UNIX file attributes

Apart from permissions and ownership, a UNIX file has several other attributes, and in this chapter, we look at most of the remaining ones. A file also has properties related to its time stamps and links. It is important to know how these attributes are interpreted when applied to a directory or a device.

This chapter also introduces the concepts of file system. It also looks at the inode, the lookup table that contained almost all file attributes. Though a detailed treatment of the file systems is taken up later, knowledge of its basics is essential to our understanding of the significance of some of the file attributes. Basic file attributes has helped us to know about - ls –l to display file attributes (properties), listing of a specific directory, ownership and group ownership and different file permissions. ls –l provides attributes like – permissions, links, owner, group owner, size, date and the file name.

Listing file and directory attributes and changing permissions can be done using various commands in Unix-like systems:

Listing Attributes:

ls -l: This command lists files and directories with details including permissions, ownership, and timestamps.

```
suman@SUMANS-LAPTOP:~$ ls -l
total 40
-rw-r--r-- 1 nike  suman    17 Jan 14 09:12 apple.sh
-rw-r--r-- 1 suman suman 10240 Jan 18 08:04 archivexl.tar
-rwxrwxrwx 1 suman suman    36 Jan 10 20:12 assign1.sh
---------- 1 suman suman    18 Jan 16 05:53 cclms.txt
-rw-r--r-- 1 suman suman    20 Feb 10 12:22 chini.txt
```

-rw-r--r-- 1 suman suman 10240 Feb 4 07:47 cloud.tar

-rwxrwxrwx 1 suman suman   474 Feb 10 12:20 fileApend.sh

ls -ld: Lists only the directory attributes.

suman@SUMANS-LAPTOP:~$ ls -ld

drwxr-x--- 1 suman suman 4096 Feb 22 09:54

File Ownership: Each file and directory is associated with a user owner and a group owner. The user owner typically has full control over the file, including the ability to change permissions and delete it. The group owner represents a group of users who have certain permissions on the file.

File Permissions: File permissions determine who can read, write, or execute a file. These permissions are typically represented by a combination of letters and symbols:

Read (r): Allows the file to be read.

Write (w): Allows modifications to the file.

Execute (x): Allows the file to be executed if it's a program or script.

Permissions are assigned separately for the owner, group, and others (everyone else).

Changing File Permissions:

Relative Permission:

When you change permissions relative to the current permissions. For example, if a file has permissions rw-r--r--, and you want to add execute permission for the owner, you would use relative permissions to change it to rwxr--r-- (adding execute for the owner while keeping other permissions unchanged).

Changing Permissions:

chmod: This command is used to change file permissions.

chmod g+w,o-rx file.txt

Absolute Permission: When you explicitly set permissions without considering the current permissions. For example, to set absolute permissions to rw-rw-r--, you would use a command like

chmod 664 filename.

Changing Permissions:

chmod: This command is used to change file permissions.

Syntax: chmod [options] mode file

Example: chmod +x filename (adds execute permission to the file for all users).

Changing File Ownership:

File ownership in Unix/Linux refers to the user and group associated with a file. Each file has an owner and group owner, which determine who can access and modify the file.

To change the ownership of a file, the **chown** command is used.

The syntax is **chown [new_owner]:[new_group] file_name**.

For example, to change the owner of a file named "example.txt" to "user1" and the group owner to "group1", you would use the command

chown user1:group1 example.txt.

Changing Group Ownership:

Group ownership allows multiple users to access files with common permissions. A file's group owner determines the group permissions for that file.

To change the group ownership of a file, the chgrp command is used.

The syntax is **chgrp new_group file_name**.

For example, to change the group ownership of a file named "example.txt" to "group2", you would use the command chgrp group2 example.txt.

File System and Inodes:

A file system is a method of organizing and storing files and directories on a storage device. It defines how data is stored, accessed, and managed.

In Unix/Linux file systems, each file and directory is represented by an inode. An inode contains metadata about the file or directory, including permissions, timestamps, and pointers to data blocks.

Inodes provide a way to efficiently manage and access files on disk. They allow the file system to track and manage file attributes and data blocks.

ls -i

Soft Link (Symbolic Link):

A soft link, also known as a symbolic link or symlink, is a special type of file that points to another file or directory by its path name.

Unlike hard links, symbolic links do not point directly to the inode of the target file. Instead, they contain the path to the target file.

Symbolic links can span different file systems and can point to directories. They are commonly used for creating shortcuts.

ln -s /path/to/target_file link_name

Hard Link:

A hard link is a directory entry that points directly to the inode of a file. It creates an additional name (link) for the same file, allowing it to be accessed from multiple locations.

Hard links are essentially multiple file names that refer to the same data blocks on disk. They have the same inode number and share the same file contents.

ln /path/to/target_file link link_name

Significance of File Attribute for Directory:

In UNIX-like operating systems, directories are also treated as files. However, they have a special attribute that distinguishes them from regular files. This attribute is commonly referred to as the "d" flag in the file permissions field when you use the ls -l command to list directory contents. This flag indicates that the file is a directory.

Directories serve as containers for other files and directories. They provide a hierarchical structure for organizing and accessing files on a filesystem. They also support operations such as creating, deleting, renaming, and listing files within them.

Default Permissions of File and Directory:

By default, when a file or directory is created in a UNIX-like operating system, it is assigned certain permissions. These permissions determine who can read, write, or execute the file or directory. The default permissions are typically set by the umask value.

For files, the default permissions are commonly set to 666, which means read and write permissions for the owner, group, and others (rwxrw-rw-).

For directories, the default permissions are commonly set to 777, which means read, write, and execute permissions for the owner, group, and others (rwxrwxrwx).

Using umask:

The umask (short for "**user file-creation mode mask**") is a UNIX command that determines the default file permissions for newly created files and directories. It

works by subtracting the specified umask value from the maximum permissions (usually 777 for directories and 666 for files).

The umask value is typically represented in octal notation. It consists of three digits, each representing the permissions that should be removed from the default maximum permissions. For example, a umask value of 022 (commonly used) means that write permissions should be removed for the group and others.

To set the umask value, you can use the umask command followed by the desired umask value. For example:

umask 022

This sets the umask value to 022, which means that newly created files will have permissions of 644 (666 - 022) .

The umask value can be set in shell startup files like .bashrc or .bash_profile to ensure that it is applied every time a new shell session is started.

Listing of Modification and Access Time:

In Unix-like operating systems, every file and directory has three timestamps associated with it:

1. Access time (atime)
2. Modification time (mtime),
3. status change time (ctime).

These timestamps record when the file was last accessed, modified, and had its metadata changed, respectively.

Access Time (atime): The access time of a file is updated whenever the file is read or accessed. This includes opening the file for reading, executing, or listing its

contents. The access time can be checked using the ls -l command, and it can be seen in the output as part of the file metadata.

Modification Time (mtime): The modification time of a file is updated whenever the contents of the file are modified. This includes changes made to the file's data, such as appending or deleting content. The modification time can also be checked using the ls -l command, and it is displayed alongside the access time in the file metadata.

Status Change Time (ctime): The status change time of a file is updated whenever the file's metadata is modified. This includes changes to permissions, ownership, or the file's name. The ctime can also be viewed using the ls -l command, and it appears alongside the access and modification times in the file metadata.

Time Stamp Changing (touch):

The touch command in Unix-like operating systems is used to modify the access and/or modification times of files.

By default, touch updates both the access time and modification time to the current system time. However, it can also be used to set specific timestamps using the **-t** option followed by a specific timestamp.

touch filename updates both the access and modification times to the current system time.

touch -t YYYYMMDDHHMM.SS filename sets both the access and modification times to a specific timestamp specified in the format YYYYMMDDHHMM.SS.

File Locating (find):

The find command is used to search for files and directories within a specified directory hierarchy based on various criteria such as filename, modification time,

size, ownership, and permissions. It is a powerful tool for locating files that match specific patterns or criteria.

find /path/to/search -name "pattern"

find filename