CHAPTER 7: BASIC FILE ATTRIBUTES

➢ LS –L COMMAND:

- Generally, it is used for listing file attributes.
- File attributes like its permission, size and ownership details.
- Example,
  To display seven attributes of all files in current directory.

```
$ ls –l

total 28
-rwxr-xr-r--    1 nirzari  nirzari     13    Jul  7 02:24    cat_v
-rw-rw-r--      1 nirzari  nirzari     47    Jul  7 03:15    cat_voption
-rw-rw-r--      1 nirzari  nirzari     12    Jul  7 03:21    file11
-rw-rw-r--      1 nirzari  nirzari     19    Jul  7 03:14    ty_cat
drwxrwxr-x   3 nirzari  nirzari   4096    Jul  7 03:49    tya
drwxrwxr-x   3 nirzari  nirzari   4096    Jul  7 03:38    tyb
drwxrwxr-x   2 nirzari  nirzari   4096    Jul  7 02:10    tybca_B
```

- The list is preceded by words total 28 which indicates that a total of 28 blocks are occupied by these files in disk, each block contain 512 bytes
- *File type and permission:*
  - 1st column shows *file types and file permissions* associated with each file.
  - 1st character is mostly – which indicates that the file is ordinary.
  - Sometimes 1st character is d which indicated that it is directory.
  - There are 3 types of permissions: read ®, write (w) and execute (x) permission.
- *Links:*
  - 2nd column indicates *number of links associated with file*.
  - This is actually the number of file names maintained by the system of that file.
  - A link count greater than one indicates that the file has more than name. that doesn't mean that there are 2 copies of the files.
- *Ownership*:

- When we create a file then we become owner automatically.
- The 3rd column shows 'nirzari' as the owner of the files.
- The owner has full authority to tamper with file contents and permissions.

- *Group Ownership:*
  - When opening a user account, the system administrator also assigns the user to some group.
  - 4th column represents the *group owner of the file*.
  - The members who represents same group can able to work on the same files.
- *File size:*
  - 5th column shows file size in the bytes i.e. the amount of data file contains.
- *Last modification time:*
  - 6th, 7th and 8th columns indicate the last modification time of the file.
- Filename:
  - Last column displays the filenames arranged in the ASCII sequence.

➢ LS –D COMMAND:

- This command list the directory attributes.
- Example,

```
        $ ls –ld tya tyb
drwxrwxr-x  3 nirzari  nirzari     4096 Jul  7 03:49 tya
drwxrwxr-x  3 nirzari  nirzari     4096 Jul  7 03:38 tyb
```

- Directories are easily identified by the 1st character of 1st column i.e. ‘d’.

➢ FILE OWNERSHIP:

- Several users may belong to same group.
- People working on a project are generally assigned a common group and all files created by group members having separate user id will have the same group owner name.
- The privileges of the group are set by the file and not by the group members.
- When the system administrator creates a user account, admin has to assign these parameters to the user.
  - The User-Id (UID):
    Both its name and numeric representation.
  - The Group-Id (GID):
    Both its name and numeric representation. The administrator has to assign the group name also if the GID represents a new group.
- The file /etc/password maintains UID (Both name and numeric) and GID (Only numeric).
- /etc/group contains the GID (both number and name)
- Example,

```
        $ id
uid=643(nirzari) gid=574(nirzari) groups=574(nirzari)
```

➢ FILE PERMISSIONS:

- When we execute ls –l command then 1st column indicates file permissions.

FILE ACCESS MODES:

- The permissions of a file are the first line of defense in the security of a UNIX system.
- The basic building blocks of Unix permissions are the read, write, and execute permissions, which are described below –
  1. Read
     Grants the capability to read i.e. view the contents of the file.
  2. Write
     Grants the capability to modify, or remove the content of the file.
  3. Execute
     User with execute permissions can run a file as a program.

DIRECTORY ACCESS MODES:

- Directory access modes are listed and organized in the same manner as any other file.
- There are a few differences that need to be mentioned:
  1. Read
     Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
  2. Write
     Access means that the user can add or delete files to the contents of the directory.
  3. Execute
     Executing a directory doesn't really make a lot of sense so think of this as traverse permission.

➢ CHMOD COMMAND:

- This command is used for changing the file permission.
- A file or directory is created with a default set of permissions. This default is determined by umask command.
- On Linux and other Unix-like operating systems, there is a set of rules for each file which defines who can access that file, and how they can access it. These rules are called *file permissions or file modes*.
- The command name chmod stands for "change mode", and it is used to define the way a file can be accessed.

- The chmod commands used to set the permissions of one or more files for all 3 categories of users (user, group and other).
- The command can be used in 2 ways:
  1. In a **relative manner** by specifying the changes to the current permissions.
  2. In an **absolute manner** by specifying the final permissions.

- **RELATIVE MANNER PERMISSIONS:**

  - When changing permissions in a relative manner, chmod command only changes the permissions specified in the command line and leaves the other permissions unchanged.
  - General form of chmod command:
    $ chmod options permissions filename

  - This command takes its argument an expression comprising some letters and symbols that completely describes the user category and type of permission being assigned or removed.
  - The expression contains 3 components:
    1. User Category (user, group, others)
    2. The operation to be performed (assigned or removed permissions)
    3. The type of permissions (Read, Write, Execute)
  - We have to use abbreviations in the arguments of chmod command

Category	Operation	Permission
u → User g → Group o → others a → all	+ → Assign permission - → Removes permission = → Assign Absolute permission	r → Read Permission w → Write Permission x → Execute Permission

  - To assign execute permission to the User of the file fl_Chmod, we need the expression from above table i.e. **u+x**

Example,

The command assigns (+) execute (x) permission to users (u) but permissions remain unchanged.

```
$ chmod u+x fl_Chmod
$ ls –l
-rwxrw-r--  1  nirzari  nirzari  36  Jul  9  03:17  fl_Chmod
```

- If we are owner of the file then owner can execute the file but for other categories (group and other) we can't.

  Example,

  The command assigns (+) execute (x) permission to all categories of users (ugo)

```
$ chmod ugo+x fl_Chmod
$ ls –l
-rwxrwxr-x  1  nirzari  nirzari  36  Jul  9  03:17
fl_Chmod
```

- If we don't specify the permissions then permissions applied to all categories

```
$ chmod a+x fl_Chmod2
$ chmod +x fl_Chmod2
```

- Chmod command accepts more than 1 filename as arguments in the command line. When you want to give same permission to list of more than one file then

  Example,

```
$ chmod u+x file1 file2 file3
```

- Permissions are removed with – operator.

  Example,

```
$ ls –l
    -rwxrwxr-x  1 nirzari  nirzari  36 Jul  9 03:17 file1
$ chmod go-r file1
$ ls –l
    -rwx-wx--x  1 nirzari  nirzari  36 Jul  9 03:17 file1
```

- Chmod command also accepts multiple expressions delimated by comma.

  Example,

To restore the original premissions to the file1then we have to remove the execute permission from all (a-x) and assign read permission to group and others (go+r)

```
$ ls –l
        -rwx-wx--x   1 nirzari  nirzari       36 Jul  9 03:17 file1
$ chmod a-x,go+r file1
$ ls –l
        -rw-rw-r--   1 nirzari  nirzari       36 Jul  9 03:17 file1
```

- **ABSOLUTE PERMISSIONS:**

  - The chmod command uses a three-digit code as an argument.
  - Here the expression used by chmod command is a string of 3 octal numbers (i.e. base 8).
  - The three digits of the chmod code set permissions for these groups in this order:
    1. Owner (you)
    2. Group (a group of other users that you set up)
    3. World (anyone else browsing around on the file system)

  - Each digit of this code sets permissions for one of these groups as follows.
    1. Read is 4
    2. Write is 2
    3. Execute is 1

  - The sums of these numbers give combinations of these permissions:-
    1. (00) 0 = no permissions whatsoever; this person cannot read, write, or execute the file
    2. (001) 1 = execute only
    3. (010) 2 = write only 3 = write and execute (1+2)
    4. (100) 4 = read only
    5. (101) 5 = read and execute (4+1)
    6. (110) 6 = read and write (4+2)
    7. (111) 7 = read and write and execute (4+2+1)

- Example,
  Following command indicates read and write permissions (4+2)

```
$ ls –l
        -rw-rw-r--    1 nirzari  nirzari       36 Jul  9 03:17 file1
$ chmod 666 file1
$ ls –l
        -rw-rw-rw-    1 nirzari  nirzari       36 Jul  9 03:17 file1
```

  Now, to remove the write permission (2) from group and others

```
$ ls –l
        -rw-rw-rw-    1 nirzari  nirzari       36 Jul  9 03:17 file1
$ chmod 644 file1
$ ls – l
        -rw-r--r--    1 nirzari  nirzari       36 Jul  9 03:17 file1
```

  To assign all permissions to the owner, read and write permissions to group and only execute permission to others then

```
$ ls –l
        -rw-r--r--    1 nirzari  nirzari       36 Jul  9 03:17 file1
$ chmod 761 file1
$ ls –l
         -rwxrw---x    1 nirzari  nirzari       36 Jul  9 03:17 file1
```

- **CHMOD COMMAND RECURSIVELY (-R):**
  - This command decend the directory hierarchy and apply the expression to every file and subdirectories finds.
  - Example,

```
$ chmod –R a+x tya
$ ls –l
        -rwxrwxr-x    1 nirzari  nirzari       12 Jul  7 03:26 file2
        -rwxrwxr-x    1 nirzari  nirzari       12 Jul  7 02:08 file3
        drwxrwxr-x    2 nirzari  nirzari     4096 Jul  7 03:32 ty_a
```

  - This makes all files and subdirectories found in tree-walk executable by all users.

➢ CHANGING FILE OWNERSHIP:

- Using 2 commands we can change file and directory ownership:
  1. chown command
  2. chgrp command

○ CHOWN COMMAND:

- Basically, this command is used to change the file ownership.
- This command transfers ownership of a file to user. And it seems that it can optionally change the group also.
- Syntax,
  $ chown options owner [:group] file(s)

- This command requires the user-id (UID) of the recipient followed by one or more file names.
- Changing ownership requires super user permission.
- Example,
  Using su command, can change status of super user.
  $ su
  Password: ******** *This is root password*
  # _ *This is another shell*

- This command requires the user-id (UID) of the recipient followed by one or more file names.
- Example,
  Following command change the ownership from Nirzari to Heena

  # ls –l file1:
  -rw-rw-r--1 nirzari nirzari 14 2015-07-15 08:25 file1
  # chown heena file1
  # ls –l file1
  -rw-rw-r--1 heena nirzari 14 2015-07-15 08:30 file1
  # exit
  $ _

- **<u>CHGRP COMMAND</u>:**

  - This command is used to change the file's group owner.
  - The syntax is same as chown command.
  - Syntax,
    $ chgrp options owner [:group] file(s)

  - Example,
    $ ls –l file1
    -rw-rw-r--1 nirzari nirzari 14 2015-07-15 08:35 file1
    $ chgrp DBA file1
    $ ls –l file1
    -rw-rw-r--1 nirzari DBA 14 2015-07-15 08:38 file1