vi) Truncating a file : The user may want to erase the contents of a file but keep its attributes.

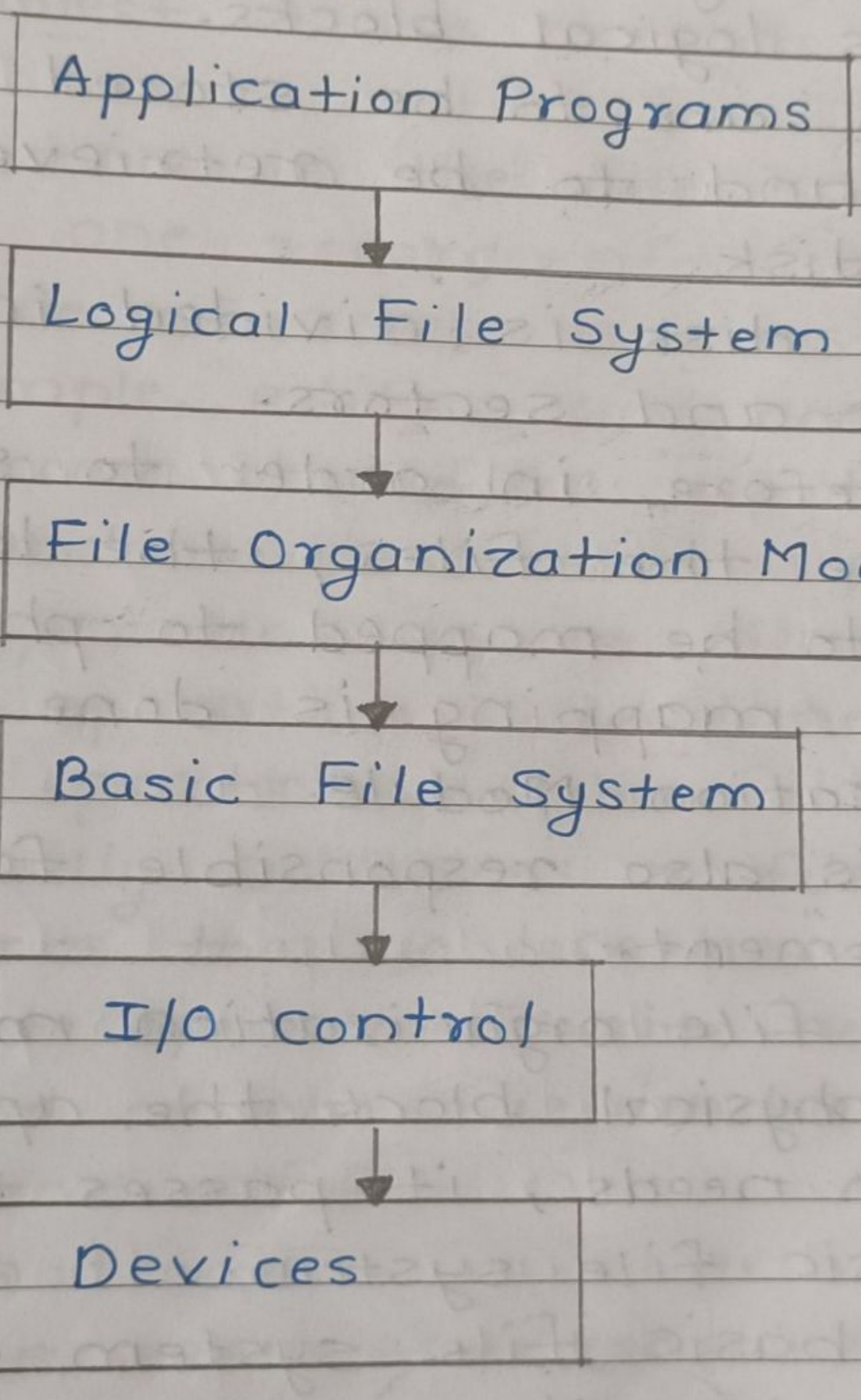
iv) File Types :

- When we design a file system, we always consider whether the OS should recognize and support file types.
- If an operating system recognizes the type of a file, it can then operate on the file in resonable ways.
- A common technique for implementing file types is to include the type as part of the file name.
- The name split into two parts :
  i) Name
  ii) Extension.
- Extension is seperated by ~~dot(dot)~~ dot.
- The system uses the extension to indicate the type of the file and the type of operations that can done on that file.

File Type	Usual Extension	Function
executable	exe, com, bin or none	ready-to-run machine language program.
object	obj, o	compiled, machine language, not linked.
source code	c, cc, java, perl, asm	source code in various languages.
batch	bat, sh	commands to the command interpreter.
markup	xml, html, tex	texual data, documents.
Word processor	xml, rtf, docx	various word processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archieve	rar, zip, tar	related files grouped into

v) File System Structure :

- File system provide effcient to the disk by allowing data to be stored, located and retrieved in a convenient way.
- A file system must be able to store the file, locate the file and retrieve the file.
- File System Layers :



- When an application program asks for a file, the first request is directed to the logical file system.

- The logical file system contains the meta data of the file and directory structure.
- If the application doesn't have the required permissions of the file then this layer will throw an error.
- Logical file systems also verify the path to the file.
- Generally, files are divided into various logical blocks.
- Files are to be stored in the hard disk and to be retrieved from the hard disk.
- Hard disk is divided into various tracks and sectors.
- Therefore, in order to store and retrieve the files, the logical blocks need to be mapped to physical blocks.
- This mapping is done by the File organization Module.
- It is also responsible for free space management.
- Once file organization module decided which physical block the application program needs, it passes this information to basic file system.
- The basic file system is responsible for issuing the commands to I/o control in order to fetch those blocks.
- I/o Controls contain the codes by using

which it can access hard disk.
- These codes are known as device drivers.
- I/O controls are also responsible for handling interrupts.

* Access methods :

i) sequential Access :
- The simplest Access method is sequential access.
- Information in the file is processed in order, one record after the other.
- This mode of access is most common;
- for example, editors and compilers usually access the files in this fashion.
- Reads and writes make up the bulk of the operations on a file.
- A read operation - read-next() - reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.
- The write operation write-next() appends to the end of the file and advances to the new end of file.
- key points :
i) Data is accessed one record right after another record in an order.
ii) When we use read command, it move ahead pointer by one.
iii) When we use write command, it will

allocate memory and move the pointer to the end of the file.

iv) such a method is reasonable for tape.

- Operations:

i) Read next:

It will read the next record in the file.

ii) Write next:

This operation is used when some more information is to be included in the file.

iii) Rewind:

This will bring the read and write pointers to the beginning of the file.

ii) Direct Access:

- Another method is direct Access or Relative Access.
- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- The direct-access method is based on a disk model of a file,
- Since disks allow random access to any file block.
- The file is viewed as a numbered sequence of blocks or records.

- Thus, we may read block 14, then read block 53, and then write block 7.
- There is no restrictions on the order of reading or writing for a direct-access file.
- A block number provided by the user to the OS is relative block number.
- The first relative block of the file is 0 and then 1 and so on.
- These methods are used in database management systems.
- Operations:

i) Read n :

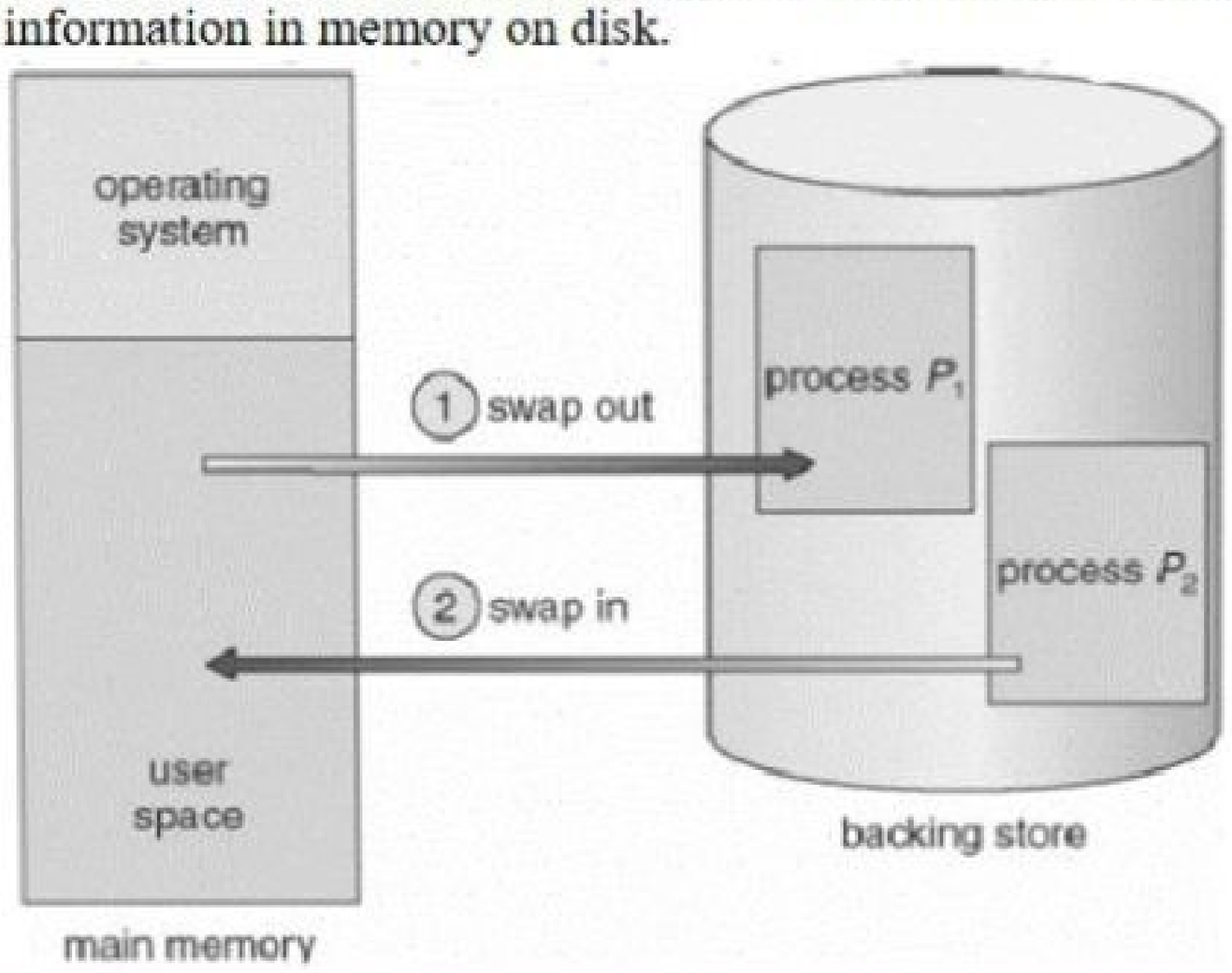
This operation is used to read the nth block.

ii) Write n : This operation is used to write in the nth block.

iii) Goto n : This operation is used to directly access the nth block.

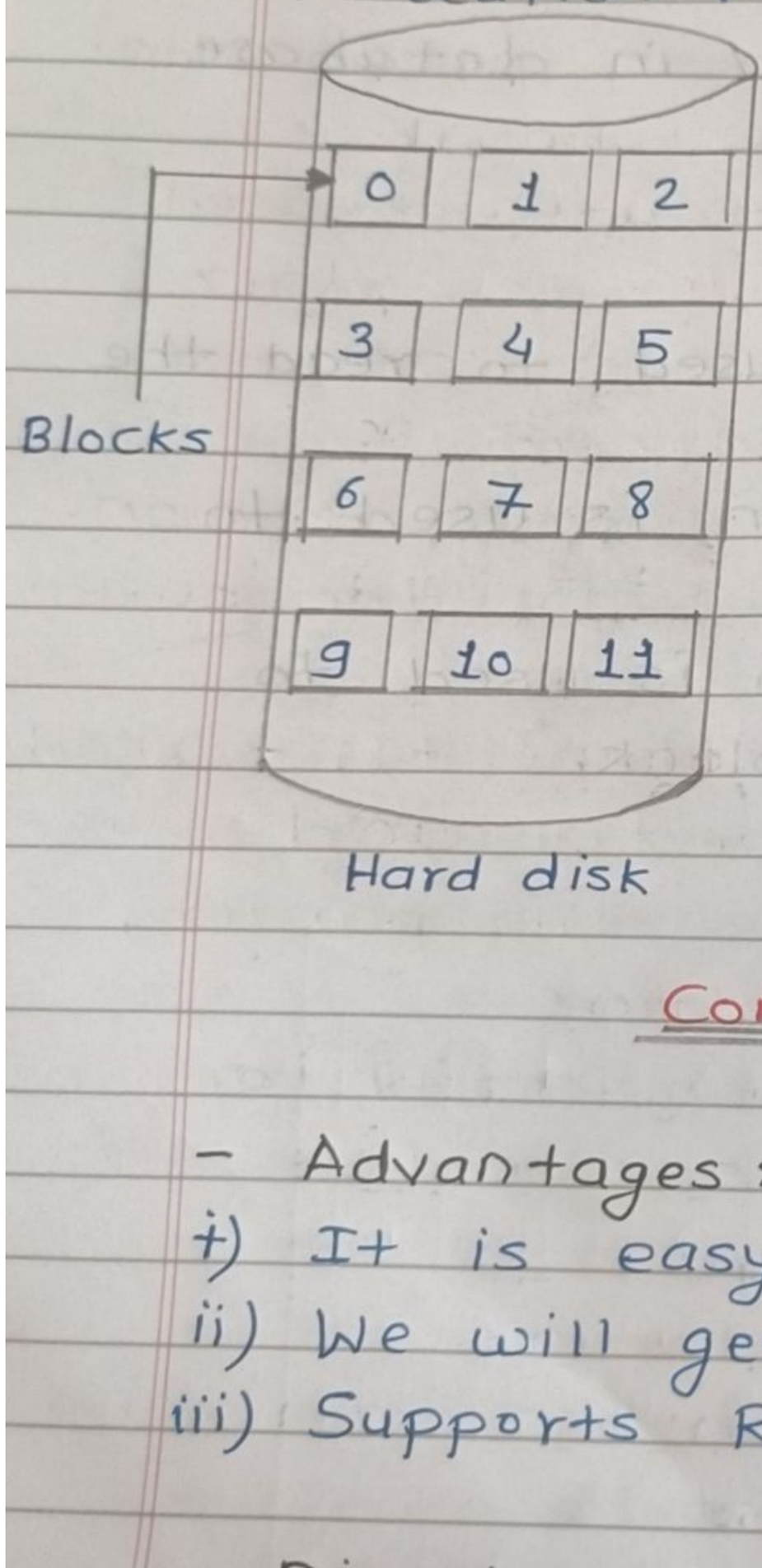
Swapping:

- ***Swapping*** is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.
- At some later time, the system swaps back the process from the secondary storage to main memory.
- Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason
- **Swapping is also known as a technique for memory compaction**.
- Swap space is a space on hard disk which is a substitute of physical memory.
- It is used as virtual memory which contains process memory image.
- Whenever our computer run short of physical memory it uses its virtual memory and stores information in memory on disk.



* File Allocation methods :

i) Contiguous Allocation :
- If the blocks are allocated to the file in such a way that logical blocks of the file get the contiguous physical block in the hard disk then this allocation method is called as contiguous Allocation method.



File Name	Start	Length	Allocated Blocks
file1.txt	0	4	0,1,2,3
sun.jpg	5	3	5,6,7
mov.mp4	9	3	9,10,11

Directory

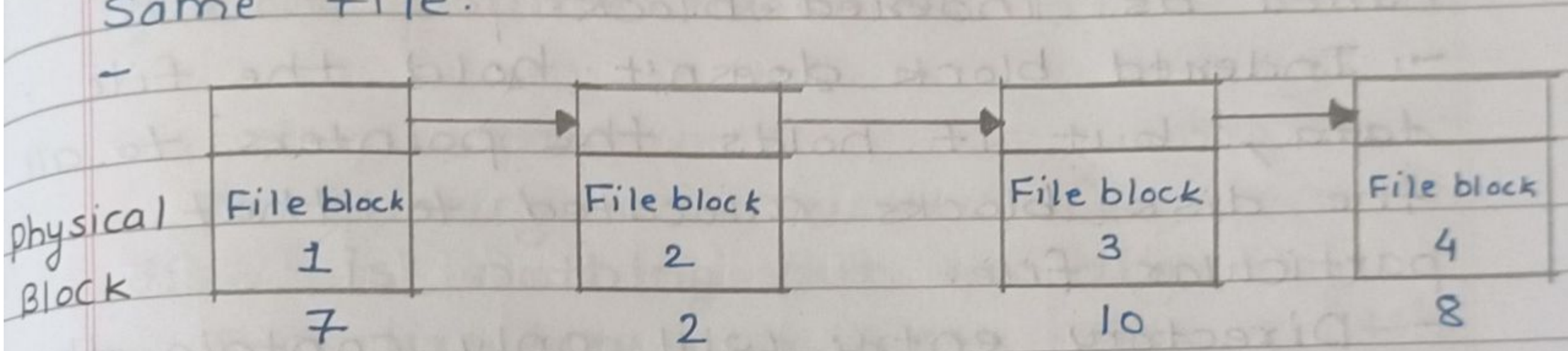
Contiguous Allocation

- Advantages :
i) It is easy to implement.
ii) We will get Excellent read performance.
iii) Supports Random Access into files.

- Disadvantage :
i) The disk will become fragmented.
ii) It may be difficult to have a file grow.

iii) Linked Allocation :

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



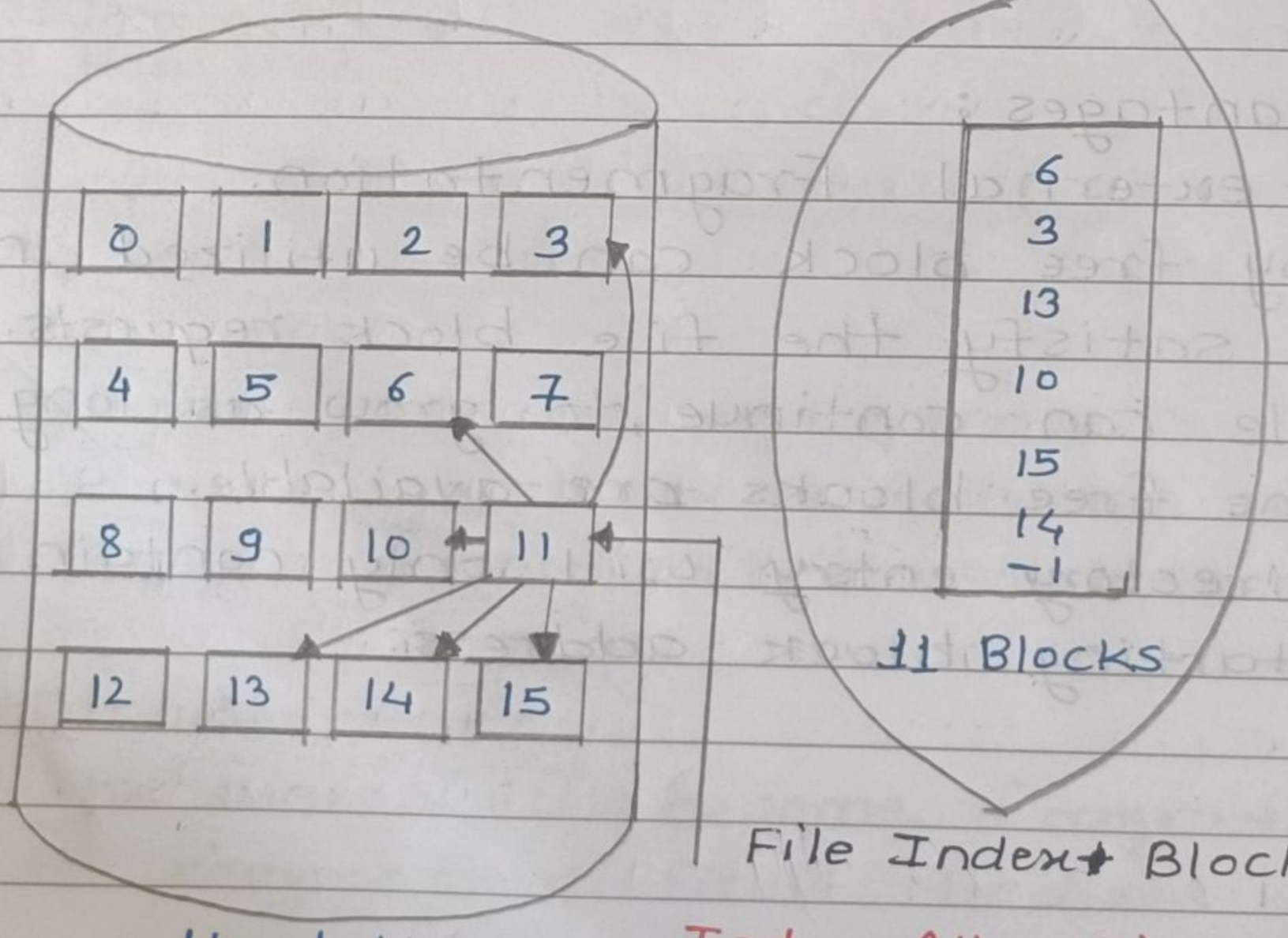
Linked List Allocation

- Advantages :

i) No external Fragmentation.
ii) Any free block can be utilized in order to satisfy the file block requests.
iii) File can continue to grow as long as the free blocks are available.
iv) Directory entry will only contain the starting block address.

- Disadvantages:
i) Random access is not provided.
ii) Pointers require some space in the disk blocks.
iii) Any pointers of the linked List must not be broken otherwise the file will get corrupted.
iv) Need to traverse each block.

iii) Indexed Allocation:
- Indexed Allocation scheme stores all the disk pointers in one of the blocks called as indexed block.
- Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file.
- Directory entry will only contain the index block address.



Index Allocation method

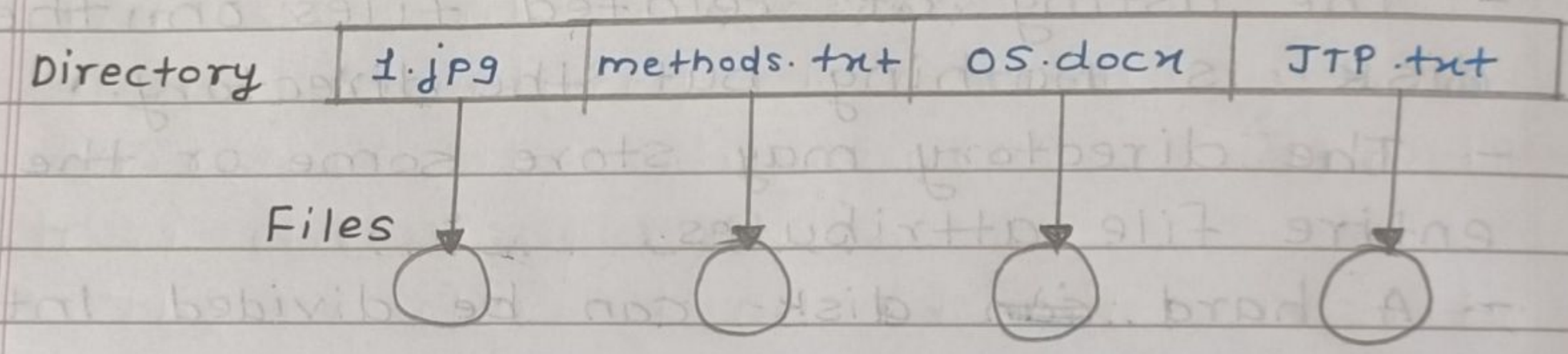
- Advantage:
i) supports direct access.
ii) A bad data block causes the lost of only that block.

- Disadvantage:
i) A bad index block could cause the lost of entire file.
ii) Size of file depends upon the number of pointers, a index block can hold.
iii) Having an index block for a small file is totally wastage.
iv) More pointer overhead.

* Directory structure:
- The listing of related files on the disk is nothing but the directory.
- The directory may store some or the entire file attributes.
- A hard ~~can~~ disk can be divided into number of partitions of different sizes called as volumes.
- Each partition must have at least one directory.
- A directory entry is maintained for each file in the directory which stores all the information related to that file.
- Operations to be performed:

i) File Creation.
ii) Search for the file
iii) File deletion
iv) Renaming the file
v) Traversing Files.
vi) Listing of files.

i) Single Level Directory :
- The single level directory is the simplest directory structure.
- All files are contained in the same directory which makes it easy to support and understand.
- It has some limitations,
- When the number of files increases or the system has more than one user.
- Since all the files are in the same directory, they must have unique name.
- If two user call same file with same name then unique name rule will be violated.
- Diagram :



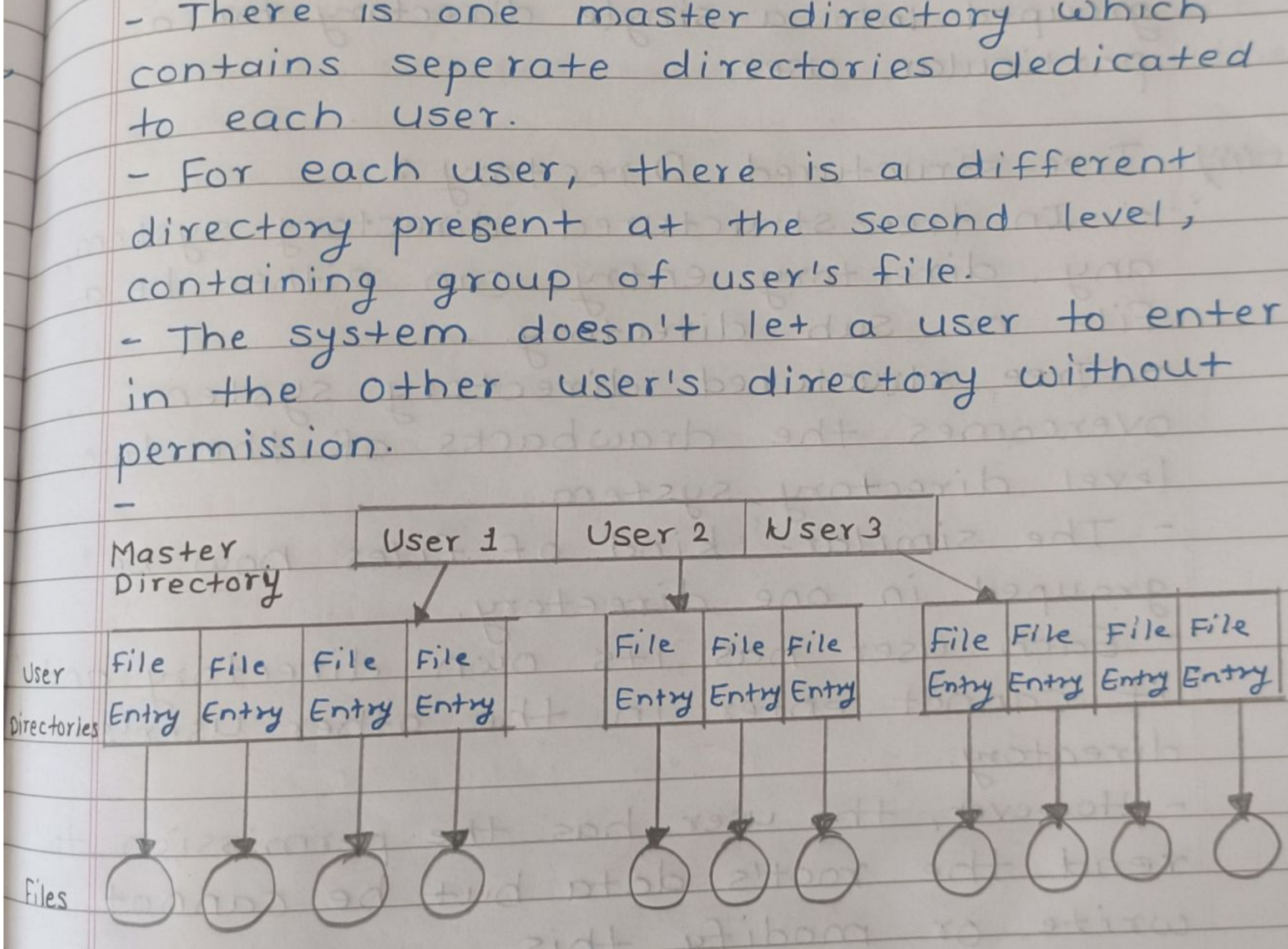
Single level Directory

- Advantages :
i) implementation is very easy.
ii) Searching will become faster.
iii) File creation, searching, deletion is very since we have only one directory.

- Disadvantage :
i) We cannot find two files with same name.

ii) Searching will become time taking if the directory is large.
iii) This can not group the same type of files together.

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



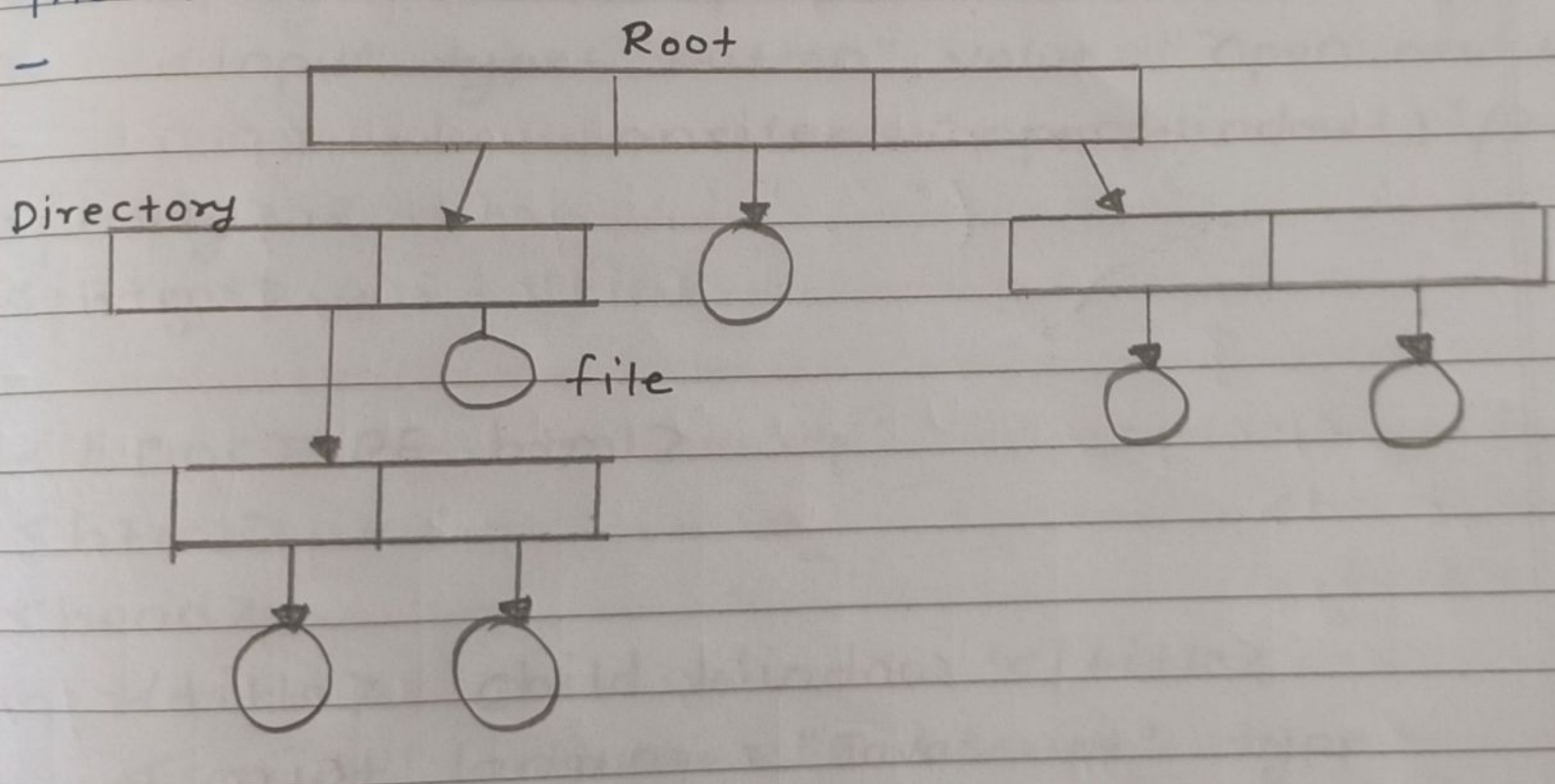
Two Level Directory

- Characteristics:
i) Each file has a path name as
/User-name/directory-name/

ii) Different user can have the same file name.
iii) Searching become more efficient as only one user's list needs to be traversed.
iv) The same kind of files cannot be grouped into a single directory for a particular user.

iii) Tree-structured Directory:
- In tree-structured directory system, any directory entry can either be a file or sub directory.
- Tree structured directory system overcomes the drawbacks of the two level directory system.
- The similar kind of files now be grouped in one directory.
- Each user has its own directory and it cannot enter in the other user's directory.
- However, the user has the permission to read the root's data but he cannot write or modify this.
- Only administrator of the system has the complete access of root directory.

- Searching is more efficient in this directory structure.
- The concept of current working directory is used.
- A file can be accessed by two types of path, either relative or absolute.
- Absolute path is the path of the file with respect to the root directory of the system.
- Relative path is the path with respect to the current working directory of the system.
- The user is given the privilege to create the files as well as directories.
-



Tree-structured Directory