# DevOps Learning Notes



## Linux Permissions

In this lecture you are learning Linux Permissions:

1. File Permissions
2. Directory Permissions
3. Sticky Bit
4. Linux Users
5. Linux Groups

**Base Permissions:**

In Linux every file and directory have below three permissions.

**Owner Permissions:** Permissions allowed to owner of file or directory what the owner can do on the file/directory.

**Group Permissions:** Group Permissions allowed to Group members of file or directory. If File or Directory is in Oracle group then the Group permissions are applied to the members' part of Oracle group.

**Other (world) Permissions:** Others Permissions are applied for the users who are not owner of File or Directory and not part of the same Group.

**How to see File/Directory attributes:**

$ls -l /home/edwiki

```
-rwxr-xr--  1  edwiki users  1024  Nov  2  00:10  myfile
drwxr-xr--- 1  edwiki users  1024  Nov  2  00:10  mydir
```

Here First column shows Base Permission associated with a file or directory.

The base permissions are broken into three groups, and each position in the group determines a specific permission, in the order: read (r), write (w), execute (x):

- First Character refers its File or Directory
  - - refers its regular file
  - d refers its directory
- The first three characters (2-4) from the base permissions section represent the permissions for the file or directory owner. In the above example, 'myfile' has read , write and execute permission for owner called edwiki
- Next three characters (5-7) from base permissions are for the group to which the file belongs. In the above example, 'myfile' is in oracle group and read, execute permissions are applied for members part of oracle group.
- The next three characters (8-10) from the permissions are for everyone else who are not owners and not part of same file/directory group. In the above example, other get only read permission to access 'myfile'
- In Linux, File and Directory has Read, Write, Execute permissions. However these permissions are differ in meaning and behavior for File and Directory.

**File Permissions:**

- In Linux, Permissions are the first level of security to restrict the access to unauthorized users. Linux Files having **read**, **write**, and **execute** permissions, and the details explanation of these permissions below:

| Permission | Files | Directories |
|---|---|---|
| r | Grants the access to read file i.e. View the contents of the file. ex: $cat | Read permission grant the access to see the list of files and sub-directories in the specific directory. ex: $ls |
| w | Grants the access to modify, or remove the content of the file, ex: | Write permission grant the access to add or delete files of the directory, ex: |

X

**How to change permissions:**

**Chmod:** $ chmod command is used to change the file or directory permissions. There are two way to change the permissions.

1. Symbolic Mode
2. Numeric Mode

**Symbolic Mode**

Operators are used to change the permissions using Symbolic Mode. Below table shows operator supported in Symbolic mode.

| Chmod Operator | Description |
|---|---|
| + | Adds the designated permission(s) to a file or directory |
| - | Removes the designated permission(s) from a file or directory. |
| = | Sets the designated permission(s). |

We can add and remove file permissions using symbolic mode as below

1. chmod u+x file1 - adding execute permission to user only
2. chmod x-w file1 - removing write permission to others.
3. chmod g=rwx file1 - overwriting the group permission with read write and execute.
4. chmod ug+rx file - adding read and execute permission to user and group.
5. chmod go+rw - adding read and write permission to group and others.

**Numeric Mode:**

Numbers are used to change the permissions using Numeric Mode. Each permission is assigned with standard identified number as

Read - 4, Write - 2, Execute - 1

To change the permissions for owner, group, and others need to follow the below table to sum the number required to fill in for each field.

| Number | Permission Representation | Symbolic Reference |
|---|---|---|
| 0 | No Permission | --- |
| 1 | Execute | --x |
| 2 | Write | -w- |
| 3 | Write, Execute | -wx |
| 4 | Read | r-- |
| 5 | Read, Execute | r-x |
| 6 | Read, Write | rw- |
| 7 | Read, Write, Execute | rwx |

[edwiki@ ~]$ls -l f1
-rw---xrw- 1 root root 0 Jun 27 02:04 f1
[edwiki @ ~]$chmod 744 f1
[edwiki @ ~]$ls -l f1
-rwxr--r-- 1 root root 0 Jun 27 02:04 f1
[edwiki @ ~]$

**Special Permissions:**

X

reasons, but when you change your password, you need to have write permission to this file to update the latest new password. This means that the **passwd** program has to give you additional permissions so that you can write to the file /etc/shadow as root privilege. .

Special permissions Set User ID ( SUID) and Set Group ID ( SGID) bits make this possible to grant the program owner permissions to the user who executing the program rather the actual user permissions.

When you execute a program that has the SUID bit enabled, you inherit the permissions of that program's owner. Programs that do not have the SUID bit set are run with the permissions of the user who started the program.

This is true for SGID as well. Normally programs execute with your group permissions, but instead your group will be changed just for this program to the group owner of the program.

The SUID and SGID bits will appear as the letter "s" if the permission is available. The SUID "s" bit will be located in the permission bits where the owners execute permission would normally reside.

```
$ ls -l /usr/bin/passwd
-r-sr-xr-x  1   root       bin   19031 Feb 7 13:47  /usr/bin/passwd*
$
```

Which shows that the SUID bit is set and that the command is owned by the root. A capital letter S in the execute position instead of a lowercase s indicates that the execute bit is not set.

**How to apply SUID/SGID bits:**

```
# Symbolic Method
$ chmod ug+s file
$ ls -l
  drwsr-sr-x 2 root root  4096 Jun 19 06:45 file


#Numeric Method


$ chmod 6742 file
$ ls -l
  drwsr-S-wx 2 root root  4096 Jun 19 06:45 file
```

**Sticky bit**:

If we applied a sticky bit permission to a file or directory, then only root and owner of file or directory can delete the file or directory. If others are having full permission but they cannot delete the file or directory.

Note: By default, /tmp directory having sticky bit permissions.

```
[root@localhost ~]# ls -ld /tmp/
drwxrwxrwt 3 root root 188 Dec 26  2020 /tmp/
[root@localhost ~]#
```

How to give sticky bit permission to directory.

```
 [edwiki@localhost ~]$ ls -ltr
drwxrwxrwx 2 edwiki devops 59 Feb  2 16:49 dir1
[edwiki@localhost ~]$ chmod +t dir1/
[edwiki@localhost ~]$ ls -ltr
drwxrwxrwt 2 edwiki devops 59 Feb  2 16:49 dir1
```

How to remove sticky bit permission to a directory.

```
 [edwiki@localhost ~]$ ls -ld dir1/
drwxrwxrwt 2 edwiki devops 59 Feb  2 16:49 dir1/
[edwiki@localhost ~]$ chmod -t dir1/
[edwiki@localhost ~]$ ls -ld dir1/
drwxrwxrwx 2 edwiki devops 59 Feb  2 16:49 dir1/
```

X

2. **System User:** System Users are those needed for the operation of system-specific components for example mail accounts and the sshd accounts. These System Users are created automatically with the specific software installation to manage the software programs. Usually we should not edit any property of System Users as that would effect the system behaviour.
3. **Normal User :** User Accounts that are created manually to access the system called as Normal Users. Normal Users granted with interactive shell access to type commands to complete certain tasks. Usually Normal Users are created to grant and restrict access to certain files are directories within the Linux System.

**Users Attributes:**

**Whenever a user is created in Linux things created by default:-**

1. A home directory is created(/home/username)
2. A mail box is created(/var/spool/mail)
3. unique UID & GID are given to user

**Linux uses UPG (User Private Group) scheme**

1. It means that whenever a user is created is has its own private group
   1. For Example if a user is created with the name **edwiki ,** then a primary group for that user will be **edwiki** only

**What is Group?**

Linux supports creating Group which logically grouping multiple users. Every user must be part of at least one group called as Primary Group.

Linux has two type of Groups:

1. Primary Group
2. Secondary Group

Groups are useful to manage file or directory permissions effectively as it allows to grant the permission to set of users who are part of the same Group.

**Primary Group:**

Every user is part of at least one group as Primary Group. When user creats file or directory then the default group of file or directory is the primary group of the owner.

**Secondary Group**

It's possible to create groups and assign users to the group. Secondary groups are more useful to grant additional permissions to the user to access files and directories of the same group.

**Commands:**

1. Useradd: We can create a new user by crating useradd command.
2. passwd: we can crate password for user by using passwd command.
3. groupadd: We can create a group by creating a groupadd command.

Creating new user by using useradd command and create password by using passwd command.

```
[root@localhost ~]# useradd edwiki
[root@localhost ~]# passwd edwiki
Changing password for user edwiki.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a diction
ary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

How to switch a edwiki user.

```
[root@localhost ~]# su edwiki
[edwiki@localhost ~]$ pwd
```

X

How to change user primary group.

```
[root@localhost ~]# id edwiki
uid=1000(edwiki) gid=1000(edwiki) groups=1000(edwiki)
[root@localhost ~]# usermod -g devops edwiki
[root@localhost ~]# id edwiki
uid=1000(edwiki) gid=1002(devops) groups=1002(devops)
```

Changing a user secondary group.

```
[root@localhost ~]# id edwiki
uid=1001(edwiki) gid=1001(edwiki) groups=1001(edwiki)
[root@localhost ~]# usermod -G devops edwiki
[root@localhost ~]# id edwiki1
uid=1001(edwiki) gid=1001(edwiki) groups=1001(edwiki),1002(devops)
```

Note: A user you can part of one primary group and can part of one more secondary groups.

Changing file and directory ownership.

```
[root@localhost dir1]# ls -l file1
---------- 1 edwiki devops 0 Feb  2 16:51 file1
[root@localhost dir1]# chown root file1
[root@localhost dir1]# ls -l file1
---------- 1 root devops 0 Feb  2 16:51 file1
```

Changing file and directory group ownership.

```
[root@localhost dir1]# ls -l file1
---------- 1 root devops 0 Feb  2 16:51 file1
[root@localhost dir1]#
[root@localhost dir1]# chgrp root file1
[root@localhost dir1]# ls -l file1
---------- 1 root root 0 Feb  2 16:51 file1
```

Changing files and directory owner and group ownership recursively.

```
[root@localhost edwiki]# ls -ld dir1/
drwxrwxrwx 2 edwiki devops 59 Feb  2 16:49 dir1/
[root@localhost edwiki]# chown -R edwiki:edwiki dir1/
[root@localhost edwiki]# ls -ld dir1/
drwxrwxrwx 2 edwiki edwiki 59 Feb  2 16:49 dir1/
[root@localhost edwiki]# cd dir1/
[root@localhost dir1]# ls -l file1
---------- 1 edwiki edwiki 0 Feb  2 16:51 file1
```

Previous Next