3.1 INTRODUCTION

Consider a simple 'C' program that reads two integer values from the keyboard and adds them. Then it displays the sum as an output on the monitor. While executing this program, a user needs to enter two integer values. These values are added, and the user gets an answer on the monitor. This is OK. But the question is: what happens to these input and output values when a program (or process) terminates?

The answer is: when the program terminates, all memory is freed allocated to variables used in the program. So, the result is: input and output values are destroyed. Therefore, they cannot be used in the future again. But what if the user wants to use those values again? This requires to store them *permanently* so that they don't get destroyed when the program terminates.

'File' and secondary storage devices come into the picture here. Secondary storage devices provide non-volatile, permanent mass storage capabilities. They come in various types, like hard disks, floppy disks, CDs, DVDs, magnetic tapes, pen drives, and solid-state drives (SSD). The most commonly used device for storing large information permanently is the Hard Disk...! Or simply called, Disk...!

A large amount of data or information can be stored in the form of files on disks permanently in a non-volatile manner. This chapter begins with a description of disk organization. This follows various concepts of files and directories. After this, file access and disk space allocation methods are described. This follows a description of the Linux file system structure, features and types.

3.2 **DISK STRUCTURE

This section describes various physical aspects of hard disks. It starts with the physical structure of the disk. It follows a logical structure, in which the disk is considered from an operating system point of view. In the end, various types of addressing and disk I/O are described.

3.2.1 Physical Structure

Figure 3.1 depicts the physical geometry of a hard disk.

A hard disk is a sealed unit. It contains one or more circular **platters**. Each platter contains a magnetic coating on both of its surfaces. The platters are stacked one on top of another. They rotate together around a central spindle. Rotation speed usually is 3600, 5400, or 7200 rpm (Rotation per Minute). Typically, disks contain 1 to 8 platters. Each platter contains two surfaces (one above and the other below).

Data is read from and written to these platters using a number of read/write **heads**. Typically there are as many heads as surfaces on a disk. Each of these heads is attached to an arm. An arm can be moved to access different parts of the platter.

A platter can be thought of as a collection of circular, concentric, flat rings. These rings are called **tracks**. Data is stored on the surface of a platter inside these tracks in the form of bits (0 or 1). Usually, there are more than thousands of tracks on a single surface.

Each track is divided into fixed-size blocks called **sectors**. Normally, each sector is of 512 bytes, and hundreds of sectors are on a single track. A sector is the smallest physical storage unit on a disk. Read and write operations are performed in a number of sectors on a disk.

A group of tracks with the same radius are called **cylinders**. Here, each track is on a different platter. And each track can be accessed without moving the read/write head of a disk. The number of cylinders is the same as that of tracks.

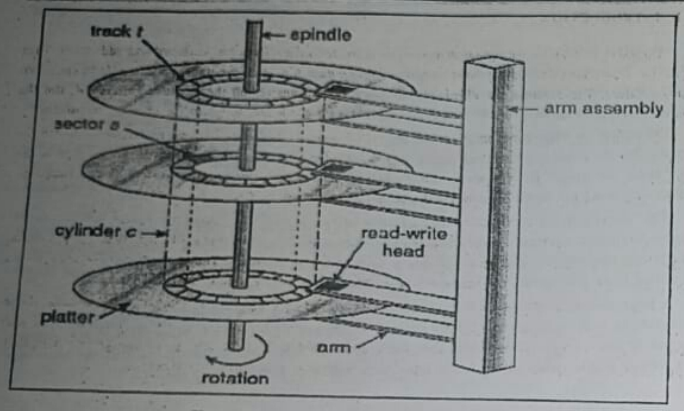


Figure 3.1 : Physical Geometry of a Disk

3.2.2 Logical Structure

A disk is considered a large one-dimensional array of fixed-size logical blocks. The size of blocks can be varied from system to system. Usually, the size will be 512 bytes (the same as that of a sector).

These blocks are numbered to identify them uniquely on a disk. Numbering starts from '0' or '1' to some maximum. The total number of blocks depends upon a storage capacity of a disk and the size of a block.

The operating system considers disk in this form, as depicted in the following figure. Here, blocks are considered the smallest unit of data transfer between the disk and main memory.

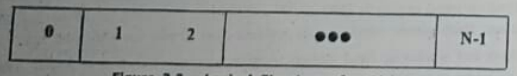


Figure 3.2 : Logical Structure of a Disk

Whenever any file requires storage space, it is allocated one or more blocks either contiguously or non-contiguously as per the allocation method. (Allocation Methods, Section 233.6) Later, these blocks can be accessed sequentially or randomly based on the access method. (Access Methods, Section 3.5)

Whenever there is a need to access any information from a disk, the operating system specifies a logical block number. This logical block number is mapped to a physical cylinder number (or track number), head number and sector number. The operating system does not worry about the cylinder, head, or sector numbers.

3.2.3 Addressing

To store data on the disks or to read it back, the operating system needs a way to specify where to write the data or from where to read it on the disk. The OS tells the drive the required position (location) to do so. Then, the drive moves the head to the proper position and reads or writes data.

Here, the smallest addressable unit is a sector. So, addresses are given to identify particular sectors. There are two methods to provide an address for such location: CHS (Cylinder–Head–Sector) and LBA (Logical Block Addressing).

These methods are described below:

- **CHS (Cylinder-Head-Sector) :**
  - This method identifies sectors by simply specifying the cylinder (radius), head (platter side), and sector (angular position) numbers.
  - Here, cylinders (or tracks) are numbered from 0 to some maximum from the outer cylinder to the inner cylinder. Sectors on each track are numbered from 0 to some maximum. Also, read/write heads are given numbers. So, a combination of a cylinder, head and sector identifies individual sectors uniquely on the disk.
  - This method was used on earlier systems.
- **LBA (Logical Block Addressing) :**
  - This method identifies sectors by simply specifying the sector number.
  - Each sector of the disk is assigned a sequential number starting from 0. A disk is considered a large one-dimensional array of fixed-size logical blocks. These types of logical blocks are mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder. Mapping proceeds in order. First, all the sectors on that track are numbered. Then other tracks in that cylinder are covered. And then, the rest of the cylinders are covered from outermost to innermost.
  - The following figure explains such type of numbering of sectors for a disk having only one surface. (Such disk is considered for simplicity only.)

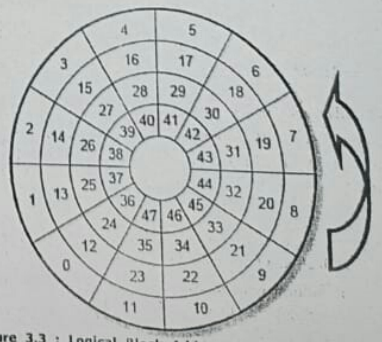


Figure 3.3 : Logical Block Addressing for Disk having Single

3.2.4 Disk I/O

As described earlier, for the
to the **main memory** from the d
on data stored on disk. Data
As described in previous
The sector is the smallest addr
Generally, the size of the sec
both – read and write – ope
Here, to perform a read
where to read data or write
of a required location.
Such address is provid
A hard disk controlle
various commands provided
them.
Addressing the systen
If the addressing syst
sector (CHS). If the addre
required to determine an
Once the location is
i) Move the disk
for this operat
ii) Wait till the s
for this opera
iii) Read/write th
After completing
operation has been co

3.3 FILES AND FI

This section int
It starts with th
describe why we ne
view and system vi
and types are descr

3.3.1 File

- "A file
- A file is
informat
- A file s
machine

3.2.4 Disk I/O

As described earlier, for the sake of performance, all the information (instructions and data) must come to the **main memory** from the **disk** before the CPU operate on it. Disk I/O refers to read and write operations on data stored on disk. Data is read from disk to memory and written to disk from memory.

As described in previous sub-sections, an entire hard disk can be considered a collection of sectors. The sector is the smallest addressable unit on the disk. Also, the sector is the smallest unit of data transfer. Generally, the size of the sector is 512 bytes. So, data transfer takes place in multiples of 512 bytes in both – read and write – operations.

Here, to perform a read or write operation, the operating system needs to provide a **location** – from where to read data or write data on a disk. For this purpose, the operating system provides the address of a required location.

Such address is provided to the hard disk controller.

A **hard disk controller** is an interface between the operating system and the disk. It converts the various commands provided by an operating system into proper signals so that disk hardware can understand them.

Addressing the system to identify locations on a hard disk can be either LBA or CHS.

If the addressing system is LBA, this controller translates the address to the appropriate cylinder-head-sector (CHS). If the addressing scheme is CHS, then there is no need for such translation. CHS value is required to determine an actual physical location on disk for read/write operation.

Once the location is identified, the following three steps are followed to perform the read/write operation:

i) Move the disk arm to position the read/write head at the specified cylinder. The time required for this operation is called **seek time**.

ii) Wait till the specified sector directly comes above/below the read/write head. The time required for this operation is called **rotational latency**.

iii) Read/write the data.

After completing these operations, the disk controller informs the operating system that the given operation has been completed. The operating system can now go on further processing of data.

3.3 FILES AND FILE SYSTEM

This section introduces files and file systems.

It starts with the basics of files and file systems. Then, some fundamental reasons are given which describe why we need files. This follows a description of two different points of view about files: user view and system view. After this, various file concepts such as naming, attributes, operations, structures and types are described.

3.3.1 File

- "A file is a named collection of related information stored in a secondary storage device."
- A file is used to store information permanently. It is also suitable to store large amounts of information (compared to variables used in programs).
- A file stores various information, like text, images, audio (songs), video, database tables, and machine codes.

- A file is an independent entity from processes, users and machines. Information stored in files remains as it is even when –
- A process, which creates a file, terminates;
- The user, an owner of the file, logs off from the system; or
- Machine gets switched off.
- Information stored in files can be shared among different processes, users as well as machines.
- A file is stored on a non-volatile secondary storage device such as a hard disk, CD-ROM, magnetic tape, pen drive, and SSD. But, access from such devices is slower. So, information being used from files is transferred to the main memory before the CPU operates them.
- A file is a static/passive entity. Information stored in files remains as it is until the user modifies it.
- A file has a longer life span. Once a program is created and stored on disk, it remains there till it is not deleted. So, the life span may be in months or even years.

3.3.2 File System

- "A part of an operating system, which deals with files, is known as a **file system** or **file manager**."
- A file system (file manager) is responsible for managing all the files.
- It provides facilities to create & delete files, read & write the contents of files, etc.
- It also provides **directories** to organize files. (Details are in Section 3.4.)
- It is the responsibility of a file system to provide efficient **access** to information stored in files.
- It is also the responsibility of a file system to **manage** memory space on secondary storage devices** when files are created, modified or deleted. (Details are given in section 3.6.)
- It also provides a **protection** mechanism to allow users to administer how other users can access the information in files.

3.3.3 Why do we need files ?

All computer applications need to store and retrieve information.

While a process runs, it can store information in its address space (data part) using some variables. The first problem with this way of storing information is: information is lost when the process terminates. Second is: it is not suitable to store very large information. Applications like railway reservation and banking require large storage space to store information. And third is: it is not possible to share information among different processes.

Thus, there are three essential requirements for long-term information storage:

1. It must be possible to store a **massive** amount of information.
2. Information should not be **lost** when the process terminates.
3. **Sharing** of information and concurrent access by two or more processes should be possible.

So, the usual solution to all these problems is to use 'Files'.

3.3.4 User-view v/s System-view

Files and file systems can be viewed from two different standpoints: user view and system view.

- **User view :**
  - From a user point of view, the essential aspect of a file system is how files appear to users. It means how files are constituted, how files are named, how files are protected, what operations are allowed on files, and so on. Here, the primary goal is user convenience.
- **System view :**
  - From a system point of view, it is crucial to see how files are managed. It means how files are accessed, how files are allocated memory space and so on. Here, the primary goal is efficiency.

For these reasons, this chapter is structured into several sections. The following sub-sections are concerned with a user interface to files. The following section covers various aspects of directories – containers for files. In the later part of the chapter, various file access and disk space allocation methods are covered. Linux file system is described at the end of the chapter.

3.3.5 File Naming

Files are used to store information on the disk. This information needs to be read back later. The user does not need to be concerned about details such as how and where the information is actually stored on the disk and how disks work.

For this reason, files are named. When a file is created, it is given a name. Later on, it is referred to by this name. This is to provide convenience to human users.

In reality, information is stored on physical devices such as disks. But, operating systems hide (abstract) physical device and provides logical entity (files) to store information. Due to this reason, files are sometimes referred to as **logical devices.**

The exact rules for naming files vary from system to system.

All current operating systems allow strings of one to eight letters, including digits and some special characters, as legal file names. Thus *test*, *xyz123*, and *chap_1* are possible file names. Many file systems support names as long as 255 characters.

Some file systems support case-sensitive file names, while others do not. Case sensitive means it distinguishes between upper and lower case letters. UNIX comes under the first category, while MS-DOS comes in the second category. So, *test1*, *Test1*, and *TEST1* are three different files in UNIX. But, all these names refer to the same file in MS-DOS.

Many operating systems support two-part file names, with the two parts separated by a period ('.'), as in test.c. The part following the period is called the *file extension*. It usually indicates something about a file.

In MS-DOS, file names are 1 to 8 characters, plus an optional extension of 1 to 3 characters.

In UNIX, the size of the extension depends upon the user; a file may contain two or more extensions, such as *chap.t.Z*.

Some of the more common file extensions are given in the following figure.

Extension	Meaning
.txt	General text file
.c	C source program
.bak	Backup file
.obj	Object file (Compiler output, but yet not linked)
.exe	Executable file
.gif	Graphical Interchange Format file
.hlp	Help file
.html	World Wide Web Hyper Text Markup Language file
.jpg	Still picture encoded with JPEG standard
.mp3	Music encoded in MPEG layer 3 audio format
.mpg	Movie encoded with the MPEG standard
.pdf	Portable Document Format file
.ps	PostScript file
.zip	Compressed archive file

Figure 3.4 : Some typical file extensions

3.3.6 File Attributes

Every file has its name and data. In addition to these, all files contain some other properties. Such properties are called file attributes. They vary from one operating system to another. The main file attributes are as follows:

1. **Name :**
   - A string of alphanumeric characters and some special characters like underscore ('_').
   - It is used by users to refer to files conveniently.
2. **Identifier :**
   - It is a unique tag, usually a number.
   - It uniquely identifies the file within the file system.
   - It is used by the operating system to refer to files.
3. **Type :**
   - It is used to identify the type of file.
   - It is generally expressed in the form of a file extension. For example, prog.cpp indicates a C++ file.
4. **Location :**
   - This is a pointer to the device and location on that device of the file.
5. **Size :**
   - It specifies the current size of the file (in bytes, words, or blocks), as well as the maximum allowed size of the file.

6. **Protection :**
   - This information determines who can read a file, write a file, execute a file and so on.
7. **Usage Count :**
   - This value indicates the number of processes that are currently using (have opened) this file.
8. **Time, Date, and User Identification :**
   - It specifies information regarding file creation, update and last access.

All information regarding a file is stored along with the file contents (actual data). Generally, directory structures are used to store information such as file names and their unique identifier. This identifier, in turn, gives the location where the other attributes are stored.

3.3.7 File Operations

Files are used to store information. This information needs to be retrieved later. It is the responsibility of an OS to provide ways to perform various operations on files.

Operating systems generally provide a set of system calls to perform various operations on files. This system calls can be called using library functions provided by programming languages.

The primary file operations are given below:

1. **Create :**
   - A new file can be created by a system call embedded in a program or by an OS command issued by an interactive user.
   - While creating a file, two steps are followed:
   - First, the necessary disk storage space is allocated to the file.
   - Second, an entry for the newly created file is made in the directory. The directory entry contains a file name, identification number and possibly some other information.
2. **Delete :**
   - When a file is no longer needed, it has to be deleted to free up disk space.
   - While deleting a file, two steps are followed:
   - First, all the file space on the disk is released.
   - Second, the directory entry is erased.
3. **Open :**
   - Before using a file, it must be opened.
   - File attributes and data contents are fetched in the main memory for rapid access on later calls. (Remember, access to main memory is faster than disk.)
   - Memory space in the main memory is allocated to store fetched information.

4. **Close :**
   - When the use of the file is finished, it should be closed to free up the main memory space.
   - File attributes and data contents are stored back on disk. This may contain modified information if the file is updated.
5. **Read :**
   - Data are read from the file.
   - The system maintains a read pointer. It specifies the location in a file from where to read data contents. In the beginning, it points to the starting location in a file.
   - The read pointer is updated automatically after each read operation.
   - The user needs to specify information like file name, how much data to read, and where to put that data.
6. **Write :**
   - Data are written to the file.
   - The system maintains a write pointer. It specifies the location in a file where to write data. In the beginning, it points to the starting location in a file.
   - If data are written at the end of a file, the file's size increases.
   - If data are written in the middle of a file, existing data are overwritten and lost.
   - The write *pointer* is updated automatically after each write operation.
7. **Append :**
   - This is a restricted form of a write operation.
   - Here, data are only added to the end of a file.
8. **Seek :**
   - For random access files, a location is needed to specify from where to start the read/write operation.
   - This operation repositions the file pointer (read/write pointer) to a specific location in a file. After this, the next read/write operation starts from that position.
9. **Get Attributes :**
   - This operation is used to retrieve file attributes.
10. **Set Attributes :**
    - This operation is used to write file attributes.
    - Generally, it is used to change file attributes such as file protection information.
11. **Rename :**
    - This operation is used to change the name of an existing file.
    - This is not strictly necessary. Because a file can be copied to a new file with a new name, and then the old file can be deleted.

3.3.8 File Structures

"File structure is the way of storing data contents (information) in a file."

It represents how data are stored in the file.

There are three common possibilities for storing data in a file. Or in other words, there are three common file structures:

- Byte Sequence
- Record Sequence
- Tree

All these three structures are depicted in the following figure.

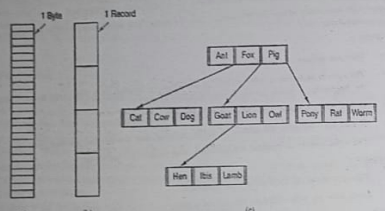


Figure 3.5 : File Structure : (a) Byte sequence, (b) Record Sequence, (c) Tree.

The description of all these three file structures is as follows:

1. **Byte Sequence:**
   - Data are stored as an unstructured sequence of bytes.
   - The operating system does not know or care about what is in the file. The entire file is treated as an array of bytes.
   - User programs can impose any meaning on data stored in a file.
   - Generally, the Read operation returns one byte, while the write operation overwrites or appends one byte.
   - It is the most flexible form of file structure.
   - Both UNIX and Windows use this approach.

2. **Record Sequence :**
   - Data are stored as a sequence of fixed-length records. Each record can have its own structure
   - Read operation returns one record, while write operation overwrites or appends one record.
   - It is suitable for special applications such as database-related applications.

3. **Tree :**
   - Data are stored as a tree of records. Each record can have its own structure.
   - Records may be of varying lengths.
   - Each record has a key in a fixed position in the record.
   - Records are sorted by key for easy search and retrieval.
   - Used with large mainframe systems.

3.3.9 FILE TYPES

In general, files can be categorized into one of the following four types:

1. **Regular files :**
   - Contains user information.
   - Regular files (*users' actual files*) can further be categorized into text files and binary files.
   - For example, a source code file is a text file, while an image, audio, or video is a binary file.
   - (This chapter is primarily based on this type of file.)

2. **Directories :**
   - They are containers for files and other sub-directories.
   - Used to manage other files and sub-directories.
   - (Remember that directories are also considered files in general.)

3. **Character special files :**
   - They are related to I/O.
   - Used to model serial I/O devices such as terminals and printers.

4. **Block special files :**
   - They are also related to I/O.
   - Used to model block I/O devices such as disks.

In UNIX, files are categorized into these four categories.

There are three different ways to differentiate among such types of categories:

1. **Encode type into file extension :**
   - This is a common technique for implementing file types.
   - The same file extensions are used for all files of the same type. Such as, all bitmap image files contain '.bmp' as an extension.
2. Encode type into file **attribute :**
   - Each file contains an attribute called file type. File type can be encoded in this attribute.
3. Encode type into **file data :**
   - File type can also be encoded in file data.
   - For example, UNIX uses a magic number to identify executable files. It contains the first few bytes of file data.

3.4 DIRECTORY SYSTEMS

Today, computer systems come with disks with Gigabytes and Terabytes of storage capacities. They are used to store vast amounts of information in various files. To manage all these files, they should be organized well.

This organization is generally done in two parts:

- First, disks are split into one or more **partitions**, known as **drives** or **logical devices**. (Remember drives like 'C:', 'D:', 'E:' in the Windows environment?) These drives are used to store files and directories.
- Second, each partition (drive) contains various **directories** or **folders** to group files and sub-directories.

In this section, directories are described along with some other concepts.

This section starts with the basics of directories. It follows various operations that can be performed on directories. Then various directory structures are described. Later, different ways are given to represent file paths and to share files.

3.4.1 Basics

- "A **directory** is a container for other files and sub-directories."
- In GUI based system, directories are generally represented as **folders**.
- Directories are used to group and organize files & other directories called sub-directories.
- Directories provide a hierarchical file structure.
- A directory is a file whose data is a sequence of entries. Each entry contains a file name and other information about a file, such as file attributes.
- Generally, a directory entry contains a file name and a file identifier. This file identifier, in turn, specifies a location where the file attributes are stored.
- Any directory contains two special entries: dot ('.') and dotdot ('..'). These two entries refer to the current and parent directories, respectively.
- An operating system uses directory entries to map file names to file identifiers.
- Directories are stored on secondary storage devices, such as disks, along with regular files. Information stored in directories must be fetched into the main memory before using it.

3.4.2 Directory Structures

"Directory structures refer to the way how directories and files are organized." There are five different types of directory structures available. All of them have their own advantages and disadvantages. These directory structures are given below:

- Single-Level Directory
- Two-Level Directory
- Tree structured Directories
- Acyclic Graph Directory
- General Graph Directory

The description of each of these directory structures is given below:

3.4.2.1 Single-Level Directory :

- This is the simplest directory structure.
- All files are contained in the same directory. Sometimes it is called the root directory.
- The directory can contain only files. (It cannot have other sub-directories.)
- This type of structure was used in early personal computers. Even the first supercomputer, the CDC 6600, used this type of directory structure for simplicity.
- The following figure represents this directory structure. In this figure, the directory is represented using a rectangle, while files are represented using circles. Here, the single directory contains nine files named cat, bo, a, test, data, mail, cont, hex and records. (Remember: names written in the rectangle are file names.)

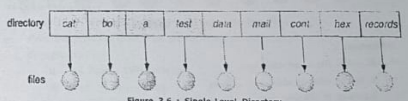


Figure 3.6 : Single-Level Directory

➢ **Advantages :**
- This type of structure is very simple.
- As there is only one directory, searching a file is very quick.

➢ **Disadvantages :**
- It is not suitable for multi-user systems. Different users may provide the same file name for different files, possibly overwriting others' files. This is called a file name collision.
- It is not even suitable for a single-user system when the number of files becomes too large. The user cannot remember the names of all files.
- Different files cannot be grouped.

3.4.2.2 Two-Level Directory :

- It gives each user a private directory.
- This is to avoid file name collisions among different users
- Two levels of directories are used here: The first one is a root directory or master file directory (MFD), and the second is a user file directory (UFD) or, simply, a user directory.
- User directories can contain only files. They cannot have sub-directories.
- The following Figure 3.7 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles.
- Here, the root directory contains four subdirectories related to four different users named user 1, user 2, user 3 and user 4. 'user 1' directory contains four different files named cat, bo, a and test. Similarly, directories 'user 2', 'user 3' and 'user 4' contain two, two, and three files, respectively.

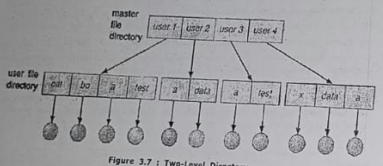


Figure 3.7 : Two-Level Directory

➢ **Advantages :**

- It avoids file name collisions.
- It can be used in multi-user systems.
- Searching is efficient here, as the user has to search in its own single directory.

➢ **Disadvantages :**

- It is not suitable for users having a large number of files.
- Different files cannot be grouped.
- One extra system directory is required to store system files.

3.4.2.3 Tree Structured Directories :

- It permits users to create their own sub-directories.
- This is also called a hierarchical file system. More than two levels of directories are possible.
- Directories can contain files as well as sub-directories also.
- Each file contains a unique file path. (A file path is described in section 3.4.3).
- The following Figure 3.8 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles. The file path for a file named 'spell' is also given in the figure.

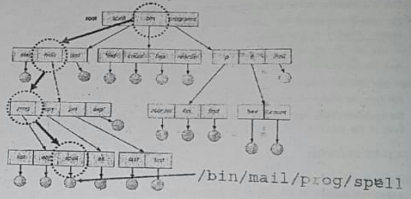


Figure 3.8 : Tree structured Directory

➢ **Advantages :**

- It avoids file name collisions.
- It can be used in multi-user systems.
- Different files can be grouped here.

➢ **Disadvantage :**

- Sharing files and directories is not possible here.

3.4.2.4 Acyclic Graph Directory :

- It permits users to create shared files and directories.
- Shared files and directories can be created using the 'link' operation. This operation allows a file or a directory to appear in more than one directory.
- Directories can contain files as well as sub-directories also.
- A file may contain more than one file path. (A file path is described in section 3.4.3).
- This directory structure does not contain cycles.
- The following Figure 3.9 represents this directory structure. In this figure, directories are represented using rectangles, while files are represented using circles. Also, the figure shows that a file named 'count' is shared from two different locations (directories), 'dict' and 'spell'.

➢ **Advantage :**

- It allows sharing of files and directories.

➢ **Disadvantages :**

- The deletion of a shared file or directory is complex. When such file/directory is deleted, directory entries from all related directories should be deleted.
- A file may have multiple file paths. This may create problems in some operations. For example, if a user copies all files to backup storage, then such files are copied multiple times.
- It is difficult to ensure that there are no any cycles in a directory structure.

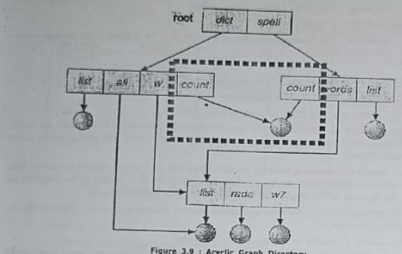


Figure 3.9 : Acyclic Graph Directory

3.4.2.5 General Graph Directory :

- It is the same as an acyclic graph directory, but it also allows cycles in a directory structure.
- The following figure represents this directory structure. In this figure, directories are represented using squares, while files are represented using circles.

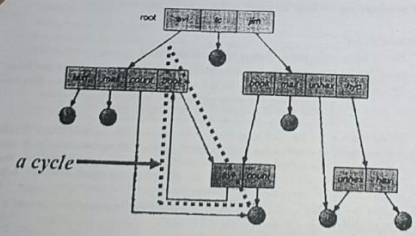


Figure 3.10 : General Graph Directory

- **Advantage :**
  - It allows sharing of files and directories.
- **Disadvantage :**
  - Searching should be done carefully. Else, it may get into infinite loops.

3.4.3 File Paths

"A file path is a character string divided into separate components by a special character such as a slash ('/')". It is used to uniquely specify a location of a file or a directory in a directory structure. A file path is also called a path name. Here, separate components are directories, except the last component can also be a file.

Different files may contain the same file names, but file paths are unique per file. Generally, each file has a single file path. But, a file can have more than one file path if shared from two different locations. (This is described in the next section).

Special characters which are used to separate different components depend on the operating systems. For example, UNIX uses a slash ('/'), while Windows uses a backward slash ('\') for the same purpose.

Examples of a file path are:

i) C:\ Progs \ CPP \ test.cpp .... on Windows environment.
ii) / user / home / test.c .... on UNIX environment.

There are two different methods to specify file paths:

1. **Absolute File-paths :**
   - It starts with the root directory (or drive name) of a file system.
   - Here, file paths are given with respect to the root directory.
   - Examples are:
     i) C:\ Progs \ CPP \ test.cpp
     ii) / user / home / test.c
2. **Relative File-paths :**
   - It starts with the current directory designated by a user.
   - Here, file paths are given with respect to such current directories.
   - A dot ('.') and dotdot ('..') can be used as a current directory.
   - Examples are: Consider that two files test.cpp and test.c contains the absolute file path as given above.
     i) If 'C:\Prgs' is a current directory, then the path for file test.cpp can be given as **CPP\test.cpp**
     ii) If '/user/home' is a current directory, then file test.c can be accessed directly using its name only **test.c** or as **./test.c** (dot represents current directory).

3.5 ACCESS METHODS

In simple terms, 'to access' means 'to obtain something for some particular use'. So, file access means to obtain information from a file for some use.

Files are used to store various kinds of information. When there is a need for this information, it must be accessed and read into the main memory of the computer system. (Remember that access from main memory is faster than disks.)

There are several ways to access the information from the file. Some systems provide only one access method for files, while others provide many different access methods. Choosing the right one for a particular application is a significant design problem.

There are mainly four different types of access methods. They are as given below:

- Sequential Access
- Direct Access / Random Access / Relative Access
- Indexed Access
- Indexed Sequential Access

A description of all these access methods is given in the following sections.

3.5.1 Sequential Access

- Information in a file is accessed in order starting from the beginning of a file.
- All bytes or records in a file are sequentially read one after another.
- It is not possible to access information out of order here.
- One file pointer is maintained here. It specifies the location of the next read or write operation in a file. Different operations use this file pointer in a different manner, as given below:

1. **Open** : Places the file pointer at the beginning of a file.
2. **Read** : Reads a record (or byte) from the current position of the file pointer and automatically advances it to point to the next record.
3. **Write** : Writes a record (or byte) at the end-of-file (eof) and automatically advances file pointer and end-of-file.
4. **Seek** : Skips 'n' records forward or backward.

- Compilers and editors usually access files in this fashion.
- This method is more convenient for storage devices like magnetic tapes rather than disks.
- The following figure depicts this scheme.

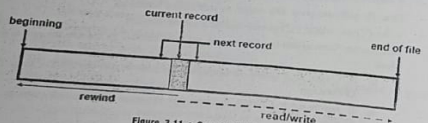


Figure 3.11 : Sequential Access

- ➢ **Advantage :**
  - **Simplicity** : This is the simplest access method.
- ➢ **Disadvantage :**
  - **Inefficiency**: An average access time of a record in a file is equal to the time to access half of the file.

3.5.2 Direct Access / Random Access / Relative Access

- Information in a file is read in no particular order.
- It is possible to access information out of order here.
- Here, the file is viewed as a numbered sequence of blocks or records of fixed length. All blocks or records can be accessed in random order. Because of this reason, this method is also called Random Access.
- It is necessary to provide a block (or record) number with read, write and other operations. This is called a **relative block number**. It starts from '0' or '1'. It is relative to the beginning of a file. Because of this reason, this method is also called **Relative Access**.
- This method is essential for many applications, such as database applications. For example, An Airline reservation.
- This method is suitable for storage devices like disks.
- It is easy to simulate sequential access on direct access. But, to simulate direct access on sequential access is difficult.

- ➢ **Advantage :**
  - **Efficient** : Any record can be accessed directly. It is not necessary to read the entire file starting from the beginning to access some particular record as in sequential access.
- ➢ **Disadvantage :**
  - Users may not know the record number to be accessed from a file.

3.5.3 Indexed Access

- Here, a file is considered a sequence of records (or blocks).
- In addition, an index for a file is constructed. Each entry in the index contains two things: i) the key field of a record and ii) the pointer (physical address) to that record. This is given below.

Key field of a record	Pointer to that record

- All the entries in the index are sorted based on the key field. The key field is used to identify a record among all records uniquely. For example, a student ID number can uniquely identify a record related to a particular student. Here, uniquely means if a user searches for a student having ID number 'st001', he will find one and only one record related to that ID. It is not possible that there is more than one record containing the ID number 'st001'.
- Now, to find a record in the file, the index is searched first. The pointer from the index is then used to access the record directly.

- The following Figure 3.12 depicts this scheme.

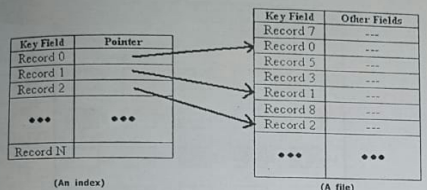


Figure 3.12 : Indexed Access : An index and a file

➢ **Advantage :**
- This method is very suitable for applications like database applications.

➢ **Disadvantage :**
- With a large file, the index itself may become too large. And it becomes difficult to keep such large indexes in the main memory.

3.5.4 Indexed Sequential Access

- This method works in a way similar to the indexed access method. But, it uses multiple indices (plural of index...!).
- Here, the main index is divided into different parts called secondary indices. A small master index is constructed, which contains some limited records. It points to the disk blocks of secondary indices. These secondary indices, in turn, point to the actual blocks (or records).
- To find a record in the file, the master index is searched first. It gives a pointer to the secondary index. Now, the secondary index is searched. The pointer from this secondary index is then used to access the record. This way, any record can be located by at most two direct-access reads.
- The following Figure 3.13 depicts this scheme.

➢ **Advantage :**
- It overcomes the problem of the large index. Here, only a (small) master index needs to be in the main memory all the time. Other secondary indices can be fetched in the main memory as per requirements.

➢ **Disadvantage :**
- To locate any record, two read operations are required: one for the master index and the other for the secondary index. This can be a burden for small files.

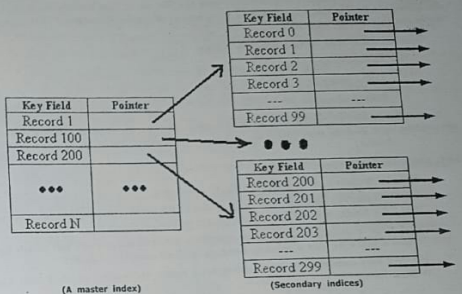


Figure 3.13 : Indexed Sequential Access : A master index and secondary indices

3.6 DISK SPACE ALLOCATION METHODS

Till now, a file system is described from the user's point of view. Users are concerned with how files are named, what operations are allowed on them, what the directory structure looks like and so on. Users need convenience in using the file system.

This section describes the file system from the system's point of view. Here, efficiency and reliability are major concerns.

As described earlier, files are stored on disks. Files contain data as well as system-specific attributes. Here, data means actual data contents stored in files. Attributes contain information about files.

When a file is created, storage space is allocated to it. Also, when new data is added to an existing file, the file size grows and needs extra storage space. Similarly, storage space is released when a file is deleted or data from a file is deleted.

There are two primary goals. These should be fulfilled while allocating space on the disk to files. These two goals are as below:

- Disk space should be utilized effectively.
- Files should be accessed quickly.

While allocating space to files, the system must track which disk blocks go with which files. Or in other words, which file contains which disk blocks.

Here, a disk is considered a collection of fixed-size logical blocks. Size varies from system to system. But, generally, it will be 512 bytes, 1 KB, or more. These blocks are numbered to identify them uniquely on a disk. Numbering starts from '0' or '1' to some maximum.

Whenever there is a need to allocate storage space to a file, one or more free disk blocks are found and allocated to the file. Similarly, whenever a file is deleted, allocated disk blocks are freed. Such disk blocks can be reused for other files.

There are mainly three different allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

The following sections describe these methods in detail. They describe in which manner disk blocks are allocated, what directory entry looks like, advantages, disadvantages and so on.

3.6.1 Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Thus, on a disk with blocks of 1-KB, a file of size 50-KB would contain 50 consecutive blocks. With 2-KB blocks, it would have 25 consecutive blocks.
- When a file is created, a disk is searched to find a chunk of free memory with enough size to store a file. Once such a chunk is found, the required memory is allocated.
- Directory entry contains a file name, starting block number and length of a file. It looks like as given below:

File Name	Starting Block Number	Length

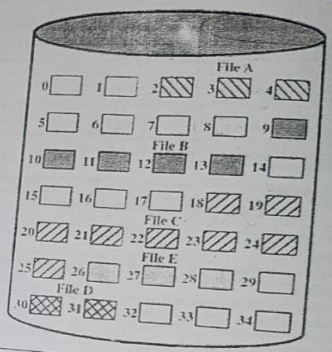
- This method is widely used on CD-ROMs. Here, all the file sizes are known in advance. Also, they will never change during subsequent uses of CD-ROM.
- The following Figure 3.14 depicts such type of allocation method. It is depicted along with directory entries for five files.

Advantages :

- Simple to implement. Information here required is only two things: one, starting block number of a file; and second, the length or size of the file as a total number of blocks.
- File access is quick. All the data blocks are on the same or neighbor tracks of the disk, requiring less seek time.

Disadvantages :

- Finding free space for a new file is time-consuming. This requires searching an entire disk until the required free memory is found.
- If the size of an existing file increases, it may not be possible to accommodate such an extension.
- External fragmentation is possible. When a file is deleted, its blocks are freed, leaving a hole on the disks. With time, the disk will consist of files and holes. Such holes may be too small to accommodate new files and will waste space on the disk. This problem is known as external fragmentation. The solution to this problem is the de-fragmentation or compaction of a disk.



File Name	Starting Block Number	Length
File A	2	3
File B	9	5
File C	18	8
File D	26	2
File E	30	3

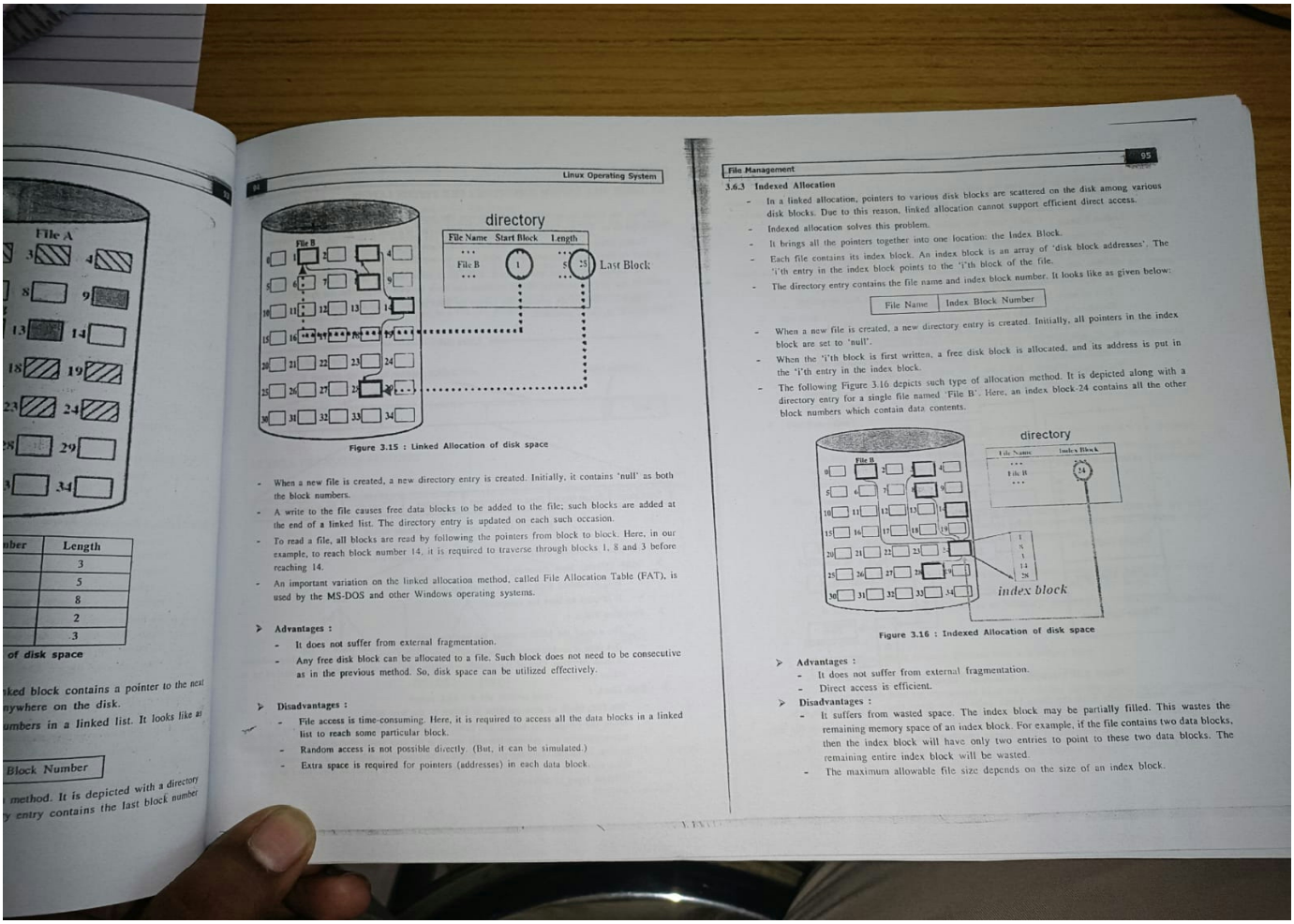
Figure 3.14 : Contiguous Allocation of disk space

3.6.2 Linked Allocation

- Here, each file is a linked list of disk blocks. Each linked block contains a pointer to the next block in a list. These disk blocks may be scattered anywhere on the disk.
- The directory entry contains the start and last block numbers in a linked list. It looks like as given below:

File Name	Start Block Number	Last Block Number

- The following Figure 3.15 depicts such type of allocation method. It is depicted with a directory entry for a single file named 'File B'. Here, the directory entry contains the last block number as well as the length of a file.


File A
3
4
8
9
13
14
18
19
23
24
28
29
34
nber
Length
3
5
8
2
3
of disk space
nked block contains a pointer to the next
nywhere on the disk.
umbers in a linked list. It looks like as
Block Number
method. It is depicted with a directory
y entry contains the last block number
84
Linux Operating System
directory
File Name
Start Block
Length
File B
1
5
Last Block
Figure 3.15 : Linked Allocation of disk space
When a new file is created, a new directory entry is created. Initially, it contains 'null' as both the block numbers.
A write to the file causes free data blocks to be added to the file; such blocks are added at the end of a linked list. The directory entry is updated on each such occasion.
To read a file, all blocks are read by following the pointers from block to block. Here, in our example, to reach block number 14, it is required to traverse through blocks 1, 8 and 3 before reaching 14.
An important variation on the linked allocation method, called File Allocation Table (FAT), is used by the MS-DOS and other Windows operating systems.
Advantages :
It does not suffer from external fragmentation.
Any free disk block can be allocated to a file. Such block does not need to be consecutive as in the previous method. So, disk space can be utilized effectively.
Disadvantages :
File access is time-consuming. Here, it is required to access all the data blocks in a linked list to reach some particular block.
Random access is not possible directly. (But, it can be simulated.)
Extra space is required for pointers (addresses) in each data block.
File Management
95
3.6.3 Indexed Allocation
In a linked allocation, pointers to various disk blocks are scattered on the disk among various disk blocks. Due to this reason, linked allocation cannot support efficient direct access.
Indexed allocation solves this problem.
It brings all the pointers together into one location: the Index Block.
Each file contains its index block. An index block is an array of 'disk block addresses'. The 'i'th entry in the index block points to the 'i'th block of the file.
The directory entry contains the file name and index block number. It looks like as given below:
File Name
Index Block Number
When a new file is created, a new directory entry is created. Initially, all pointers in the index block are set to 'null'.
When the 'i'th block is first written, a free disk block is allocated, and its address is put in the 'i'th entry in the index block.
The following Figure 3.16 depicts such type of allocation method. It is depicted along with a directory entry for a single file named 'File B'. Here, an index block 24 contains all the other block numbers which contain data contents.
directory
File Name
Index Block
File B
24
index block
Figure 3.16 : Indexed Allocation of disk space
Advantages :
It does not suffer from external fragmentation.
Direct access is efficient.
Disadvantages :
It suffers from wasted space. The index block may be partially filled. This wastes the remaining memory space of an index block. For example, if the file contains two data blocks, then the index block will have only two entries to point to these two data blocks. The remaining entire index block will be wasted.
The maximum allowable file size depends on the size of an index block.

➢ **Solutions :**

Here, the problem is: how large the index block should be. If it is too large, it may waste space. If it is too small, it cannot accommodate enough pointers for a large file. There are some solutions to this problem, as depicted below:

- **Linked Scheme :**
  - An index block is generally of one disk block size.
  - For larger files, more than one index block can be used by linking them together as a linked list.
- **Multilevel Index / Hierarchical Indexing :**
  - Two or more level of index blocks is used here.
  - First-level index blocks point to a set of second-level index blocks.
  - Second-level index blocks point to disk blocks containing file data.
  - For larger and larger files, levels can be increased.
- **Combined Scheme :**
  - The following Figure 3.17 gives an idea about this scheme.

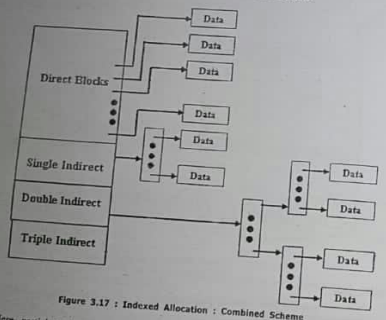


Figure 3.17 : Indexed Allocation : Combined Scheme

- Here, partial entries from the index block point to the direct block containing file data.
- Partial entries point to indirect blocks, which are index blocks. They do not contain file data. But they contain the addresses of disk blocks (pointers) that do contain file data.
- UNIX uses such type of combined scheme.

3.7 LINUX FILE SYSTEM STRUCTURE / FILE SYSTEM LAYOUT

Files are stored on secondary storage devices like hard disks, CD-ROMs, DVDs, magnetic tapes, pen drives and so on. The most common secondary storage device is a hard disk, simply called a disk nowadays.

Today, computer systems come with disks with Gigabytes and Terabytes of storage capacities. They are used to store large amounts of information in various files.

To manage such large storage space, disks are divided into one or more partitions. Such types of partitions are known as drives. Every partition can have its own independent file system.

Figure 3.18 depicts the possible file system (or disk) layout. It may vary from system to system. But, here general layout is described.

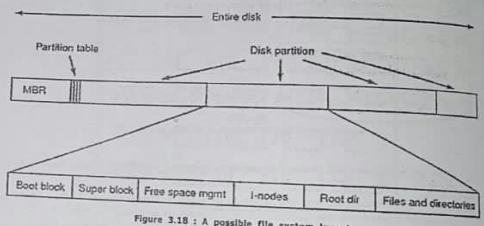


Figure 3.18 : A possible file system layout

Various parts of the file system layout are described as given below:

➢ **MBR (Master Boot Record) :**
- It is a starting sector (sector 0) of the disk.
- It is used to boot the computer.

➢ **Partition Table :**
- The end of the MBR contains the partition table.
- This table is used to locate various partitions on the disk. It gives starting and ending addresses of each partition. One of the partitions contains the operating system. It is marked as an active partition.

➢ **Boot Block :**
- The first block of each partition is known as the boot block.
- When the computer is booted, the MBR is executed by the BIOS. The MBR locates the active partition and executes the code stored in the boot block of that partition. This code, in turn, loads the operating system stored in that partition. After this, an operating system takes control of the computer system. And various system processes are started to provide different types of services.

➢ **Super Block :**
- It contains all the
- It contains inform
  or the size of the

➢ **Free Space Managem**
- It keeps the info

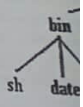
➢ **I-nodes :**
- An i-node (index
  contains all the
  data contents of
- There will be

➢ **Root Directory :**
- It contains the

➢ **Files and Directori**
- This is a remo
  that drive (par

3.8 LINUX FILE SYSTEM

Linux uses a hierarchica
tree-like structure. The follo
system tree.



Key features of th

➢ **Root directo**
- The top
  by "/

- **Super Block :**
  - It contains all the key parameters of the file system.
  - It contains information such as file system type, the number of blocks in the file system or the size of the file system, and other key administrative information.
- **Free Space Management :**
  - It keeps the information about free blocks in the file system.
- **I-nodes :**
  - An i-node (index-node) is a data structure containing all the information about the file. It contains all the attributes related to a file and pointers to blocks, which contain the actual data contents of a file.
  - There will be one i-node per file.
- **Root Directory :**
  - It contains the top of the file system tree.
- **Files and Directories :**
  - This is a remaining part of a partition. It contains all the directories and files stored on that drive (partition).

3.8 LINUX FILE SYSTEM FEATURES

Linux uses a hierarchical directory structure similar to UNIX. Here all files are organized in an inverted tree-like structure. The following Figure 3.19 shows this type of directory structure. It is also called a file system tree.

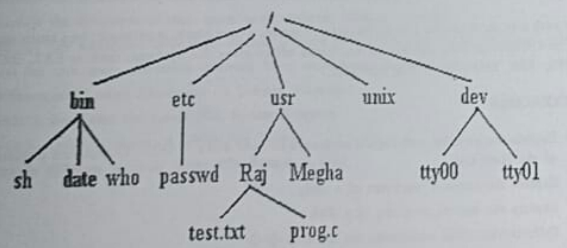


Figure 3.19 : A file system tree.

Key features of the Linux file system are as follows:

- **Root directory :**
  - The top of the hierarchy is called a "root" directory or simply a "root". It is denoted by " / ".

- **File Types :**
  - Files in Linux can be regular files, directory files or device special files.
  - Regular files are used to store user information. They can be text files like source code files or binary files like pictures, videos and object files.
  - Directories are used to store files and sub-directories.
  - Various physical devices such as disks, printers, scanners, CD-ROMs, and memory are considered files also. The kernel maintains special device files, one such file per device, to represent that device. All operations related to that device are performed via reading or writing to such special files.
- **File Path :**
  - The file path describes the location of a file in a hierarchy. Linux uses a forward slash " / " to separate components. (In contrast, Windows uses backward slash " \ " fot this purpose.)
  - All the absolute path names start from a root ("/"). For example, in Figure 3.19, an absolute path for a file *test.txt* can be given as: / usr / Raj / test.txt
  - For the same file, if the current working directory is *usr*, then the relative path can be given as: Raj / test.txt
- **File Extension :**
  - In Linux, the file extension is not mandatory. A file can exist even without a file extension.
  - This may create confusion between the directory and regular file while working with the shell. Such problem does not arise while working with a graphical file system as it symbolizes directories as folders.
- **Case Sensitivity :**
  - In Linux, the file system is case-sensitive. It distinguishes between lowercase and uppercase file names. For example, test.txt and Test.txt refers to two different files. This rule is also applied to directories, Linux commands and variable names.
- **Partitions or Drives :**
  - In contrast to Windows, Linux does not use drive letters to organize the partitions or drives. Partitions or drives are recognized using file names similar to directories.
  - In Linux, a partition/drive, a device, or an "ordinary" directory are addressed using the same manner.
- **Hidden Files :**
  - Linux distinguishes between standard files and hidden files.
  - The configuration files are kept as hidden files in Linux.
  - These hidden files are represented by a dot (.) before the file name, such as ".sample.txt".
  - While working with the shell, specific commands can be used to access hidden files. Similarly, a view needs to be changed if working with a graphical file system.

3.9 LINUX FILE SYSTEM TYPES

Linux offers many file systems, such as EXT, EXT2, EXT3, EXT4, JFS, XFS, ReiserFS, BtrFS, and ZFX. These file systems are briefly introduced below.

- **EXT :**
  - The file system EXT stands for Extended File System.
  - It was primarily developed for MINIX operating system. The EXT file system is an older version. It is no longer used due to its limitations as Linux grew in popularity and usage.
- **EXT2 :**
  - EXT2 (Second Extended File System) is the first Linux file system. It was introduced in 1993.
  - EXT2 file system can be up to 32TB in size. An individual file can be up to 2 TB in size.
  - It is suitable for flash drives and USB drives.
  - It is likely to become corrupt during power failures, and computer crashes when data is being saved to the disk. It also faces data fragmentation issues and reduces performance.
- **EXT3 :**
  - EXT3 (Third Extended File System) is an upgraded version of EXT2. It was introduced in 2001.
  - The total filesystem size and individual file size remained the same as with EXT2.
  - A directory in EXT3 can have up to 32,000 subdirectories.
  - It allows journaling. Journaling creates a separate area of the filesystem where all file changes are tracked. The Journal can then be used to restore data in case of a power failure or system crash.
  - It does not support file recovery and disk snapshot. So it is not suitable for servers.
- **EXT4 :**
  - EXT4 (Fourth Extended File System) was introduced in 2008. It is the default file system in Linux distribution nowadays.
  - EXT4 file system can be up to 1 EB (Exabyte) in size. An individual file can be up to 16 TB in size. A directory in EXT4 can have up to 64,000 subdirectories.
  - It reduces fragmentation issues and increases performance. It provides an option to turn off the journaling feature.
  - Ext4 file system is the faster file system among all the Ext file systems. It is a compatible option for SSD (solid-state drive) disks.

Apart from the above general file systems, there are some other file systems developed for specific Linux distributions. These includes

- JFS (Journaled File System): developed by IBM for AIX Unix.
- XFS (Non-EXT File System): developed for Red Hat Enterprise Linux.
- ReiserFS: earlier used as the default file system in SUSE Linux.
- BTRFS (B-Tree File System): designed by Oracle and released in 2009.
- ZFX (equivalent to Zettabyte file system): developed by Sun Microsystems.

3.10 SUMMARY

In modern computer systems, hard disks are the major secondary storage devices. Hard disks have almost totally outdated earlier drums and magnetic tapes. Physically hard disk is considered in the form of cylinders (tracks), heads and sectors, while logically, it is considered as a large one-dimensional array of logical fixed-size blocks. Disk I/O involves a disk controller and requires an address location from where the read/write operation is to be done.

A file can be considered a sequence of logical records. A logical record may be a single byte, a fixed or variable length line, or a more complex data item. These logical records are grouped and packed into fixed-size blocks on disk.

From a user point of view, a file system is a collection of files, directories, and operations on them. Files contain various attributes, and users can perform various operations on files. Directories are used to organize files. Most modern file systems support a hierarchical (tree-structured) directory system, in which directories may contain files and sub-directories. Files can be accessed in various ways, but direct access is the most common method.

From a system point of view, a file system can be considered an entity that allocates storage space for files on disk, provides safety, organizes files & directories, and so on. In short, it does everything necessary for a file to be there. Among various allocation methods, contiguous allocation is suitable for devices like CD-Rom; linked allocation is used on MS-DOS and Windows along with File Allocation Tables, while indexed allocation is used on UNIX by using i-nodes.

Everything in Linux is treated either as a file or as a process. As devices are treated as files in Linux, some regular file related commands could also be used with devices, making the interface consistent. Files and directories are maintained in a hierarchical structure. Here, the root directory doesn't start with any drive letter such as C:, D:, and so on, as in Windows. But it starts with a slash ('/'). All the partitions (drives) of the disk are treated as separate logical devices and attached to this hierarchical structure as files.

For each disk partition, the Linux file system contains a boot block, superblock, free space management, i-nodes, root directory, files and directories. Linux offers many file systems, such as EXT, EXT2, EXT3, EXT4, JFS, XFS, ReiserFS, BtrFS, and ZFX.

3.11 EXERCISES

1. Explain the physical and logical structures of hard disks. – OR – Discuss the physical structure of the Hard disk.
2. Explain the physical structure of a disk.
3. Explain the logical structure of a disk.
4. Differentiate: CHS addressing v/s LBA.
5. Explain how disk I/O is performed.
6. What is a file? Why is it required? – OR – Define file in file management.
7. Explain in brief: file system.
8. Explain various file concepts very briefly.
9. Explain the file system from the user and system point of view.

10. What are the
    OR – What
11. List out vari
    – OR – Ex
12. Explain diff
13. What are th
14. Write a sh
15. Explain var
    directory st
16. Explain sin
17. Explain the
18. Explain tre
19. List the di
20. Differentia
21. Write a sh
22. Explain the
23. Explain the
24. Explain the
25. Explain the
26. What are
27. Write a sh
28. Explain the
29. Explain the
30. Explain the
31. Justify: Th
32. List out di
33. Differentiat

...ary storage devices. Hard disks have
... hard disk is considered in the form
...red as a large one-dimensional array
...uires an address location from where

...ecord may be a single byte, a fixed
...cords are grouped and packed into

...rectories, and operations on them
...s on files. Directories are used to
...tured) directory system, in which
... various ways, but direct access

...ity that allocates storage space
...n. In short, it does everything
...uous allocation is suitable for
...ng with File Allocation Tables,

... are treated as files in Linux,
...he interface consistent. Files
...ctory doesn't start with any
...sh ('/'). All the partitions
...is hierarchical structure as

...k, free space management,
...h as EXT, EXT2, EXT3,

...s the physical structure

...nt.

10. What are the file attributes? Describe various file attributes. – OR – List out file attributes. – OR – What are file attributes? Explain various file attributes.
11. List out various file operations and describe each of them. –OR – Explain file operations in detail. – OR – Explain any four file operations.
12. Explain different attributes of files and operations on a file.
13. What are the various file structures available? Explain each in brief.
14. Write a short note on file types.
15. Explain various directory structures with their advantages and disadvantages. –OR – Explain the directory structure of the Operating System. – OR – List out types of directory structures.
16. Explain single-level directory structure.
17. Explain the two-level directory structure.
18. Explain tree structured directory in file management.
19. List the disadvantage of a single level, two level and acyclic graph directory.
20. Differentiate: Absolute file-paths v/s Relative file-paths.
21. Write a short note on: File Access Methods
22. Explain the sequential access method in detail.
23. Explain the direct access method in detail.
24. Explain the indexed access method in detail.
25. Explain the indexed sequential method in detail.
26. What are the various allocation methods? Explain any one of these methods in detail.
27. Write a short note on the disk space allocation method.
28. Explain the allocation of disk space in the Contiguous Allocation method.
29. Explain the allocation of disk space in the Linked Allocation method.
30. Explain the allocation of disk space in the Indexed Allocation method.
31. Justify: The contiguous allocation method is more suitable for CD-ROMs than hard disks.
32. List out disk space allocation methods. Then, explain any one of them.
33. Differentiate: Linked Allocation v/s Indexed Allocation.
34. Write a short note on: Linux File System Structure
35. Explain various features of Linux File System.
36. Explain different types of Linux File System in brief.

❖❖❖

CHAPTER

4. Security a

[** This topic is not part of the GTU syllab
Operating System Security and Protection

4.1 INTRODUCTION

Computers are used to store vital data in the form of files and databases nowadays – across all fields – whether it is medical, finance or anything else – from personal to private companies to government offices. So it becomes the utmost urgency to protect these data.

With advanced technological progress, the earlier mechanisms of appointing a security guard or lock and key are insufficient to provide security and protection. Apart from the data stored in computer systems, computer resources such as memory and processors also need to be secured and protected. As we discussed in the first chapter, the operating system is the resource manager for the computer system. And it is the responsibility of the computer system to provide security and protection.

This chapter discusses various concepts related to operating system security and protection. Both words – "Security" and "Protection" – are used interchangeably in general communication. But in technical terms, they both differ. These both are differentiated at the beginning of the chapter. This follows the description of various aspects of security in operating systems. It describes goals and threats, program threats, system threats, user authentication, various security measures along with security policies and procedures. The later part of the chapter focuses on protection. It covers topics like the need for protection, protection domain, access matrix and access control list.

4.2 **SECURITY V/S PROTECTION

As discussed earlier, "Security" and "Protection" are used interchangeably in general communication. But in technical terms, they both differ. Security refers to the overall problem. Protection mechanism refers to the specific operating system mechanisms used to protect data in the computer system.

➢ **Security in the Operating System**

- Security in the operating system is the process of securing system resources, including the user's programs and data from external interference. External interference can be unauthorized users of other systems.
- **Example :**
- Suppose the system has two users. Then security is concerned with the safety of both these users' programs and data. An external user must not access or modify these programs and data without permission.

➢ **Protection in the Operating System**

- Protection in the operating system is the process of protecting a particular user's programs and data from other authorized users of the system.
- **Example :**
- Suppose the system has two users. Then one user can only access the shared data of the other user. The private data must be protected from each other. Protection refers to this type of concept.

➢ **Differences between Security and Protection**

Security	Protection
- Protects the system resources from external unauthorized users and threats.	- Protects the system resources from the intervention among the different users and processes of a system.
- Controls access to system resources by granting permissions to external users.	- Controls how system resources are shared among operating system users.
- Deals with external threats.	- Deals with internal threats.
- Can be enforced using mechanisms like authentication, encryption, and certification to validate the legitimacy of the external user.	- Can be enforced using mechanisms such as authorization to provide access to resources.

Figure 4.1 Security v/s Protection

4.3 SECURITY IN OPERATING SYSTEM

This section focuses on the security in the operating system. It begins with describing security goals and general threats. This follows a detailed explanation of program threats and system threats. Various mechanisms to provide user authentication are given next. Security measures and security policies and procedures are explained in the later part of this section.

4.3.1 Security Goals and Threats

Security can be enforced by assuring integrity, confidentiality and availability of the operating system. Goals and corresponding threats related to the security of operating systems are listed in the following figure.

Goal	Threat
Data Confidentiality	Exposure of Data
Data Integrity	Tampering with Data
System Availability	Denial of Service

Figure 4.2 Security goals and threats

➢ **Data Confidentiality :**

- Data confidentiality suggests that confidential data should always remain confidential.
- Unauthorized reading of data should not be allowed.

- The owner of the data should be able to specify which other users can view that data. An operating system should ensure that only authorized users should be allowed to view such data.
- Private or confidential data like medical information, financial information, and credit card numbers should not be exposed to unauthorized persons and processes.

Data Integrity :

- Data integrity suggests that the data stored in the system should remain unchanged until the owner decides to change them.
- Unauthorized modification of data should not be allowed.
- The owner of the data should be able to specify which other users can modify (insert, update or delete) that data. An operating system should ensure that any user without sufficient rights should not be allowed to modify such data.

System Availability :

- System availability suggests that the system should remain usable at any time. In other words, system resources should remain accessible to all authorized users.
- No single user or process should be able to consume all system resources – making the system unusable for other users.
- Denial of Service (DoS) attacks are considered a violation of this goal. (DoS is explained in section 4.3.3.)

4.3.2 Program Threats

If a user program is altered and made to perform malicious tasks, it is known as Program Threat. In a computer system, processes and kernel, combined together, carry out user-specified tasks. Program threats occur when a user program causes these processes to do malicious operations. A typical example of a program threat is when a program is installed on a computer, it could store and transfer user credentials to a hacker. There are various program threats. Some of them are as follows:

1. Virus

- A computer virus is a malicious program that is installed into a computer without the knowledge of its user. Generally, it is a small code embedded in another program.
- Computer viruses are similar to biological viruses. They are dangerous. They can replicate themselves on the system. They can infect all the programs and files that are in the system.
- They can modify/delete user files and so the user data. They can make the computer system unusable. They can even crash the entire system.
- They remain inactive unless someone accidentally or knowingly activates them.
- Antivirus software can be used to detect and cope with viruses. This type of software needs to be updated regularly to keep it effective.

2. **Trojan Horse**
   - (Trojan horse story: In ancient times, Greek soldiers were able to win the city of Troy only after hiding in a giant wooden horse supposedly left as a gift and entered a city through it.)
   - A Trojan horse is a program that secretly performs some malicious operation in addition to its legitimate operations.
   - A classic Trojan horse is a login emulator. It captures the username and password of a user. Then it transfers these data to another user. This malicious user can log in to the computer using stolen login credentials and access system resources. Among all these, original users don't realize that their information has been stolen.
   - Unlike viruses and worms, a Trojan horse doesn't replicate itself.
   - To avoid Trojan horse issues, one should not download or install software from an unknown source. One should not open an attachment or run a program sent to you in an email from an unknown source. A Trojan antivirus can be used to cope with a Trojan horse.

3. **Logic Bomb**
   - A logic bomb is a situation in which a program misbehaves only when certain conditions are met. Otherwise, it functions as a genuine program.
   - Logic bombs are harder to detect.
   - A Logic Bomb is a code that does not cause havoc all the time. But it becomes harmful only when a specific set of circumstances occurs. For example, a particular date or time is reached. Or some other specific event occurs.
   - A classic example is a Dead-Man Switch. It checks for a presence of a specific person by observing login activities. If that person does not log in for a long time, the logic bomb goes off and performs some harmful operations.

4. **Trap Door**
   - A legitimate program has a trap door if it performs illegal action without the user's knowledge apart from working as expected.
   - For example, a compiler can have a trap door. So, any programs compiled with that compiler would contain a security hole. This is more dangerous as inspection of the code being compiled would not reveal any problems.
   - Trap doors are also known as back doors.
   - Trap Doors are quite difficult to detect. The source code of all the components of the system needs to be checked to find the Trap door.

5. **Ransomware**
   - Ransomware prohibits users from accessing their systems or personal files. It demands a ransom payment to regain access for them.

4.3.3 System Threats

If a system service or network connection is altered and made to perform malicious tasks, it is known as System Threat.

System threats involve misuse of system services and network connections. System threats alter an environment where system resources and user files can be misused. They can be used to invoke program threats over an entire network, known as program attacks. There are various system threats. Some of them are as follows:

1. **Port Scanning**
   - Port scanning itself is not an attack. But it is a mechanism to detect system vulnerabilities to attack the system.
   - It is a fully automated process that creates a TCP/IP connection to a specific port. Once it is found that some computer is listening or a specific port, the next step is to determine a security flaw that can be exploited.
   - Port scanning can be easily detected and traced. So it is launched through the zombie system. Zombie systems are previously hacked systems. They are used without the knowledge and permission of their owner. So it is essential to safeguard the system against such attacks.
   - Port scanners are available that can be used by administrators to check their own systems. These scanners can report any weaknesses found, but they do not cause any problems. Two such systems are nmap (https://nmap.org/) and nessus (http://www.nessus.org). The nmap identifies which operating systems are used in a network, which firewalls are used, and which services are listening to what ports. The nessus also contains a database of known security holes and identifies if any of them is found.

2. **Worm**
   - The worm is a process that chokes down system performance by using system resources to extreme levels.
   - Worms replicate themselves and make several clones. Each one consumes system resources and prevents other legitimate processes from getting the required resources. Worms infect other computers while remaining active on the infected system.
   - Worms are harder to detect until their extreme replication slows down the system or stops other activities.

3. ***Denial of Service***
   - *Denial of service (DoS) attacks usually prevents users from legitimately using the system.*
   - *DoS attacks are generally network-based.*
   - *They can use so many system resources that the system cannot perform useful work. For example, a repetitive and multiple simultaneously downloading of a file from a website can use all available CPU time. Eventually, the server becomes overloaded and cannot serve other legitimate requests.*
   - *They can disrupt the network by abusing the fundamental functionality of TCP/IP. Eventually, this may halt an entire network. For example, a server needs to acknowledge each incoming request for login. If a large number of repetitive login requests are sent to this server, the system resources of the server get exhausted. Again, the server cannot serve other legitimate requests.*

   - Locking users after a certain number of requests can be used to avoid DoS. For example, an account can be locked after a certain number of failed login attempts.
   - Sometimes a DoS attack can cause damage to the underlying infrastructure also.

4.3.4 User Authentication

User authentication is the process of verifying the identity of a user that requests access to a system, network or device.

User privileges depend upon the user's identity. Based on the user identification, the operating system can apply various authorizations like who can view which data and who can modify which data. The operating system determines who the user is while logging into the system. Credentials like username and password are used for this purpose. Other authentication technologies like biometrics and authentication apps are also used to authenticate user identity.

Note that authentication and authorization differ from each other, as given below.

➢ **Authentication versus Authorization :**
   - Authentication is the process of verifying an identity of a user. A user requires to produce evidence like a password so that the operating system can verify that user's identity.
   - Authorization defines what the authenticated user is allowed to do or access. Authorization has two steps. In the first step, a user is granted privileges. For example, some users can read and write a specific file while others can only read that file. In the second step, the user's access rights are verified when a user needs to access a file.

User authentication requires the user to present a credential to verify identity. These credentials can be based on three things: the user's possession like a key or card; the user's knowledge like username and password; user's attributes like fingerprint or signature.

Various **authentication techniques** are explained below:

1. **Passwords**

Passwords are the most common methods of user authentication. Passwords can be in the form of a string of letters, digits, or special characters. Strong passwords include a combination of all these options.

Users need to provide a username (user id) and password to authenticate themselves. If the username and password match with those stored in the system, the user is considered an authentic user.

Passwords are also used to protect resources in the computer system. For example, a password can be associated with each resource (such as a file). Users need to provide a password to use that resource. If the password is correct, access is granted. Different passwords can be associated with different resources.

➢ **Password Vulnerabilities**

Passwords are the most widely used authentication method. But they have some vulnerabilities, as listed below.

- **Guess :** Passwords can be guessed using intelligent guessing or Brute-force guessing. Intelligent guessing uses some knowledge about the password. Brute-force guessing uses each possible combination of letters, digits and special characters. Commonly used passwords can be guessed easily.

- **Shoulder Surfing :** Passwords can be known by looking over people's shoulders while they are typing their password.
- **Packet Sniffing :** Passwords can be sniffed by putting a monitor on a network connection and reading data transferred over the internet.
- **Written Passwords :** If a password is written down somewhere, it might be read by an unauthorized person or lost.
- **Sharing :** Authentic users may share their passwords with unauthentic users like friends and colleagues.

The conclusion is that passwords have many weaknesses though they are widely used. They are insufficient in protecting crucial data or other resources, specifically if care is not taken to manage passwords.

2. One-Time Passwords (OTPs)

One-time passwords are solutions to the problems like shoulder surfing related to simple passwords. A one-time password can be used only once as per its name.

One-time passwords provide an extra layer of security. To log in to the system with a one-time password mechanism, each time a unique password is required. Once a one-time password has been used, it cannot be reused.

Apart from login purposes, one-time passwords can also be used to provide access to system resources.

One-time passwords may be implemented in several ways, as listed below:

- **Secret Key**

  A special hardware device or a software application can be used that generates a one-time password. The user needs to provide this password during each login.

- **Random numbers**

  Users are given cards that have alphabets and numbers printed on them. For example, a debit card of a bank. The system randomly chooses some of the alphabet at the login time. The user needs to enter numbers corresponding to these alphabets.

- **Network password**

  Unique passwords are generated and sent over a network to the registered mobile number or email id at the login time. The user needs to input this password to log in.

3. Authentication using a Physical Object

This technique requires machine-readable physical objects such as badges, cards and smart cards. These objects need to be inserted into a reader associated with the computer system or terminal for authentication.

In some systems, plastic cards are required to authenticate entry into the organization – like scanning a card at an office door.

In some systems, physical objects are combined with passwords. For example, apart from inserting a card, a user needs to input a password also. This type of two-factor authentication prevents misuse of stolen or lost cards. The use of the debit card at the Automated Teller Machine (ATM) of a bank is an example of such a system.

4. Biometrics

Biometrics authentication relies on the unique biological characteristics of an individual user, like fingerprint and eye retina. Biometrics are hard to forge or duplicate. Also, they are unique among multiple users.

Biometric authentication requires three components: First, a scanner to collect the necessary data of the user. Second, software to convert this data into a form that can be stored and compared. Third, a database that stores information of all authorized users.

Common biometric authentication methods and their unique characteristics are listed below:

- Fingerprint: Patterns of ridges and valleys.
- Face: Location, shape, and spatial relationships of facial components like eyes, nose, lips, and nostrils.
- Iris: Circular colored membrane surrounding the eye's pupil.
- Hand Geometry: Shape of hand, size of a palm, and lengths and widths of fingers.
- Palm Prints: Hand geometry combined with features of fingerprints.
- Gait: A manner in which a person walks.
- Voice or Speech: A speaker's speech patterns – formation of specific shapes and sound qualities.

4.3.5 Security Measures in Operating System

Operating system security refers to practices and measures that ensure the confidentiality, integrity, and availability of operating systems.

Operating system security protects the system from various program threats like viruses, Trojan horses, logic bombs, trap doors, ransomware and system threats like port scanning, worms, and denial of service attacks.

Various measures regarding security are as below. (They can be remembered as **A3 DEF SUP**).

- **Authentication :**
  - The operating system should provide secure authentication to determine the user's identity.
- **Authorization :**
  - The operating system should also provide secure authorization to access resources. The privileges should be granted to users and applications by thorough analysis. No one can harm the system without proper access.
- **Antivirus and malware protection :**
  - Antivirus and anti-malware software should be installed and updated regularly to provide protection against viruses and other malware.
- **Data Backup :**
  - It is always a good practice to have a backup of essential data. If data corruption occurs for any reason, it can be retrieved from the backup.
- **Encrypted Data Transfer :**
  - Encryption and decryption techniques must be used to transfer important data over the internet. This mechanism protects data from unauthorized access even if data is stolen, as it would remain unreadable.

- **Firewalls :**
  - Firewalls should be used to ensure that only authorized users are allowed to access or transfer data.
  - Firewalls are programs (or sometimes devices) that monitor all incoming and outgoing traffic over a network. They can impose local security, like restricting the traffic that can pass between them based on certain criteria. For example, a firewall may allow HTTP requests to pass through a web server and restrict telnet requests and other traffic.
- **Suspicious Emails and Links :**
  - Users must beware of suspicious emails and links as they can cause severe issues by acquiring access to system resources.
- **Use of Secure Wi-Fi Only :**
  - It is always safe to use secure Wi-Fi over free or insecure one. Attackers can transmit harmful programs over the network. They can also monitor and record user activity. These may cause severe issues if Wi-Fi networks are not secure.
- **Personal Information Safeguards :**
  - Personal information and credentials must be shared only with trusted and safe persons. This is helpful to avoid unauthorized access to system resources.
  - Users should also keep their credentials safe to avoid illegal access to resources.

4.3.6 Operating System Security Policies and Procedures

Policy and procedure differ from each other. Policy determines what should be done. The procedure determines how it should be done.

A security policy is a document that specifies the procedures (rules, expectations, and overall approach) used by an organization to ensure the confidentiality, integrity, and availability of the system.

Different organizations can have different security policies based on the technologies they choose to work with.

The following subsections describe various concepts related to security policy, like the importance of security policy and examples of the most common security policies and procedures.

4.3.6.1 Importance of Security Policy

A well-designed and properly implemented security policy can provide various benefits as described below:

1. A security policy provides guidance to implement access rights regarding which privileges to be granted to which users. A security policy can specify the overall intention and expectations of the organization. This provides the security team a starting point from where they can build security controls across different users and groups of users.
2. A security policy provides guidance regarding sharing of resources of an organization. Some questions can be answered based on the security policy of an organization. For example, can a manager share passwords with other employees? Can unapproved software be installed on the computer of an organization? Can a company device be used for personal use?

3. A security policy is helpful to meet regulatory and compliance requirements. Documented security policies are required from legislation as well as regulations and standards point of view. They are also helpful in fulfilling security and data privacy requirements.
4. A security policy also improves organizational efficiency. It helps to prevent confusion among employees regarding access and usage of resources. It helps to provide consistency in monitoring and enforcing compliance.

4.3.6.2 Examples of Common Security Policies and Procedures

Nowadays, security policies are the basic needs of any moderate size organization. There can be some organizations where no rules are specified regarding the use of resources and networks by employees. But there are organizations where every aspect of security is thoroughly managed.

The most common security policies that an organization can require are described below.

- **Acceptable Use Policy (AUP)**
  - Acceptable Use Policy defines the conditions and constraints under which a user can use an organization's resources.
  - For example, users need to follow specific rules to access shared printers over an intranet.
- **Access Control Policy (ACP)**
  - Access Control Policy (ACP) defines the access rights or privileges for users to access resources such as data and information of an organization. This policy also specifies when and how privileges are granted to users as well as removed from the users.
- **Data/Information Security Policy**
  - The Data/Information Security Policy defines data classification, ownership, encryption principles, and data backup methodologies for the organization.
  - This policy also makes users accountable regarding the sensitivity of data and information of the organization, like privacy or confidentiality.
- **Remote Access Policy**
  - The remote access policy defines methods of remotely connecting to an organization's network and accessing resources.
  - This policy is required for organizations that have dispersed networks as a work environment. For example, the HR department and canteen may have different internal networks.
- **Firewall Policy**
  - Firewall policy describes the types of traffic an organization's firewall should allow/restrict.
  - This is one of the most common system-specific policies, and each organization must have this policy. Also, it should be ensured that the firewall is set correctly to monitor all incoming and outgoing traffic.
- **Antivirus Policy**
  - Antivirus policy describes installing and updating specific antivirus software into the computer systems.
  - This policy should be strictly followed for each computer system of the organization to get protection from viruses and other malware programs.
- **Email/Communication Policy**
  - Email/Communication policy defines rules or guidelines to communicate within the organization using organization-specific email domains or some other applications.

4.4 PROTECTION MECHANI

Security secures/protects re
authentication to verify the iden
Protection secures/protects resou
applies authorization to determ

Basically, protection refers
users of the system and their p
printer or logical such as pro

This section discusses va
of the protection. Then, it di
an access control list.

4.4.1 Need of Protection

Protection determines w
two users. Then one user can
from each other.

In reality, computer sy
different operations that c
are listed in Figure 4.3. W
to keep track of access t

Reso
Data
Prog
Proc
Me
Pri
Se
Cl

The following p
- Controls ho
- Ensures th
be set by
- Prevents
accidental
- Deals wi
- Improves

Protection pla
concurrently. Also
Finally, it is up

...ents. Documented security
...ndards point of view. They
...ts.
...ps to prevent confusion among
... provide consistency in monitoring
...dures

... moderate size organization. There can be some
... use of resources and networks by employees. But
...ity is thoroughly managed.
...ganization can require are described below.

...onditions and constraints under which a user can use

...ecific rules to access shared printers over an intranet.

... for users to access
...ecifies when

...ion

...mation
...dentiality.

...f remotely connecting to an organization's network.

... have dispersed networks as a work environment.
...n may have different internal networks.

... organization's firewall should allow/restrict.
... policies, and each organization must have
...wall is set correctly to monitor all incoming

...fic antivirus software into the computer

...ter system of the organization to get

...es to communicate within the
... some other applications.

4.4 PROTECTION MECHANISM

Security secures/protects resources from unauthorized external users. The operating system applies authentication to verify the identity of a user and determines between authorized and unauthorized users. Protection secures/protects resources among different authorized users of the system. The operating system applies authorization to determine what the authenticated user is allowed to do or access.

Basically, protection refers to a mechanism that controls the access to system resources among different users of the system and their processes. Resources can be either physical such as processor, memory, and printer or logical such as program files and data files.

This section discusses various concepts related to protection. It begins with a description of the need of the protection. Then, it discusses two mechanisms to implement protection – protection domains and an access control list.

4.4.1 Need of Protection

Protection determines who can access which resource and how. For example, suppose the system has two users. Then one user can only access the shared data of the other user. The private data must be protected from each other.

In reality, computer systems can have different types of resources. Furthermore, each resource allows different operations that can be performed with that resource. Some of the resources and their operations are listed in Figure 4.3. With these combinations of resources and allowed operations, it becomes essential to keep track of access to these resources.

Resource	Operations...
Data File	Create, Delete, Read, Write, Open, Close
Program File	Create, Delete, Read, Write, Execute, Open, Close
Processor	Execute
Memory	Read, Write
Printer	Write / Print
Semaphore	Wait, Signal
CD/DVD	Read

Figure 4.3 System Resources and Operations

The following points list out the need of protection.

- Controls how system resources are shared among different authentic users.
- Ensures that each shared resource is used only as intended. For this, Access Control Policy can be set by system designers or system administrators.
- Prevents unauthorized users or their processes from accessing resources either knowingly or accidentally. Thus, preventing misuse of system resources.
- Deals with internal threats. Ensures that errant programs cause minimal possible damage.
- Improves reliability by detecting possible unintentional errors in accessing resources.

Protection plays a crucial role in a multiuser environment where several users can use various resources concurrently. Also, note that protection provides mechanisms for enforcing policies and ensuring reliability. Finally, it is up to administrators and users to implement them effectively.

4.4.2 Protection Domain

- **Object :**
  - A computer system can have various resources – physical as well as logical. These resources are considered 'objects'.
  - Each object needs protection in a multiuser environment.
  - Each object is given a unique name (or id) and a finite set of operations to implement a protection mechanism.
- **Access Rights / Privileges :**
  - The operating system grants 'access rights' or 'privilege' to objects to perform operations.
  - An access right specifies the ability to perform an operation on an object.
  - For example, a data file can have read and write access rights. A CD/DVD can have only read access right. Figure 4.3 lists some resources (objects) and their operations (access rights).
- **Subject :**
  - Users that want to access objects are known as subjects.
- **Domain :**
  - A domain is a set of object and access rights pair. It can be defined as a set of {Object, Access Rights} pairs.
  - Each pair in the domain specifies an object and operations that can be performed on it.
  - These types of domains are also called protection domains.
  - Some Domains can be disjoint. Some domains may overlap.
  - An operating system maintains several such domains with different combinations of access rights. The user processes can execute in one of those domains. They can access the objects in that domain according to the given access rights.
- **Example :**
  - The following figure explains the concept of the protection domain.

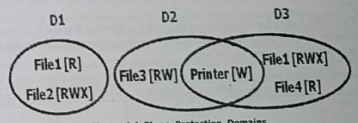


Figure 4.4 Three Protection Domains

- This figure represents three domains, namely D1, D2 and D3. The R, W and X represent access rights Read, Write and eXecute, respectively.
- Domain D1 has two resources: File1 and File2. Processes executing in this domain can read File1 and read, write & execute File2.
- Domain D2 has two resources: File3 and Printer. Processes executing in this domain can read & write File3 and print on a printer.

- Domain D3 has three resources: File1, File4 and Printer. Processes executing in this domain can read, write & execute File1, read File4 and print on a printer.
- Domain D1 is disjoint. Domain D2 and D3 overlap.
- It is not necessary that all domains have separate resources. Different domains can share a resource with the same or different access rights. For example, File1 is present in Domain D1 and D3. Both these domains have different access rights on File1. Similarly, the Printer is shared in Domain D2 and D3.

➢ **Domain Switch :**

- During the execution of a process, it may need to access an object which is in another domain. If it has a right to access that object, it switches to the new domain and accesses the object.
- This process is known as domain switching.
- Each process in Linux has two halves: the user part and the kernel part. When the process does a system call, it switches from the user part to the kernel part. The kernel part has more access rights on objects compared to the user part. For example, the kernel can access all the protected resources. Thus, a system call causes a domain switch.

➢ **Implementation :**

- Access Control Policy can be formed using the protection domain. This policy requires three elements: domains, objects, and access rights.
- This policy can be implemented using an access matrix, access control lists and capability lists. The access matrix and access control lists are discussed in the following subsections.

4.4.3 **Access Matrix

Access Control Policy modeled using protection domain can be implemented using access matrix. This matrix can also be considered as a table. Each row represents a domain. Each column represents an object. The intersection between a row and column, i.e., a cell, contains access rights.

Using the access matrix, the operating system can determine the allowed operations on a resource in a specific domain. Thus, it can control access to resources.

➢ **Example :**

- The access matrix for an example of Figure 4.4 is given in Figure 4.5. In this matrix, three domains are represented using three rows. Five objects are represented using five columns. Access rights are given in the cell.

Objects

Domains	File1	File2	File3	File4	Printer
D1	Read	Read Write Execute			
D2			Read Write		Write (Print)
D3	Read Write Execute		Read		Write (Print)

Figure 4.5 Access Matrix

- Processes executing in domain D1 can read File1 and read, write & execute File2.
- Processes executing in domain D2 can read & write File3 and print on the printer.
- Processes executing in domain D3 can read, write & execute File1, read File4 and print on the printer.

➢ **Example with domain switch :**

- The domain switch can be incorporated into the access matrix by considering *domains* as objects and *switch* operation as an access right.
- Consider the example of Figure 4.4. Suppose the process is allowed to switch domains from D1 to D3, D2 to D1, and D3 to D2.
- The updated access matrix is given in Figure 4.6. Note that a process can switch the domain from D1 to D3, but the reverse is not possible as there is no access right available for this switch.

Objects

Domains	File1	File2	File3	File4	Printer	D1	D2	D3
D1	Read	Read Write Execute						Switch
D2			Read Write		Write (Print)	Switch		
D3	Read Write Execute			Read	Write (Print)		Switch	

Figure 4.6 Access Matrix with Domain Switch

- A process running in domain D1 can switch to domain D3.
- A process running in domain D2 can switch to domain D1.
- A process running in domain D3 can switch to domain D2.

➢ **Disadvantages :**

- The access matrix becomes large in size when it incorporates all domains and objects. So, the entire access matrix cannot be kept in memory. This requires the use of virtual memory techniques.
- Determining specific access rights from a large access matrix becomes difficult and time-consuming.
- Most domains have no access to all objects. So, many cells in the access matrix remain empty, making this matrix sparse. This results in a waste of storage space.
- Common access rights, based on some groups, cannot be specified. For example, if all domains have the same access to some resource, it still needs a separate entry for each domain.

4.4.4 Access Control

Access control list
ACL uses a linked
for each object and so
column are stored in
➢ Example :
- Conside
- This fig
differe
rights.

4.4.4 Access Control List (ACL)

Access control lists resolve issues related to the access matrix.

ACL uses a linked list instead of a matrix or table. The operating system maintains a separate list for each object and so for each column of the access matrix. Only the non-empty cells of the corresponding column are stored in the ACL, discarding blank entries.

➢ **Example :**

- Consider the example given in Figure 4.4. The corresponding ACLs are given in Figure 4.7.
- This figure shows that five separate lists are required to represent the access rights of five different objects. Access control is determined based on the pair of the domain and access rights.

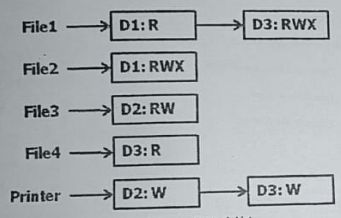


Figure 4.7 Access Control List

- Each domain can correspond to exactly one user.
- File1 is shared between domains D1 and D3. It can be read by a process running in domain D1. It can be read, written and executed by a process running in domain D3.
- File2 can be read, written and executed by a process running only in domain D1.
- File3 can be read and written by a process running only in domain D2.
- File4 can only be read by a process running in domain D3.
- The printer is shared between domains D2 and D3.
- A process running in domain D1 can read File1. Likewise, a process running in domain D2 can read, write and execute File1.

➢ **Advantages :**

- Reduces storage space required to implement Access Control Policy.
- Increases search efficiency. Only a single list is required to check to determine access control on an object.
- Common access rights, based on some groups, can be specified. A separate list of default access rights can be maintained and checked first for this purpose.

4.5 SUMMARY

Security in the operating system is the process of securing system resources, including the user's programs and data from external interference. Security can be enforced by assuring integrity, confidentiality and availability of the operating system. Data confidentiality suggests that confidential data should always remain confidential. Data integrity suggests that the data stored in the system should remain unchanged until the owner decides to change them. System availability suggests that the system should remain usable at any time. In other words, system resources should remain accessible to all authorized users.

If a user program is altered and made to perform malicious tasks, it is known as Program Threat. Examples of program threats include viruses, Trojan horses, logic bombs, trap doors and ransomware. If a system service or network connection is altered and made to perform malicious tasks, it is known as System Threat. Examples of system threats include port scanning, worms, and denial of service attacks.

User authentication is the process of verifying the identity of a user that requests access to a system, network or device. Various authentication techniques include the use of simple passwords, one-time passwords, physical objects, and biometrics.

Security measures include authentication, authorization, antivirus and malware protection, data backup, encrypted data transfer, firewalls, suspicious email and links, secure Wi-Fi and safeguards to personal information.

A security policy is a document that specifies the procedures (rules, expectations, and overall approach) used by an organization to ensure the confidentiality, integrity, and availability of the system. Common security policies include an Acceptable Use Policy, Access Control Policy, Data/Information Security Policy, Remote Access Policy, Firewall Policy, Antivirus Policy, and Email/Communication Policy.

Protection in the operating system is the process of protecting a particular user's programs and data from other authorized users of the system. In fact, protection determines who can access which resource and how.

The protection domain is a set of object and rights pair. It can be defined as a set of {Object, Rights} pair. A computer system can have various resources – physical as well as logical. These resources are considered 'objects'. The operating system grants 'access rights' or 'privilege' to objects to perform operations. Users that want to access objects are known as 'subjects'.

Access Control Policy modeled using protection domain can be implemented using Access Matrix. This matrix can also be considered as a table. Each row represents a domain. Each column represents an object. The intersection between a row and column, i.e., a cell, contains access rights.

Access Control List uses a linked list instead of a matrix or table. The operating system maintains a separate list for each object and so for each column of the access matrix. Only the non-empty cells of the corresponding column are stored in the ACL, discarding blank entries.

6 EXERCISES

1. Differentiate between security and protection.
2. Explain security goals and threats.
3. Write a short note on: Program Threats
4. Write a short note on: System Threats
5. What is user authentication? How does it differ from authorization?
6. Explain various authentication techniques.
7. Explain the use of passwords in the authentication. Why are passwords insufficient in protecting crucial data? – OR – Explain the use of passwords and their vulnerabilities.
8. Explain one-time passwords in brief.
9. Explain authentication using physical objects.
10. Explain authentication using biometrics.
11. Write a short note on: Security measures in operating systems
12. Explain operating system security policies and procedures.
13. What is protection? Why is it required?
14. Explain the protection domain in detail with a suitable example.
15. Write a short note on: Access Matrix
16. Write a short note on: Access Control List (ACL)
17. Justify: Access Control List is better than Access Matrix.
18. Consider the following domains D1, D2, and D3. Also, consider the following resources and their access rights. Provide access matrix and access control list for them.

    D1: Resource1 (RW), Resource3 (RWX)

    D2: Resource2 (RWX), Resource3 (RW)

    D3: Resource1 (R), Resource2 (RW)