**File Ownership & Permission**

Linux is multiuser OS so that more than one user can access resources simultaneously. Any logged on user can modify or execute data/application of other user. It compromises security and privacy of data. To avoid such scenarios and provide security, Linux uses file ownership & permission concept.

**Ownership:**

* **User:** A User is the owner of the file. By default, a user is the person who created a file.
* **Group:** Collection of multiple users. All users belonging to a group have the same access

Permissions to the file. [If group of people requires access to the same file with

same kind of permissions then instead of giving permissions individually, make a

a group of them and assign group permission.]

* **Other:** Any user that is neither part of *User* nor *Group* class.

**Permissions:**

|  |  |  |
| --- | --- | --- |
| **Permission** | **On a normal file** | **On a directory** |
| **Read(R)** | Only allows to read/view content of the file. | Only allows to view the names of files (list of files) in a directory. |
| **Write(W)** | Allows to modify and delete the file. | Allows to delete the directory, modify contents of files. Also allows to create, remove or rename the file. |
| **Execute(X)** | Allows to execute the file (Read permission is must.) | Allows to access/traverse (through cd ) and access metadata of files. |

To display file/directory permissions,

Command: ls –l <filename>



File size (KB)

Mode

File/Directory name /path

Hard link count

Last Modified time and date

Group’s Owner

User’s Owner

In *Mode*  part,

* 1st letter indicates type of file.

|  |  |
| --- | --- |
| **1st letter** | **type of the file** |
| -(hyphen) | Normal file |
| d | Directory |
| b | Block file |
| c | Character file |
| l | Symbolic link |
| p | Pipe file |
| s | Socket file |

* Next three letters represent *User* Owner’s permission.
* Next three letters represent *Group* Owner’s permission.
* Next three letters represent *Other*’s permission.

Example:

C:\Users\vbhvjari\Desktop\Untitled.png

* Here, 1st letter “d” indicates it is directory.
* Next three letters: rwx represent User Owner’s permission. It means User has read, write and execute permission.
* Next three letters: r-x represent Group Owner’s permission. It means Group Owner has read and execute permission.
* Next three letters: r-x represent Other’s permission. It means other user has read and execute permission.

**Modifying File/Directory Permissions:**

There are 2 methods to modify the permission.

1. **Symbolic method:**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Add a permission to file/directory |
| - | Remove the permission |
| = | Set the permission and overrides the permissions set earlier. |

|  |  |
| --- | --- |
| **Acronym** | **Full Form** |
| u | User/Owner |
| g | Group owner |
| o | Other |
| a | all |

Example:

Here, **chmod** command is used to modify the permission. Say v1 is file under vbhv directory.

* chmod u+r v1: add read permission to user.
* chmod u=rwx v1: set read, write and execute permission to User regardless of previously set permissions.
* chmod g-x v1 : Remove execute permission from group owner.
* chmod a+w v : set write permission to User, Group owner and other user.
* chmod ug-r v : Remove read permission from User and Group owner.

1. **Numeric(Absolute) Method:**

Permissions are represented by numbers. 1st number indicates permission of User/Owner. 2nd number indicates permission of Group Owner. 3rd number indicates permission of Other.

|  |  |
| --- | --- |
| **Number** | **Permissions** |
| 7 | rwx |
| 6 | rw- |
| 5 | r-w |
| 4 | r-- |
| 3 | -wx |
| 2 | -w- |
| 1 | --x |
| 0 | --- |

|  |  |
| --- | --- |
| r | 4 |
| w | 2 |
| x | 1 |

Example:

* chmod 777 v1 : set all permissions to all users.
* chmod 421 v1: set read permission to User/Owner, write permission to group owner, execute permission to Other.

**Setting Default Permission:**

umask(User mask) command is used to set default permission based on maximum allowable default permission and umask value.

1. **Numeric Method:**

|  |  |  |
| --- | --- | --- |
| Maximum allowable default permission | Symbolic | Numeric |
| File | rw-rw-rw- | 666 |
| Directory | rwxrwxrwx | 777 |

Maximum allowable default permission

umask value

Default permission

* This newly set default permission applicable to files/directories once umask value is set.
* Default umask value for file/directory is 0022.

|  |  |  |
| --- | --- | --- |
|  | file | directory |
| Maximum allowable default permission | 666 | 777 |
| default umask value | 022 | 022 |
| default permission | 644 | 755 |

* To see umask value,

Command: umask

* To set umask value,
* Command: umask <umask\_value>

e.g., umask 244

1. **Symbolic Method:**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Allows the specified permissions to be enabled for the specified user classes (permission that are not specified are unchanged in the mask) |
| - | Prohibit the specified permissions from being enabled for the specified user classes(permission that are not specified are unchanged in the mask) |
| = | Allows the specified permissions to be enabled for the specified user classes(permission not specified will be prohibited by the mask during file creation) |

Example : Say default umask is 022.

* umask a= >> permissions: ---------
* umask u-r >> permissions: -w-r—r—

**Special Permissions:**

Three special permissions which can be set for binary/executable files and public directories to meet requirements.

1. SUID(Set User ID)
2. SGID(Set Group ID)
3. Sticky bit
4. **SUID(Set User ID):**

* This permission is given to a file.
* In Linux, when program runs, it inherits access permissions from logged on user.
* In some case, program needs privilege access that normal user does not own.
* In such cases, SUID is defined by giving temporary permissions to a normal user who runs a program as owner rather giving permanent access to normal user.
* SUID is applicable only to user/owner.

To set SUID,

Symbolic Method >> Command: chmod u+s <filename>

Numeric Method >> Command: chmod 4665 <filename>

[Say 665 is previously set permission. **Append 4**]



To convert from S to s,

chmod u+x <filename>



To remove SUID, Use same command except “–“ in place of “+”.

1. **SGID(Set Group ID):**

* This permission is given to a group.
* It gives temporary access to a user to run the program with the permissions of the file group permissions to become member of that group to execute the file**.**
* **SGID is useful when it is required to give temporary access to more than one users from different groups.**

To set GUID,

Symbolic Method >> Command: chmod g+s <filename>

Numeric Method >> Command: chmod 2665 <filename>

[Say 665 is previously set permission. **Append 2**]



To convert from S to s,

chmod g+x <filename>



To remove GUID, Use same command except “–“ in place of “+”.

1. **Sticky Bit:**

* It is used on folders in order to avoid deletion of a folder and its content by other users though they having write permissions on the folder contents.
* If Sticky bit is enabled on a folder, the folder contents are deleted by only owner who created them and the root user.

To set Sticky bit,

Symbolic Method >> Command: chmod o+t <filename>

Numeric Method >> Command: chmod 1665 <filename>

[Say 665 is previously set permission. **Append 1**]



To convert from T to t,

chmod o+x <filename>



To remove Sticky bit, Use same command except “–“ in place of “+”.

|  |  |
| --- | --- |
| **Permissions** | **Meaning** |
| --S------ | SUID is set, but user (owner) execute is not set. |
| --s------ | SUID and user execute are both set. |
| -----S--- | SGID is set, but group execute is not set. |
| -----s--- | SGID and group execute are both set. |
| --------T | Sticky bit is set, but other execute is not set. |
| --------t | Sticky bit and other execute are both set. |

**Note:** Use of S/G/T:

To restrict particular user/group/others to have execution privilege.

**Modifying Ownership of file/directory:**

* To change ownership of user,

Command: chown <new\_username> <filename>

* To change ownership of user as well as of group,

Command: chown <new\_username> <new\_groupname> <filename>

* To change only ownership of group,

Command: chgrp <new\_groupname> <filename>

**Notes:**

* File /etc/group contains all the groups in the file system.
* No file belongs to more than one group.
* Nesting of group is not possible in Linux.