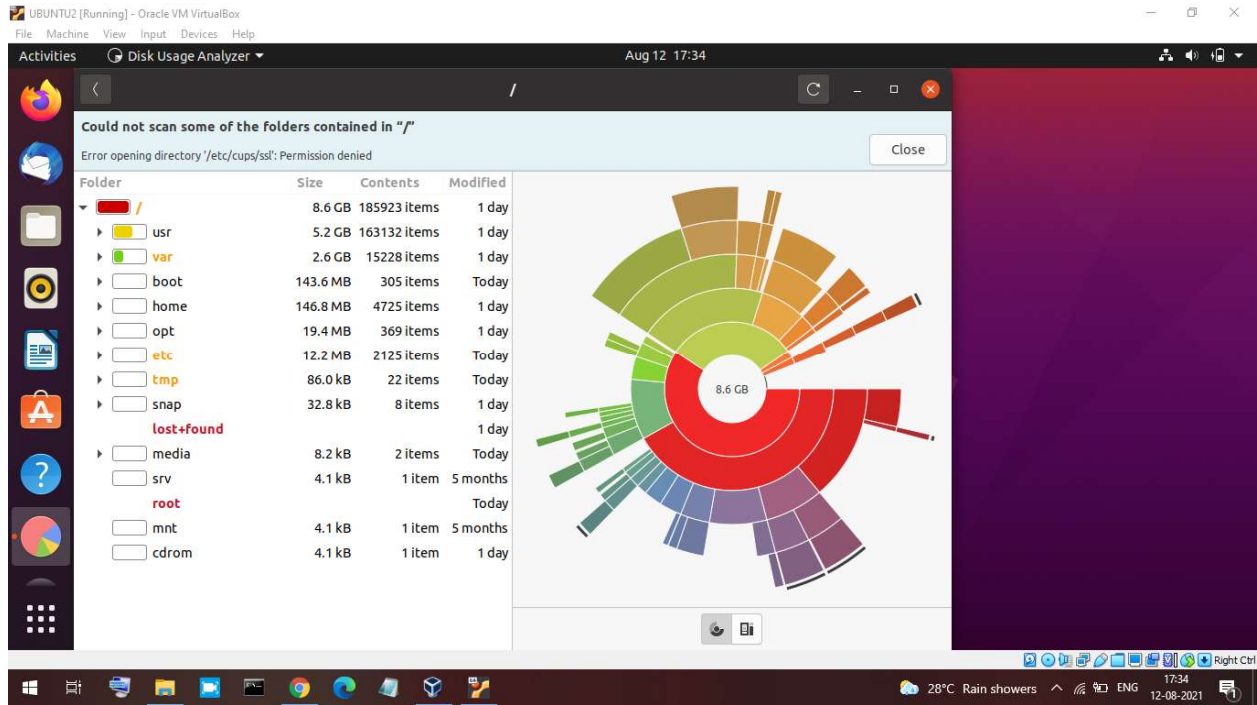
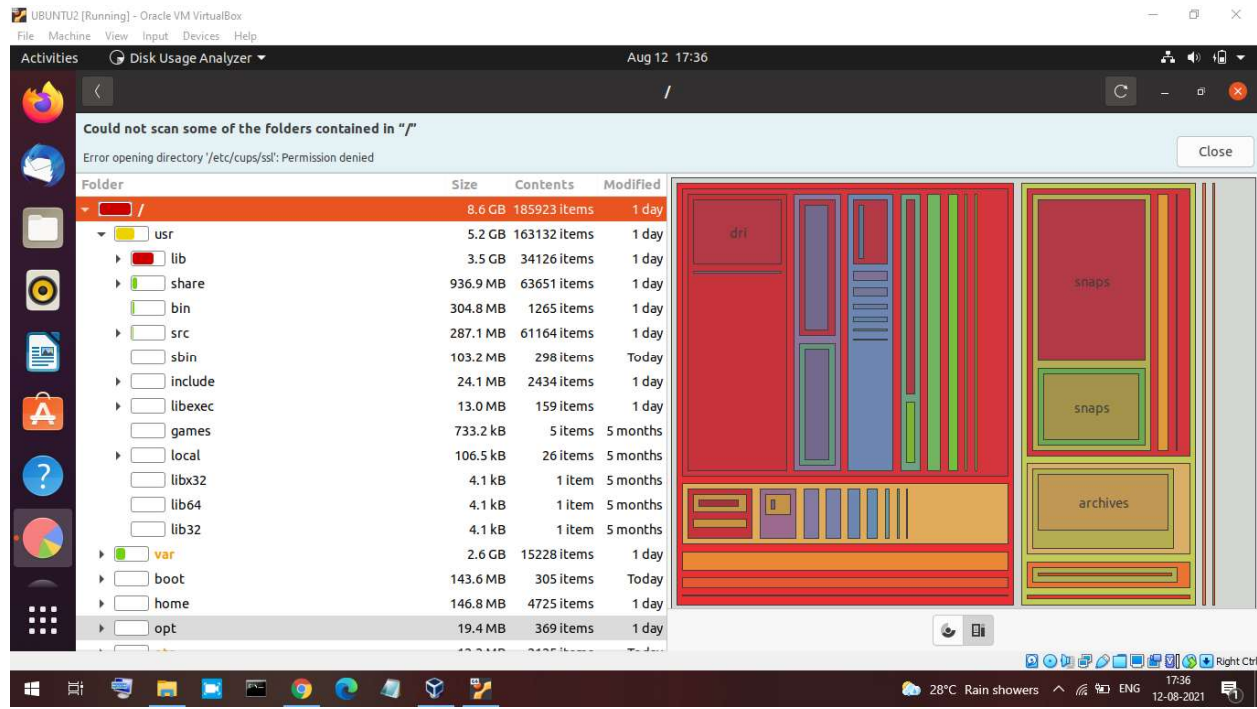


EXPERIMENT NO :-3

AIM:- FILE SYSTEM HEIRARCHY & FILE PERMISSIONS IN LINUX

LINUX DIRECTORY STRUCTURE (File System Structure)





- **/bin** : All the executable binary programs (file) required during booting, repairing, files required to run into single-user-mode, and other important, basic commands viz., cat, du, df, tar, rpm, wc, history, etc.
- **/boot** : Holds important files during boot-up process, including Linux Kernel.
- **/dev** : Contains device files for all the hardware devices on the machine e.g., cdrom, cpu, etc
- **/etc** : Contains Application's configuration files, startup, shutdown, start, stop script for every individual program.
- **/home** : Home directory of the users. Every time a new user is created, a directory in the name of user is created within home directory which contains other directories like Desktop, Downloads, Documents, etc.
- **/lib** : The Lib directory contains kernel modules and shared library images required to boot the system and run commands in root file system.
- **/lost+found** : This Directory is installed during installation of Linux, useful for recovering files which may be broken due to unexpected shut-down.

- **/media** : Temporary mount directory is created for removable devices viz., media/cdrom.
- **/mnt** : Temporary mount directory for mounting file system.
- **/opt** : Optional is abbreviated as opt. Contains third party application software. Viz., Java, etc.
- **/proc** : A virtual and pseudo file-system which contains information about running process with a particular Process-id aka pid.
- **/root** : This is the home directory of root user and should never be confused with '/'
- **/run** : This directory is the only clean solution for early-runtime-dir problem.
- **/sbin** : Contains binary executable programs, required by System Administrator, for Maintenance. Viz., iptables, fdisk, ifconfig, swapon, reboot, etc.
- **/srv** : Service is abbreviated as 'srv'. This directory contains server specific and service related files.
- **/sys** : Modern Linux distributions include a /sys directory as a virtual filesystem, which stores and allows modification of the devices connected to the system.
- **/tmp** : System's Temporary Directory, Accessible by users and root. Stores temporary files for user and system, till next boot.
- **/usr** : Contains executable binaries, documentation, source code, libraries for second level program.
- **/var** : Stands for variable. The contents of this file is expected to grow. This directory contains log, lock, spool, mail and temp files.

FILE PERMISSIONS IN LINUX

Linux file permissions, attributes, and ownership control the access level that the system processes and users have to files. This ensures that only authorized users and processes can access specific files and directories.

The basic Linux permissions model works by associating each system file with an owner and a group and assigning permission access rights for three different classes of users:

- The file owner.
- The group members.
- Others (everybody else)

User/Owner

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence, a user is also sometimes called an owner.

Group

A user- group can contain multiple users. All users belonging to a group will have the same Linux group permissions access to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for the world.

Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above.

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.

- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.

Changing file/directory permissions with 'chmod' command

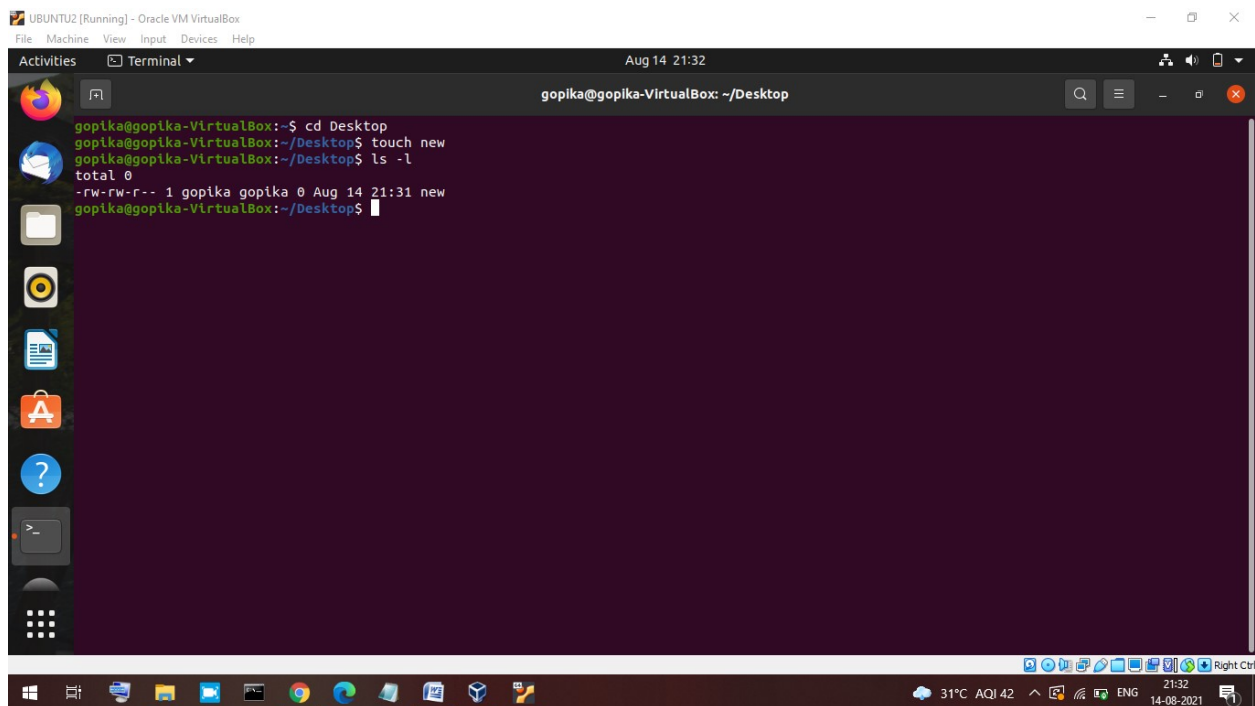
Symbolic Mode

In the Absolute mode, you change permissions for all 3 owners. In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the Unix file permissions.

OPERATOR	DESCRIPTION
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permission set earlier

The various owners are represented as –

User Denotations	
u	User /owner
g	group
o	others
a	all



The screenshot shows a terminal window titled "gopika@gopika-VirtualBox: ~/Desktop". The user has executed the following commands:

```
gopika@gopika-VirtualBox:~$ cd Desktop
gopika@gopika-VirtualBox:~/Desktop$ touch new
gopika@gopika-VirtualBox:~/Desktop$ ls -l
```

The output of the `ls -l` command is:

```
total 0
-rw-rw-r-- 1 gopika gopika 0 Aug 14 21:31 new
```

The permissions `-rw-rw-r--` are displayed for the file `new`, indicating that the owner (gopika) has read and write permissions, while the group (gopika) and others have read permissions.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –

- The first three characters (2-4) represent the permissions for the file's owner. For example, **-rw-rw-r--** represents that the owner has read (r) and write (w) permission, but no execute permission.

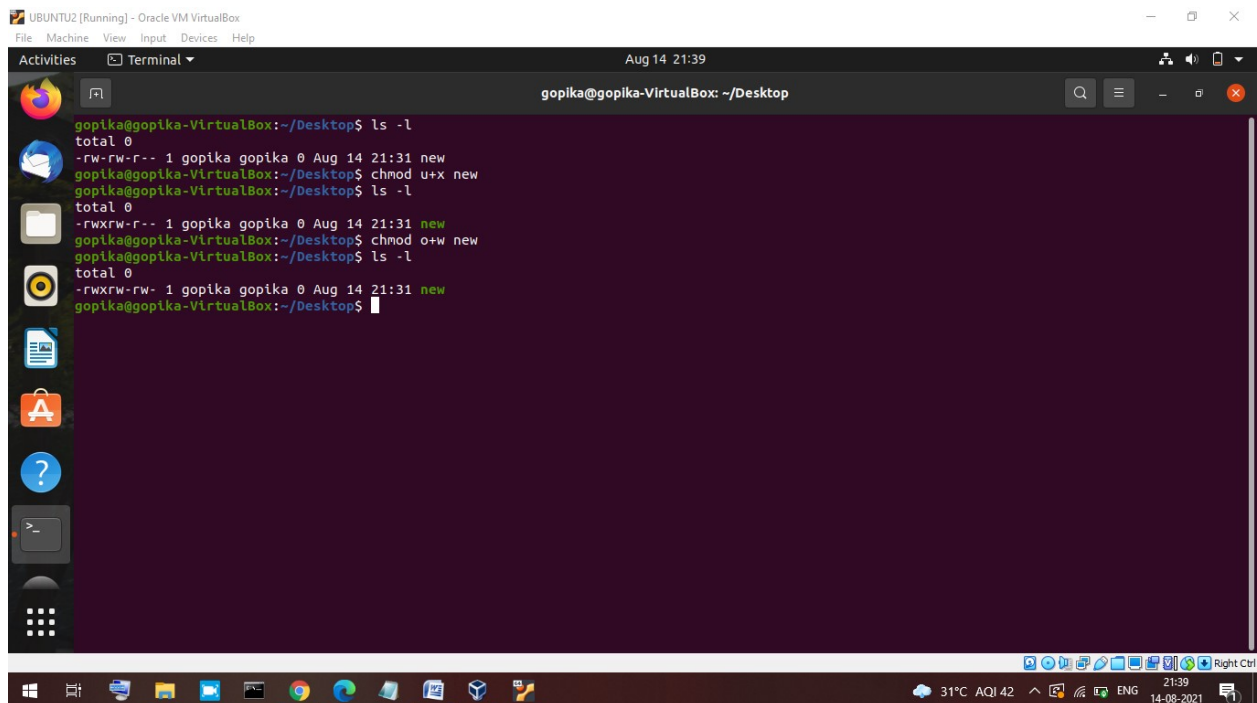
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rw-rw-r--** represents that the group has read (r) and write(w) permission but no execute (x) permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rw-rw-r--** represents that there is **read (r)** only permission.

To change directory permissions in Linux, use the following:

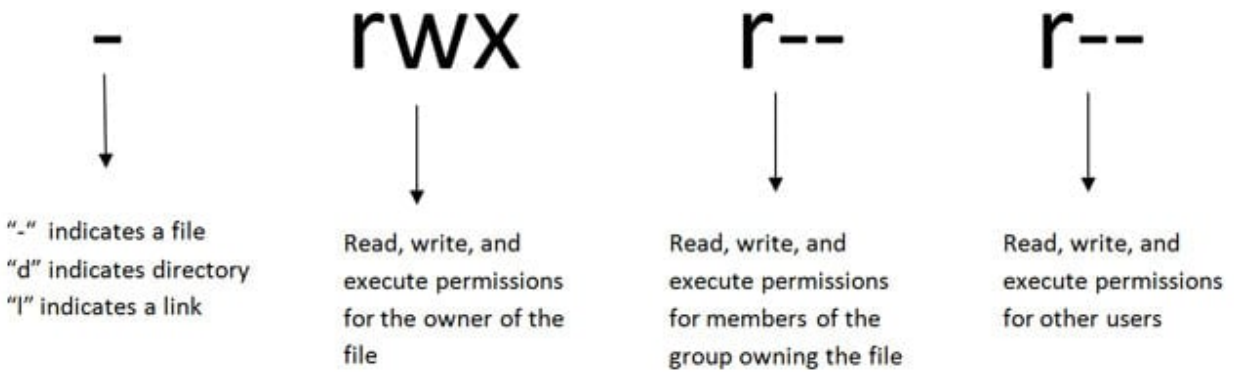
- **chmod + rwx filename** to add permissions.
- **chmod - rwx directoryname** to remove permissions.
- **chmod + x filename** to allow executable permissions.
- **chmod - wx filename** to take out write and executable permissions.

Here “r” is for read, “w” is for write, and “x” is for execute.

This only changes the permissions for the owner of the file.



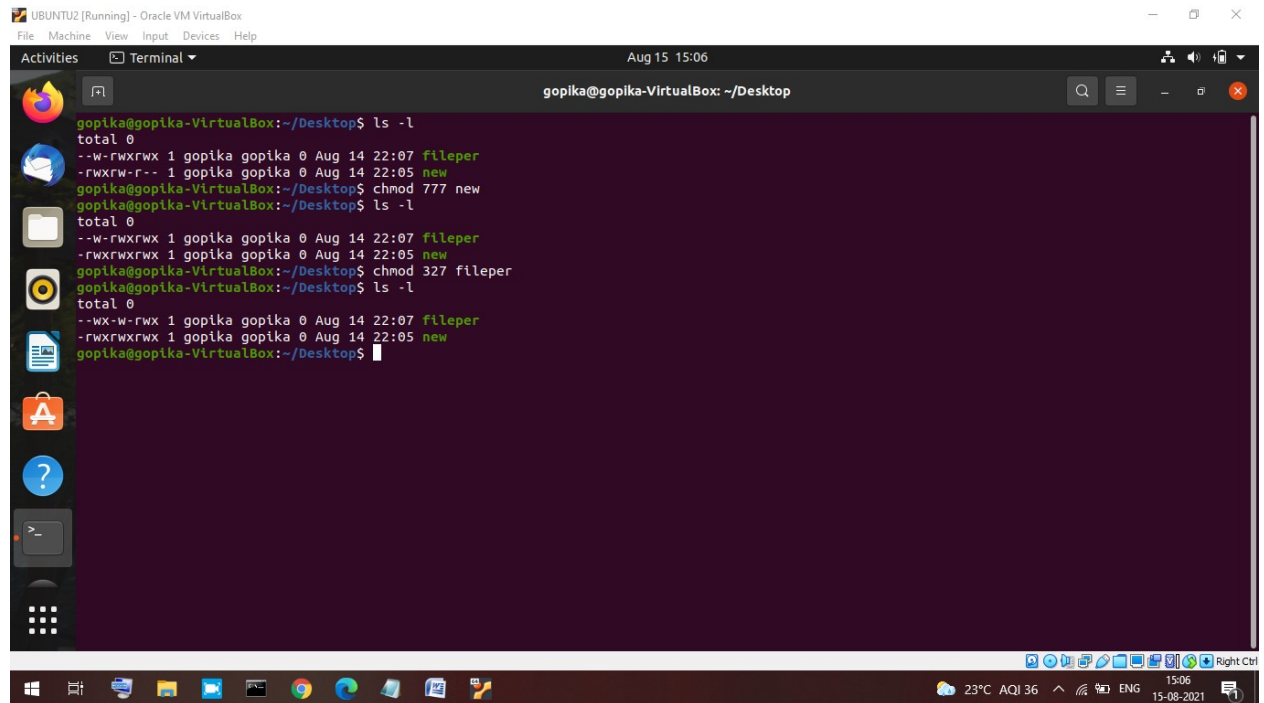
```
gopika@gopika-VirtualBox: ~/Desktop
gopika@gopika-VirtualBox:~/Desktop$ ls -l
total 0
-rw-rw-r-- 1 gopika gopika 0 Aug 14 21:31 new
gopika@gopika-VirtualBox:~/Desktop$ chmod u+x new
gopika@gopika-VirtualBox:~/Desktop$ ls -l
total 0
-rwxrw-r-- 1 gopika gopika 0 Aug 14 21:31 new
gopika@gopika-VirtualBox:~/Desktop$ chmod o+w new
gopika@gopika-VirtualBox:~/Desktop$ ls -l
total 0
-rwxrw-rw- 1 gopika gopika 0 Aug 14 21:31 new
gopika@gopika-VirtualBox:~/Desktop$
```



Numeric Mode

Permission numbers are:

- **0** = --- no permission
- **1** = --x execute permission only
- **2** = -w- permission to write only
- **3** = -wx permission for write and execute
- **4** = r- permission for read
- **5** = r-x permission for read and execute
- **6** = rw- permission for read and write
- **7** = rwx permission for read ,write and execute



```
gopika@gopika-VirtualBox: ~/Desktop$ ls -l
total 0
--w-rwxrwx 1 gopika gopika 0 Aug 14 22:07 fileper
-rwxrwxr-- 1 gopika gopika 0 Aug 14 22:05 new
gopika@gopika-VirtualBox:~/Desktop$ chmod 777 new
gopika@gopika-VirtualBox:~/Desktop$ ls -l
total 0
--w-rwxrwx 1 gopika gopika 0 Aug 14 22:07 fileper
-rwxrwxrwx 1 gopika gopika 0 Aug 14 22:05 new
gopika@gopika-VirtualBox:~/Desktop$ chmod 327 fileper
gopika@gopika-VirtualBox:~/Desktop$ ls -l
total 0
--wx-w-rwx 1 gopika gopika 0 Aug 14 22:07 fileper
-rwxrwxrwx 1 gopika gopika 0 Aug 14 22:05 new
gopika@gopika-VirtualBox:~/Desktop$
```

CHANGING FILE OWNERSHIPS

The chown command allows us to change the user and/or group ownership of a given file, directory, or symbolic link.

In Linux, all files are associated with an owner and a group and assigned with permission access rights for the file owner, the group members, and others.

These commands will give ownership to someone, but all sub files and directories still belong to the original owner.

Chown <name> <file name>

Or

chown [OPTIONS] USER[:GROUP] FILE(s)

USER is the user name or the user ID (UID) of the new owner. GROUP is the name of the new group or the group ID (GID). FILE(s) is the name of one or more files, directories or links. Numeric IDs should be prefixed with the + symbol.

- USER - If only the user is specified, the specified user will become the owner of the given files, the group ownership is not changed.
- USER: - When the username is followed by a colon :, and the group name is not given, the user will become the owner of the files, and the files group ownership is changed to user's login group.
- USER:GROUP - If both the user and the group are specified (with no space between them), the user ownership of the files is changed to the given user and the group ownership is changed to the given group.
- :GROUP - If the User is omitted and the group is prefixed with a colon :, only the group ownership of the files is changed to the given group.
- : If only a colon : is given, without specifying the user and the group, no change is made.

By default, on success, chown doesn't produce any output and returns zero.

Superuser permissions are necessary to execute the chown command.