# UNIT 5: BASICS OF LINUX FILE SYSTEM

## UNIT STRUCTURE


5.1 Learning Objectives

5.2 Introduction

5.3 Basics of Linux File System

- 5.3.1 ext
- 5.3.2 ext2
- 5.3.3 ext3
- 5.3.4 ext4
- 5.3.5 xfs
- 5.3.6 jfs
- 5.3.7 Reiser FS
- 5.3.8 ISO 9660

5.4 Linux File System Internal Structure

5.5 How Linux Access Files

5.6 Storage of Files

5.7 Linux Directory Structure

5.8 LILO, GRUB Boot Loader

5.9 Basic Types of Files

- 5.9.1 Regular Files
- 5.9.2 Derectory Files
- 5.9.3 Special or Device Files

5.10 Other Common Types of Files

5.11 File Attributes

5.12 Let Us Sum Up

5.13 Further Readings

5.14 Answer to Check Your Progress

5.15 Model Questions


## 5.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- define file and file system

- describe different file system types
- explain the features of different file system and boot loader
- learn about boot block, super block, inode and data block
- describe the journaling concept in file system
- know about various file types used in Linux system
- describe different file attributes.

## 5.2 INTRODUCTION

In the previous units, the features of Linux Operating System and the working of Linux Kernel were described. Also, the redirection and piping concepts in Linux Operating System were discussed. In this unit, we will study the Linux File System and the related properties. Subsequently, the functions of different file systems in Linux will be explained here in this unit. Further, we will study the different types of files, I-node and the file attributes.

## 5.3 BASICS OF LINUX FILE SYSTEM

In Linux, all data and programs are stored in files. No restriction is imposed on the structure of a file. The size of a linux file is the exact size of the contents that stored inside the file. The files in a file system are connected logically. A file system manages how the files are stored and can be retrieved easily. Without a file system, it is not possible for an operating system to know the positioning of a file i.e., where a file starts and where ends and thus the operating system keeps track of how files are organised in a disk partition so that it can be found easily. There are different file systems available in linux operating system. They are different from each other based on the structure, logic and other important properties like flexibility, speed, size etc. Some of the file systems that are being commonly used for better understanding of the functionalities of a file system are described as follows.

Any Linux distribution has different kinds of file system to partition a disk. Some of the commonly used file systems include the following:

- ext

- ext2
- ext3
- ext4
- xfs
- JFS
- ReiserFS
- ISO9660 etc.

### 5.3.1 ext

ext stands for extended file system. It is considered the first file system used in Linux. It was developed by Remy Card in April, 1992 especially for Linux Operating System. It is the first version in the series of extended file systems. It was replaced by ext2 in 1993. The ext file-system uses a system called inodes to track information about the files stored in the virtual directory. The inode system creates a separate table on each physical device, called the inode table, to store the file information. Each stored file in the virtual directory has an entry in the inode table. Linux maintains each inode in the inode table with a unique number (called the inode number), assigned by the file system.

### 5.3.2 ext2

ext2 is meant for second extended file system created for Linux OS and it is an improved version of ext. It was also designed by Remy Card. It is one of the most popular and commonly used file systems in Linux and has been used for many years. Here, the data stored in files are kept in data blocks. All the data blocks have same length. However, the length is not fixed in all ext2 file systems. The size of the file system is depended on the block size which may vary from 1 KB to 8 KB. The maximum size of the ext2 file system is from 2 TB to 32 TB whereas the maximum file size is from 16 GB to 2TB. The size of the filename may be of maximum of 255 characters.

The limitation of ext2 file system is that it takes very long time to recover after an unclean shutdown which may occur due to situations like power failure or system crash. This problem leads to development of the improved version of ext2.

### 5.3.3 ext3

ext3 is the third extended file system in Linux. It is an extended version of ext2 file system which was introduced by Stephen Tweedie in November 2001 with Linux. It overcomes the limitation of ext2 file system by introducing the concept of journaling. Moreover, ext3 also adds two more features which are the growth of online file system and Tree indexing for larger directories. An ext3 file system works similar to ext2 file and an ext2 file system can easily be upgraded to ext3. It uses the same inode table structure as the ext2 filesystem, but adds a journal file to each storage device to journal the data written to the storage device. This Ext3 file system is used as default file system in many standard Linux distributions.

A journaling file system handles a special file known as a journal. It is used to keep track of changes that are not committed to the file system completely. It is used for correcting the inconsistencies that may occur at the time of system crash or power failure. It may keep track of both stored data and metadata. This file system keeps the information in a log area. After the updating of log, the system wrote the actual data in the appropriate areas of the file system. If a file system is crashed out, it can be repaired very quickly back online using the journal log. So, there is a very little chance of data loss in journaling file system.

ext2 file system runs a program called e2fsck (ext2 file system check) at the time of rebooting after an unclean shutdown. It checks whether any problem occurs during the improper shutdown and tries to fix the problems before the file system be mounted. Thus, ext2 tries to maintain the data integrity. But the problem is

that e2fsck checks the entire file system which may take a long time to recover. The main advantage of ext3 file system is that it supports journaling. This file system does not need any checking in the file system at the time of rebooting after an unclean shutdown unlike ext2 file system since it guarantees the internal consistency of the system.

The size of the file system and file size is similar to file system of ext2. The maximum size of the ext3 file system is from 4 TB to 32 TB whereas the maximum size of a file is 16 GB to 2 TB. The filename size is also same as ext2. The disadvantage of the ext3 file system is that it occupies more system memory to store the information and can make the I/O system little slower. In spite of this, many of the distributions use ext3 file system for its faster recovery.

### 5.3.4 ext4

The result of expanding the ext3 file system is the ext4 file system. The ext4 file system was officially supported in the Linux kernel in 2008, and is now the default file system used in most popular Linux distributions, such as Fedora and Ubuntu. It was developed over ext3 in order to increase the size of the file and entire file system to improve the performances. The various features of ext4 file system are as follows:

- **Compatibility:** Any ext3 file system can be upgraded to ext4 easily. So, we can easily increase the size of the file system and improve the performance without doing any formatting or reinstallation of the operating system. Ext4 uses the new data structures for the new data only. The old data can be modified whenever required.
- **Extents:** ext2 and ext3 file systems use an indirect block mapping scheme to keep track of blocks used for particular data. It is not efficient for the larger files. An extent is a collection

of contiguous physical blocks. A single extent can have a contiguous block size of 128 MB where size of a single block is 4 KB.

- **Multiblock Allocation:** In ext file system, a program block allocator decides which free block will be used at the time of writing a new data. But it allows only one block at a time. So, it is not efficient technique for writing a file of large data size. To improve the performance, ext4 uses multiblock allocator which allows to use more than one free block at a single call.
- **Journal Checksumming:** ext4 file system checksums the journal data as it is one of the most used files in the disk. It is used to check whether any journal block has corrupted or not.
- **Delayed Allocation:** In ext4, blocks are not allocated to data immediately. There is a delay in allocating the block unlike ext3 and the files are kept in cache. In this case, block allocator can optimize the situation.
- **No Journal Mode:** A journaling file system keep a track of information of the current changes. For this, it occupies little storage. Some programs may have special requirement where it can run without journal. Accordingly, journaling option can be disabled in ext4 file system.
- **Larger File System and File Sizes:** ext4 file system has significantly larger sizes than ext3 file system. A file in ext4 can have a maximum size of 16 TB and on the other side, the file system has maximum size of 1 EiB or 1 ExbiBytes [1 EiB = 260 bytes].

### 5.3.5 xfs

xfs file system is a high-performance and highly scalable 64 bit system. It supports journaling and thereby it ensures the consistency of the data. It was created by Silicon Graphics and introduced in 1994 for its commercial IRIX Unix system. It was used in Linux

kernel in 2001. This file system is supported by almost all Linux distributions. The xfs file system uses the writeback mode of journaling, which provides high performance and also allows online resizing of the file system. For space allocation, extents with stored data structure in B+ tree is used. One of the unique features of xfs file system is pre-allocation of input-output bandwidth at previously determined rate which may be beneficial for real time applications. xfs file system has block size of 512 bytes to 64 KB.

### 5.3.6 jfs

JFS is considered one of the oldest journaling file systems. It is a 64-bit file system created by IBM. It's first version was developed for AIX operating system of IBM in 1990 known as jfs1. JFS2 was introduced in 2001 with Linux OS. The term "journaled" depicts that the file system keeps the record of ongoing modification of files in a log file. This log file is used to recover the loss that may occur during an improper shutdown. JFS uses the ordered journaling method, storing only the inode table data in the journal, but not removing it until the actual file data is written to the storage device. In addition, JFS file system uses extent-based file allocation, allocating a group of blocks for each file written to the storage device. This method provides for less fragmentation on the storage device. It uses B+ tree to search in directories. Since ext4 shows better performance compared to JFS, so it is not widely used in Linux.

### 5.3.7 ReiserFS

In 2001, Hans Reiser created the first journaling file system for Linux, called ReiserFS. The ReiserFS file system only supports writeback journaling mode, writing only the inode table data to the journal file. Due to this feature, the ReiserFS file system is considered one of the fastest journaling file systems in Linux. Two interesting features available with the ReiserFS file system refer

the following.

- We can resize an existing file system while it's still active, and
- It uses a technique called tail packing, which stuffs data from one file into empty space in a data block from another file and thereby reduces internal fragmentation.

### 5.3.8 ISO 9660

ISO 9660 is a file system created for optical media. It is published by International Standard Organization (ISO). The main aim of this file system is that it can support in any other standard operating system like Linux, Windows and Mac etc. for exchanging the data between the media and the system. This file system standard evolved from the High Sierra Format file system. It has three different levels for files:

- **Level 1** - supports only 8.3 filenames (eight characters with a three-character extension).
- **Level 2** - supports longer 32-character filenames.
- **Level 3** - supports longer 32-character filenames.

## 5.4 LINUX FILE SYSTEM INTERNAL STRUCTURE

Linux files have disk partitions or blocks containing file data. Linux File System is divided mainly into 4 blocks.

- Boot block
- Super block
- inode
- Data Block

Partition is divided into data blocks with of fixed size which are used to store content of file. In case if file is of greater size than of block then it is stored in multiple blocks.

| Boot Block | Super Block | Inodes | | | ... | Data Blocks |

**Boot Block:** It is located in the first few sectors of a file system. The boot block contains the initial bootstrap program used to load the

operating system. Typically, the first sector contains a bootstrap program that reads in a larger bootstrap program from the next few sectors, and so forth. The boot block is usually a part of the disk label, a special set of blocks containing information on the disk layout. The boot block holds the loader to boot the operating system.

**Super Block:** Each filesystem has one super block. The superblock contains the basic information about the entire file system. This includes the size of the file system, the list of free and allocated blocks, the name of the partition, and the modification time of the filesystem.

**Inodes:** Information about each file in the filesystem is kept in a special kernel structure called an inode. The inode contains a pointer to the disk blocks containing the data in the file, as well as other information such as the type of file, the permission bits, the owner and group, the file size, the file modification time, and so on. The inode does not, however, contain the name of the file. The name of each file is listed in the directory the file is associated with. Every file has one inode and vice versa. In linux, every file is recognized with integer number known as inode number. Each inode has an entry in inode table. An inode provides following information about a file:

- File Types (i.e., regular, directory, device file or other)
- File Permission or access modes (i.e., read, write or execute to different users)
- User ID of the owner
- Group ID of the group
- Last access, last modification detail
- Number of soft links and hard links
- Size of the file
- Location or path of the file
- Other related data

We can find the inode number of a particular file using the following Linux command:

**ls -i <filename>**

When a program or a user is referred by its name, the operating

system maps the filename with its corresponding inode. Then, through the inode further operations in the file is performed.

Inodes include pointers to the data blocks. Each inode contains 15 pointers:

- the first 12 pointers point directly to data blocks
- the 13th pointer points to an indirect block, a block containing pointers to data blocks
- the 14th pointer points to a doubly-indirect block, a block containing 128 addresses of singly indirect blocks
- the 15th pointer points to a triply indirect block (which contains pointers to doubly indirect blocks, etc.)

**Data Blocks:** Data blocks start at the end of the inode list and contain file data and directory blocks. An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be of the same kind. The three types of data blocks are:

- Plain data blocks
- Symbolic-link data blocks
- Directory data blocks

Plain data blocks contain the information stored in a file. Symbolic-link data blocks contain the path name stored in a symbolic link. Directory data blocks contain directory entries. fsck can check the validity only of directory data blocks.

Directories are distinguished from regular files by an entry in the mode field of the inode. Data blocks associated with a directory contain the directory entries. Directory data blocks are checked for inconsistencies involving:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers greater than the number of inodes in the file system
- Incorrect directory inode numbers for "." and ".." directories
- Directories disconnected from the file system

## 5.5 HOW LINUX ACCESS FILES

Linux file system is generally a built-in layer of Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file. A file system is designed in a way so that it can manage and provide space for non-volatile storage data. All file systems required a namespace that is a naming and organizational methodology.

In Linux, the file system creates a tree structure. All the files are arranged as a tree and its branches. The topmost directory called the root (/) directory. All other directories in Linux can be accessed from the root directory.

Some key features of Linux file system are as following:

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; it uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, whereas, in Linux, it would be stored in /home/ My Document/ Work.
- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell whether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitivity:** Linux file system is case sensitive. It distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but it is not necessary that a file should have a file extension. While working with Shell, it creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, it symbolizes the files and folders.
- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we

don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name. To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

## 5.6 STORAGE OF FILES

File systems are defined based on where they are used. There are file systems defined for operating systems, networks, databases, and other special-purpose file systems. A filesystem is designed to provide space for non-volatile storage of data. All filesystems need to provide a namespace with a name and organizational methodology. This defines how a file can be named, specifically the length of a filename and the subset of characters that can be used for filenames out of the total set of characters available. It also defines the logical structure of the data on a disk, such as the use of directories for organizing files instead of just lumping them all together in a single, huge conglomeration of files.

Once the namespace has been defined, a metadata structure is necessary to provide the logical foundation for that namespace. This includes the data structures required to support a hierarchical directory structure; structures to determine which blocks of space on the disk are used and which are available; structures that allow for maintaining the names of the files and directories; information about the files such as their size and times they were created, modified or last accessed; and the location or locations of the data belonging to the file on the disk. Other metadata is used to store high-level information about the subdivisions of the disk, such as logical volumes and partitions. This higher-level metadata and the structures, it represents, contain the information describing the filesystem stored on the drive or partition, but is separate from and independent of the filesystem metadata.

File systems also require an Application Programming Interface (API) that provides access to system function calls which manipulate filesystem objects like files and directories. APIs provide for tasks such as

creating, moving, and deleting files. It also provides algorithms that determine things like where a file is placed on a filesystem. Such algorithms may account for objectives such as speed or minimizing disk fragmentation. Modern filesystems also provide a security model, which is a scheme for defining access rights to files and directories. The Linux filesystem security model helps to ensure that users only have access to their own files and not those of others or the operating system itself. The final building block is the software required to implement all of these functions. Linux uses a two-part software implementation as a way to improve both system and programmer efficiency.

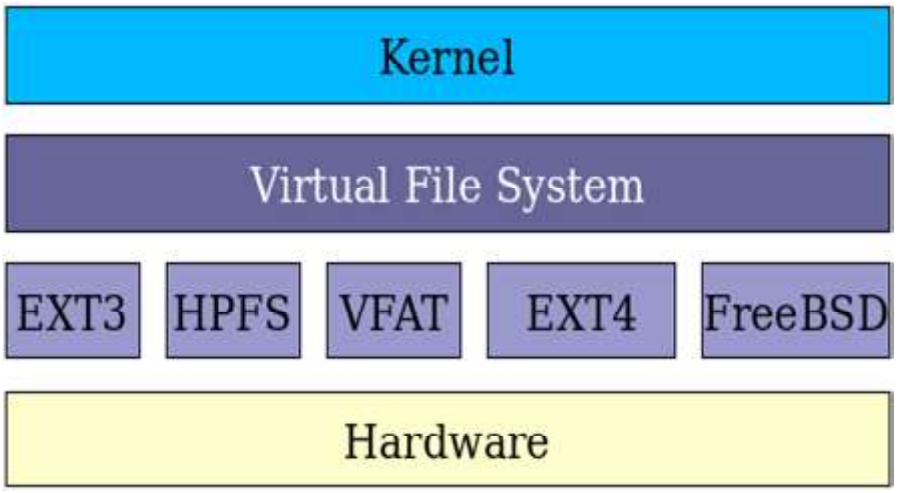



Fig. 5.1: The Linux two-part filesystem software implementation.

The first part of this two-part implementation is the Linux Virtual File System. This virtual file system provides a single set of commands for the kernel, and developers, to access all types of file systems. The virtual file system software calls the specific device driver required to interface to the various types of file systems. The file system-specific device drivers are the second part of the implementation. The device driver interprets the standard set of file system commands to ones specific to the type of file system on the partition or logical volume.

## 5.7 LINUX DIRECTORY STRUCTURE

The File System Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions. It is maintained by

the Linux Foundation. In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices. Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Most of these directories exist in all Unix-like operating systems and are generally used in much the same way. However, the descriptions here are those used specifically for the FHS, and are not considered authoritative for platforms other than Linux. Some standard Linux directories with their description are listed in the following table (Table-5.1).

| Directory | Description |
|---|---|
| / | Primary hierarchy root and root directory of the entire file system hierarchy |
| /bin | Essential command binaries that need to be available in single user mode; for all users, *e.g.*, cat, ls, cp. |
| /boot | Boot loader files, *e.g.*, kernels, initrd. |
| /dev | Device files, *e.g.*, /dev/null , /dev/disk0 , /dev/sda1 , /dev/tty , /dev/random. |
| /etc | Host-specific system-wide configuration files. |
| /home | Users' home directories, containing saved files, personal settings, etc. |
| /sys | Contains information about devices, drivers, and some kernel features. |
| /usr | contains the majority of (multi-)user utilities and applications |
| /var | Contains Variable files such as logs, spool files, and temporary e-mail files. |

Table 5.1: Few standard Linux directories with description

What the file system looks like is represented graphically in the following figure 5.2. Some of these directories have standard files as well like /bin, /etc and /sbin specifically.

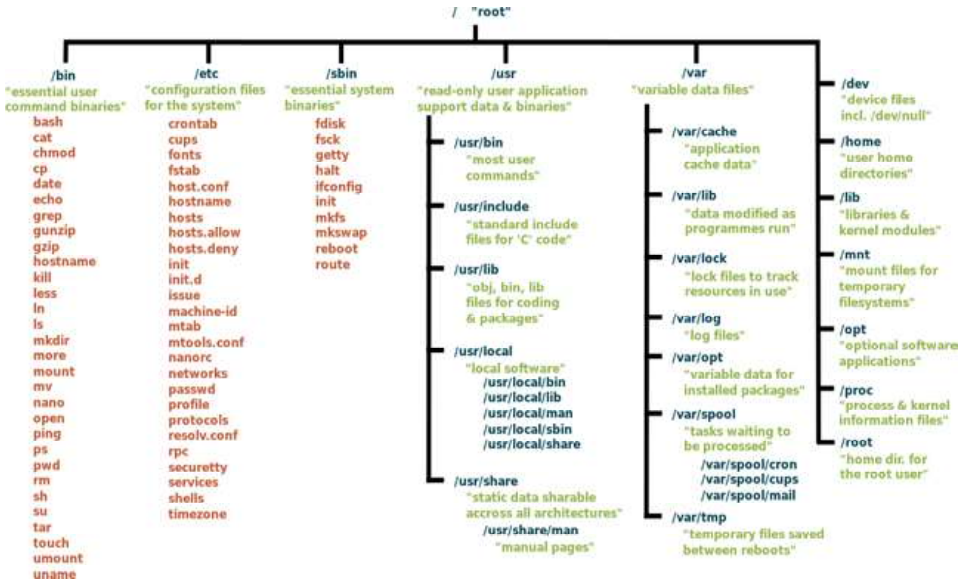



Fig. 5.2 Linux Directory structure

## 5.8 LILO, GRUB BOOT LOADER

A boot loader refers to a program which is responsible for loading the Linux kernel with optional kernel parameters and the Linux initial RAM disk, known as initrd. Linux kernel is the core of the Linux operating system, and it initiates the init process immediately after being loaded. The Linux initial RAM disk provides a temporary storage space for loading critical files into memory before the real root file system can be mounted. On older computers with BIOS (Basic Input/Output System), a boot loader resides in the MBR (Master Boot Record), which occupies the first 512 bytes on a disk, but newer computers with UEFI (Unified Extensible Firmware Interface) store it in a special partition called EFI System Partition.

A boot loader is loaded by BIOS (Basic Input Output System) after a successful POST (Power-On Self-Test), which is a self-test process performed immediately after a computer or other digital electronic device is powered on. There are several boot loaders in Linux system. Among them, two mostly used Linux loader programs are described here.

**LILO (Linux Loader)**

LILO once used to be the most popular Linux boot loader, but it has gradually fallen from beauty because it lacked support for multi-boot

environments. The development of LILO was stopped in December 2015, and the limited support for modern file systems and non-existing support for UEFI both reflect this reality.

**GRUB (GRand Unified Bootloader)**

GRUB is the most popular and the most feature-packed boot loader for the Linux operating system. It is based on the obsolete GRUB Legacy, which was created in 1995 by Erich Boleyn for the operating system GNU/ Hurd. GRUB supports both BIOS and UEFI, and it can handle all popular Linux file systems, including ext4, ReiserFS, VFAT, and XFS. Compared to GRUB Legacy, modern GRUB is cleaner, more powerful, and safer. GRUB has been designed to be compatible with the multiboot specification which allows GRUB to boot many versions of Linux and other free operating systems.

GRUB can also allow the user to choose to boot from among several different kernels for any given Linux distribution. This affords the ability to boot to a previous kernel version if an updated one fails somehow or is incompatible with an important piece of software. GRUB can be configured using the /boot/grub/grub.conf file.

## 5.9 BASIC TYPES OF FILES

Linux operating systems have different type of files. Most of the commonly used file types are regular files, directory files and device files.

### 5.9.1 Regular Files

Regular files or ordinary files are the mostly used files in Linux system. All the text files, binary files, executable program files, image files, shared libraries etc. are known as regular files. Regular files are denoted by a dash ('-') symbol. Regular files can be created using touch, vi, gedit commands.

For listing the regular files following Linux command is used:

**ls - l | grep ^-**

Where grep is a filter which will find all the regular files which are

denoted by '-' symbol.

### 5.9.2 Directory Files

Linux stores all the data and programs in files. These files are contained in directories. A directory contains all the regular files, special files, other folders stored in a physical device. These files are denoted by symbol 'd'. Directories can be listed using the following Linux command:

**ls - l | grep ^d**

Some of the important commands related with directories are given below:

**mkdir :** used for creating a new directory.

**Syntax: mkdir <directory name>**

**rmdir:** used for removing or deleting any empty directory from the system.

**Syntax: rmdir <directory name>**

**cd:** used for changing the directory.

**cd <directory name>**

### 5.9.3 Special or Device Files

Device files or special files in Linux are the files that work as an interface to device drivers. It allows an application to interact with a device via any device driver. Most of the device files are stored in / dev directory in Linux. Each device in the system should have an entry in /dev. These files are created during installing the system and device drivers. There are two different types of device files present in Linux operating systems. These are: character special files and block special files.

- **Character Special Files:** They are also known as character devices. Their behaviour is like serial ports, pipes etc. They provide a direct access to hardware devices. They do not allow any program to perform read or write operation. It is up to the

device. Sometimes character devices are known as raw devices also. Character devices are symbolically denoted by 'c'. To list all the character files locate in /dev directory, following Linux command can be used:

**ls -l /dev | grep ^c**

- **Block Special Files:** They are also known as block devices. These files differ from the character devices in the fact that a block device allows a program to read or write a block of any size. These are buffered access to hardware devices whereas character devices are unbuffered access. Block devices are denoted by 'b'.

To list all the symbolic link files located in /dev directory, following Linux

command can be used:

**ls -l /dev | grep ^b**



**CHECK YOUR PROGRESS**

**Q 1:** Choose the correct options from the followings:

i) Which among the following allows fast file system recovery?

a) Ext2 b) Journaling c) Caching d) Sysfs

ii) Filesystem for CDROM is:

a) Ext2 b) Ext3 c) ISOfs d) Procfs

iii) Which file contains the filesystems to be automatically mounted during boot?

a) /etc/mount b) /etc/fstab c) /etc/inittab d) /etc/boot

iv) Which command is used to check filesystem usage in a system?

a) mount

b) df c) du d) dd

v) In which of the following directory does the configuration files are present?

a) /bin/ b) /root/ c) /etc/ d) /dev/

## 5.10 OTHER COMMON TYPES OF FILES

Apart from these three types of files, Linux system offers some other type of files which are:

- **Symbolic Links:** A symbolic link is a reference to another file located in a disk. This file is stored as referenced file's path. These files are denoted by 'l', also known as soft link. Each soft link contains a different inode value which directs to the original file. A hard link has the same inode value as the original file. For this, it remains linked even the original file is moved to another location unlike a soft link file. To list all the block files located in /dev directory, following Linux command can be used:

  **ls -l /dev | grep ^b**

- **Named Pipes:** It performs communication between two or more processes locally with each other using file. This file acts as a pipe among the processes. These are also known as FIFO. Named pipes are denoted using 'p'. To list all the named pipe files located in /dev directory, following Linux command can be used:

  **ls -l /dev | grep ^p**

- **Socket Files:** Socket files are used for inter process communication. It is used to pass the information between applications. The main difference between socket and named pipe is that a socket can perform communications between processes which are running on two different systems. But named pipes perform communication locally. A socket file can be created using a system call named socket (). Socket files are denoted by 's'.

To list all the named pipe files located in /dev directory, following Linux command can be used:

**ls -l /dev | grep ^s**

## 5.11 FILE ATTRIBUTES

Linux system is a typical multi-user system. Different users in a

different position, with different privileges can access Linux OS. File attributes are used along with the computer files as settings. They are used to either allow or restrict certain operations for a user or for the operating system. To protect the security of the system, Linux system provides file permissions for different users to do different provisions to access the same files (including the catalogue file). The files and directories in Linux can have following attributes:

- **a - append only:** this attribute allows a file to be added to, but not to be removed. It prevents accidental or malicious changes to files that record data, such as log files.
- **c - compressed:** it causes the kernel to compress data written to the file automatically and uncompress it when it is read back.
- **d - no dump:** it makes sure the file is not backed up in backups where the dump utility is used
- **e - extent format:** it indicates that the file is using extents for mapping the blocks on disk.
- **i - immutable:** it makes a file immutable, which goes a step beyond simply disabling write access to the file. The file can't be deleted, links to it can't be created, and the file can't be renamed.
- **j - data journaling:** it ensures that on an Ext3 file system the file is first written to the journal and only after that to the data blocks on the hard disk.
- **s - secure deletion:** it makes sure that recovery of a file is not possible after it has been deleted.
- **t - no tail-merging:** Tail-merging is a process in which small data pieces at a file's end that don't fill a complete block are merged with similar pieces of data from other files.
- **u - undeletable:** When a file is deleted, its contents are saved which allows a utility to be developed that works with that information to salvage deleted files.
- **A - no atime updates:** Linux won't update the access time stamp when you access a file.

- **D - synchronous directory updates:** it makes sure that changes to files are written to disk immediately, and not to cache first.
- **S - synchronous updates:** the changes on a file are written synchronously on the disk.
- **T - and top of directory hierarchy:** A directory will be deemed to be the top of directory hierarchies for the purposes of the Orlov block allocator.

Linux provides us the access control by file and directory permissions on three levels which are user, group and other. These file permissions provide the basic level of security and access control. In Linux, we can use the ls -l command to display user and group attributes of a file and the file belongs to, such as:

```
[root@www /]# ls -l
total 64
dr-xr-xr-x   2 root root 4096 Dec 14  2012 bin
dr-xr-xr-x   4 root root 4096 Apr 19  2012 boot
```

Example, the first attribute in bin and boot file with the "d" represents that it is a directory file.

In Linux, the first character represents the file as a directory, the ordinary file or linked files or device file such as

- the [d] is the directory
- the [-] is the ordinary file;
- If it is [l] then it refers to a linked document (link file);
- If it is [b], it indicates that it is a block device;
- the [c] represents the character device such as a keyboard, a mouse (one reading device).

The next character in groups of three, and are a combination of "rwx" of the three parameters. Wherein, [r] represents read permission, [w] represents write and [x] means execute permission. It should be noted that the position of these three rights will not change. If there is no authority, there will be a minus sign [-] only.



**CHECK YOUR PROGRESS**

**Q 2:** Choose the correct options from the followings:

i) Which of the following is true about files?

A) All data in Unix is organized into files.

B) All files are organized into directories.

C) The directories are organized into a tree-like structure called the filesystem.

D) All of the above

ii) Each entry of directory file has component(s) namely _____.

A) filename B) inode number C) file size

D) Both A and B

iii) Which of the following symbol represents the current directory?

A) Double dot (..) B) Dollar dot($.) C) Single dot (.)

D) Hash dot(#.)

iv) The directory files have:

A) File names & File location B) File names & Inode Numbers

C) File names & Permissions

D) File names & Address v) Device files have following two types?

A) Character & Block B) Character & Socket

C) Block & FIFO D) Only A



## 5.12 LET US SUM UP

- A file is a collection of related data stored in a computer.
- A file system is used to store and organize all the files in a Linux system.
- ext2 file system runs e2fsck program for recover the file after any unclean shutdown.
- ext3 is an extended version of ext2 file system. Journaling concept is introduced in ext3.
- A journal is used to correct the inconsistencies which can occur in case improper or unclean shutdown.

- ext4 file system has some improvised features such as multiblock allocation, delayed allocation, no journaling mode, extents etc.
- ISO 9660 is a file system created for optical media.
- A regular file contains the entire text, binary, image, program and other files.
- Directory contains all the files and folders, sub folders. We can create, delete directories using mkdir and rmdir command respectively.
- cd command is used to change the current working directory of the system.
- A special file called device file is located in /dev directory of the file system. Character device and block device are two types of special files.
- A socket and named pipe file are used for inter process communication.
- Symbolic link file is a reference to a file located in the disk.
- An inode or index node contains all the details about a file. The inode files can be displayed by using ls -i command.
- File attributes are some setting options which allow or restrict specific operations on a file.



## 5.13 FURTHER READINGS

1) Das, S. (2000).UNIX: Concepts and Applications. Tata McGraw-Hill.
2) Bovet, D. P. & Cesati, M. (2005). Understanding the Linux Kernel:from I/O ports to process management. "O'Reilly Media, Inc."



## 5.14 ANSWERS TO CHECK YOUR PROGRESS

**Ans to Q No 1:** i. b ii. c iii. b iv. b v. c

**Ans to Q No 2:** i. D ii. D iii. C iv. B v. D



## 5.15 MODEL QUESTIONS

**Q 1:** What are the purposes of using file system in Linux?

**Q 2:** Name the different series of extended file system.

**Q 3:** Compare the features of ext2, ext3 and ext4 file system.

**Q 4:** What is the limitation of ext2 file system? How it is overcome by ext4 file system?

**Q 5:** What is journaling file system?

**Q 6:** Explain the features of ext4 file system.

**Q 7:** Explain any three commonly used file types in Linux.

**Q 8:** Differentiate between following:

- Hard link vs. Soft link
- Named pipe file vs. Socket files
- Character files and Block files

**Q 9:** Explain the functions of inode in Linux file system.

**Q 10:** Define file attributes. List the different type of attributes used in Linux with their functions.

**Q 11:** Describe different types of file systems used in Linux.

**Q 12:** Write short notes on following file systems:

- xfs
- JFS
- ReiserFS.

**Q 13:** Define boot loader. Explain various boot loaders used in Linux system.

**Q 14:** Discuss about Linux directory structure.

*** ***** ***