IT2654

Systems administration & security

Topic 3: LINUX FILE PERMISSION & DISK QUOTA

Objectives

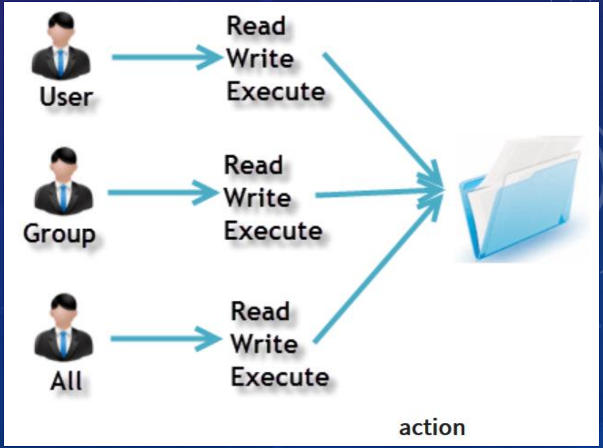
- Manage file security
  - Ownership and Permission
  - SUID, SGID and Sticky Bit
- Implement Disk Quota

Linux user security

- Linux is a multi-user system
- User's security is defined into 2 levels:
  - Ownership
  - Permission
- Ownership – every file and directory is assigned to 3 types of owners:
  1. User (owner)
  2. Group
  3. Others (All others)

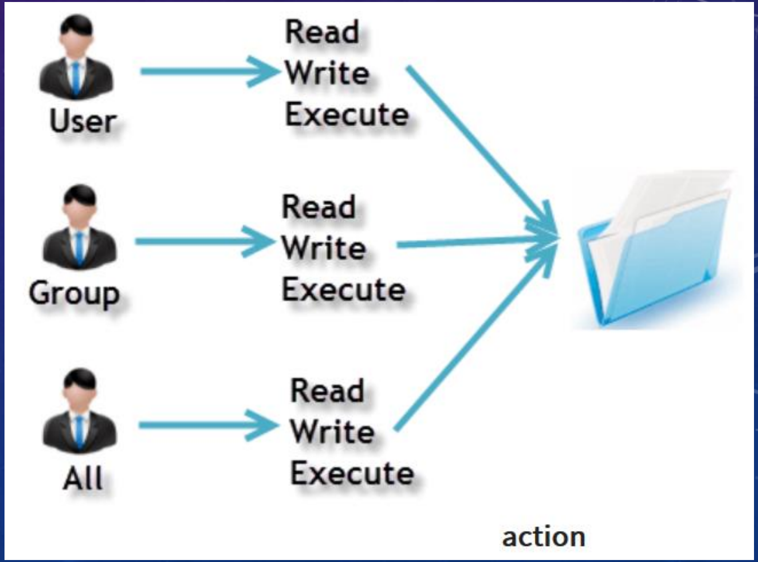
Linux ownership

- User – owner of the file. By default the user who created the file is the owner.
- Group – a group can contain one or many users. All users in a group will have the same access permissions.
- Other – anyone else who is not the user (owner) or belong in the group.

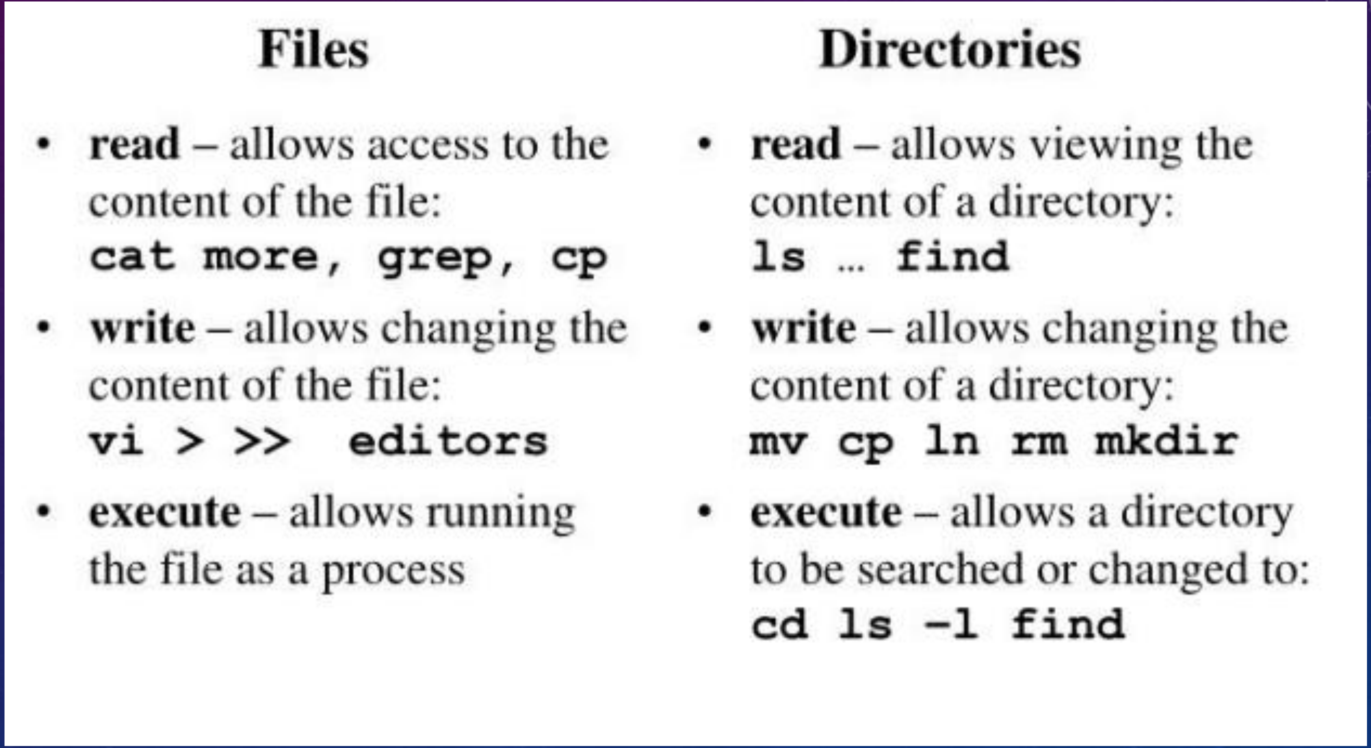


Linux Permission

- Every file and directory in Linux has 3 types of permissions:
  1. Read
  2. Write
  3. Execute

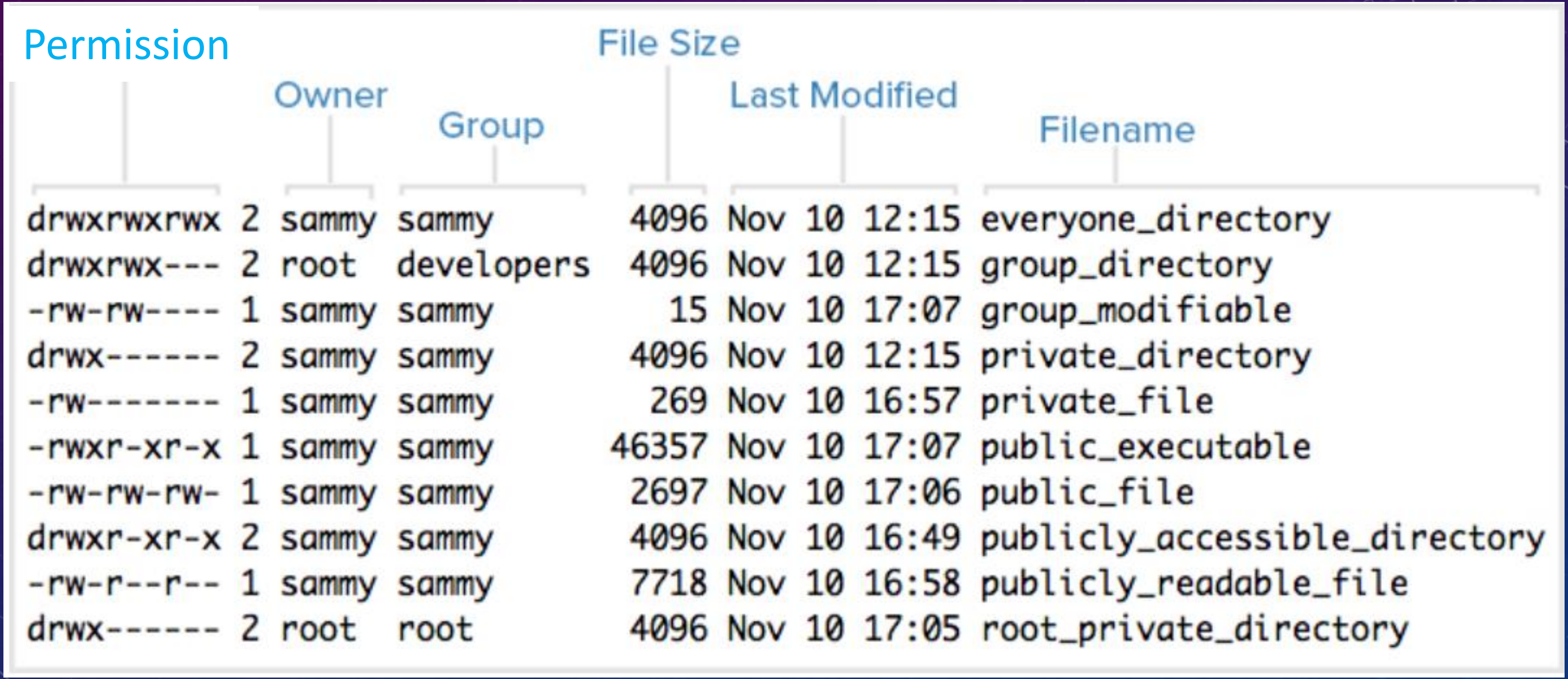


File vs Directory Permissions



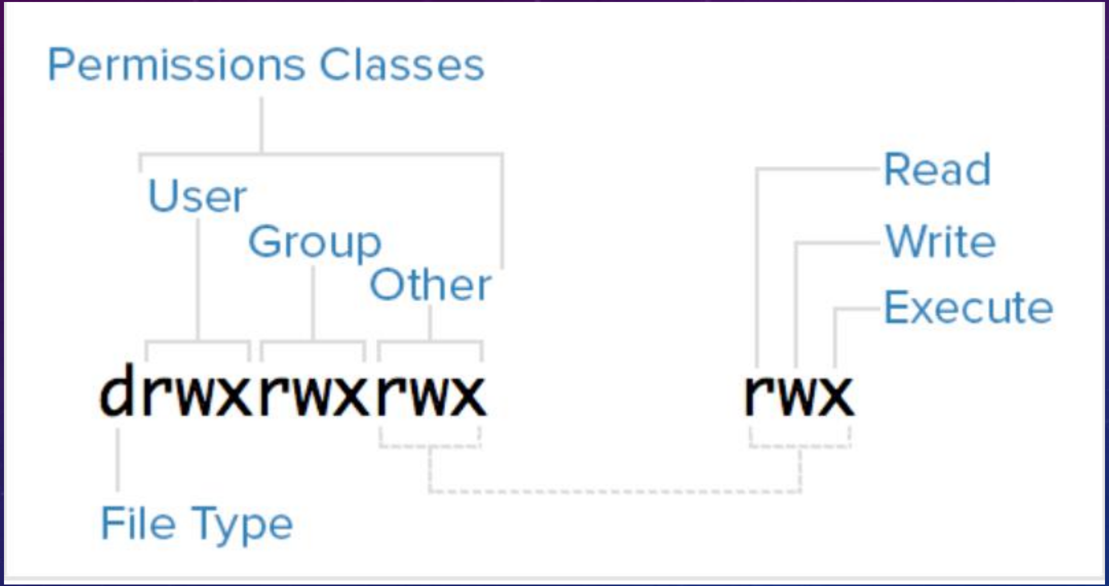
Files	Directories
• **read** – allows access to the content of the file: `cat more, grep, cp`	• **read** – allows viewing the content of a directory: `ls … find`
• **write** – allows changing the content of the file: `vi > >> editors`	• **write** – allows changing the content of a directory: `mv cp ln rm mkdir`
• **execute** – allows running the file as a process	• **execute** – allows a directory to be searched or changed to: `cd ls -l find`

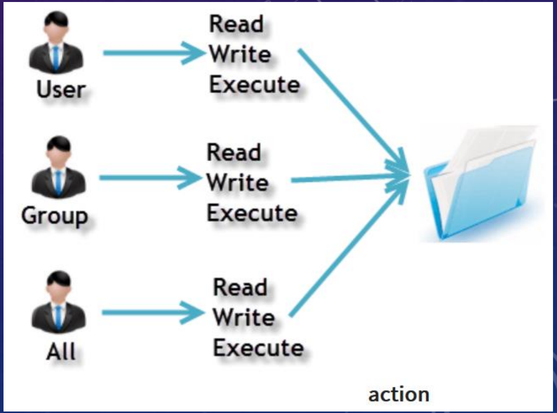
View Permission : ls -l



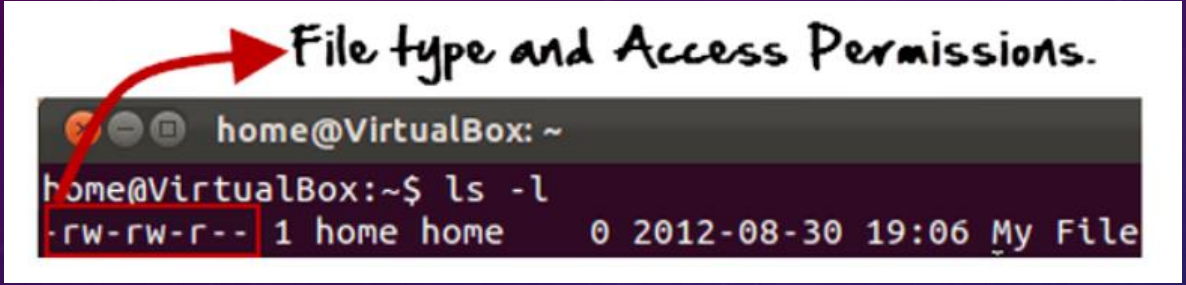
```
drwxrwxrwx 2 sammy sammy       4096 Nov 10 12:15 everyone_directory
drwxrwx--- 2 root  developers  4096 Nov 10 12:15 group_directory
-rw-rw---- 1 sammy sammy         15 Nov 10 17:07 group_modifiable
drwx------ 2 sammy sammy       4096 Nov 10 12:15 private_directory
-rw------- 1 sammy sammy        269 Nov 10 16:57 private_file
-rwxr-xr-x 1 sammy sammy      46357 Nov 10 17:07 public_executable
-rw-rw-rw- 1 sammy sammy       2697 Nov 10 17:06 public_file
drwxr-xr-x 2 sammy sammy       4096 Nov 10 16:49 publicly_accessible_directory
-rw-r--r-- 1 sammy sammy       7718 Nov 10 16:58 publicly_readable_file
drwx------ 2 root  root        4096 Nov 10 17:05 root_private_directory
```

Permission

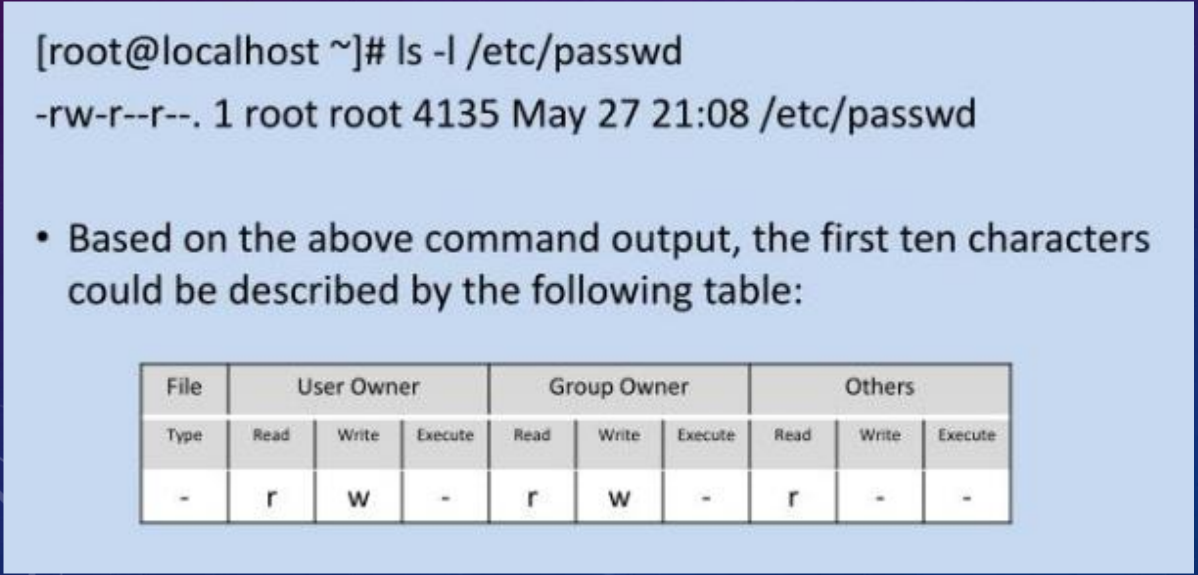

Permissions Classes
User
Group
Other
drwxrwxrwx
rwx
Read
Write
Execute
File Type


User
Read
Write
Execute
Group
Read
Write
Execute
All
Read
Write
Execute
action

Viewing Permissions



```
home@VirtualBox:~$ ls -l
-rw-rw-r-- 1 home home     0 2012-08-30 19:06 My File
```



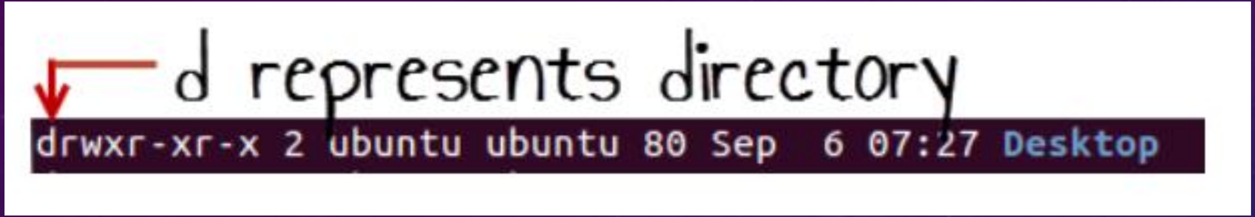
```
[root@localhost ~]# ls -l /etc/passwd
-rw-r--r--. 1 root root 4135 May 27 21:08 /etc/passwd
```

- Based on the above command output, the first ten characters could be described by the following table:

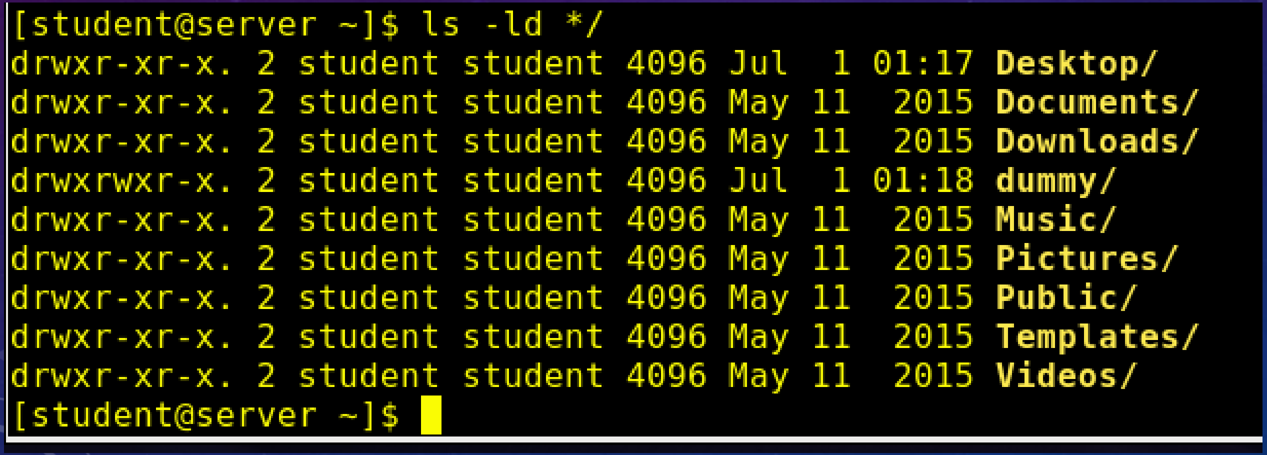
File	User Owner			Group Owner			Others		
Type	Read	Write	Execute	Read	Write	Execute	Read	Write	Execute
-	r	w	-	r	w	-	r	-	-

r	= read permission
w	= write permission
x	= execute permission
-	= no permission

Listing Directory - ls –ld */



```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep  6 07:27 Desktop
```



```
[student@server ~]$ ls -ld */
drwxr-xr-x. 2 student student 4096 Jul  1 01:17 Desktop/
drwxr-xr-x. 2 student student 4096 May 11  2015 Documents/
drwxr-xr-x. 2 student student 4096 May 11  2015 Downloads/
drwxrwxr-x. 2 student student 4096 Jul  1 01:18 dummy/
drwxr-xr-x. 2 student student 4096 May 11  2015 Music/
drwxr-xr-x. 2 student student 4096 May 11  2015 Pictures/
drwxr-xr-x. 2 student student 4096 May 11  2015 Public/
drwxr-xr-x. 2 student student 4096 May 11  2015 Templates/
drwxr-xr-x. 2 student student 4096 May 11  2015 Videos/
[student@server ~]$ 
```

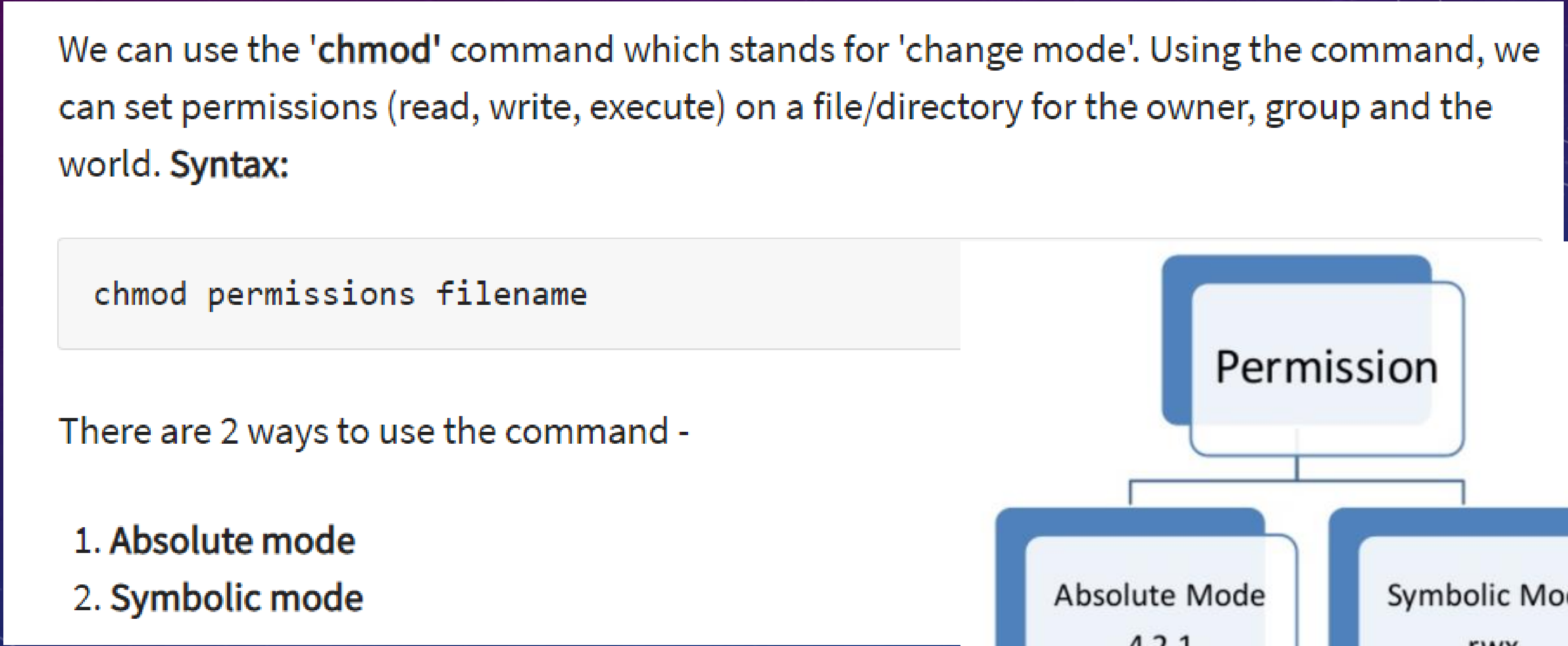
r = read permission

w = write permission

x = execute permission

- = no permission

Changing Permissions with chmod

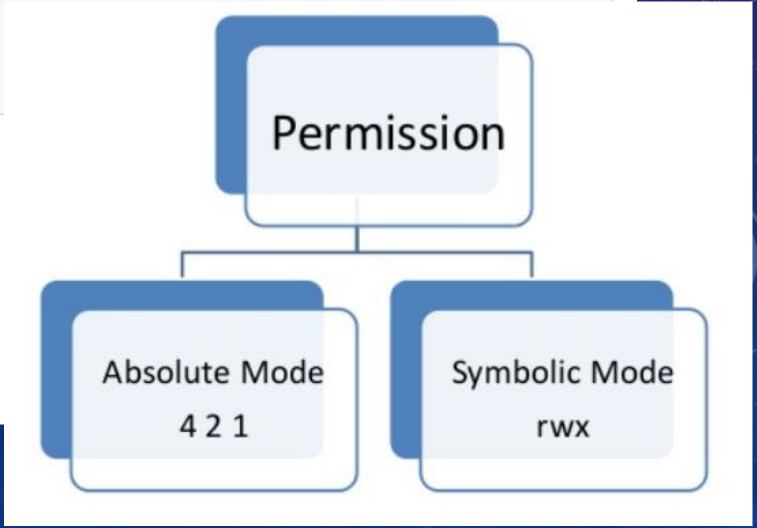


We can use the '**chmod**' command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world. **Syntax:**

```
chmod permissions filename
```

There are 2 ways to use the command -

1. **Absolute mode**
2. **Symbolic mode**

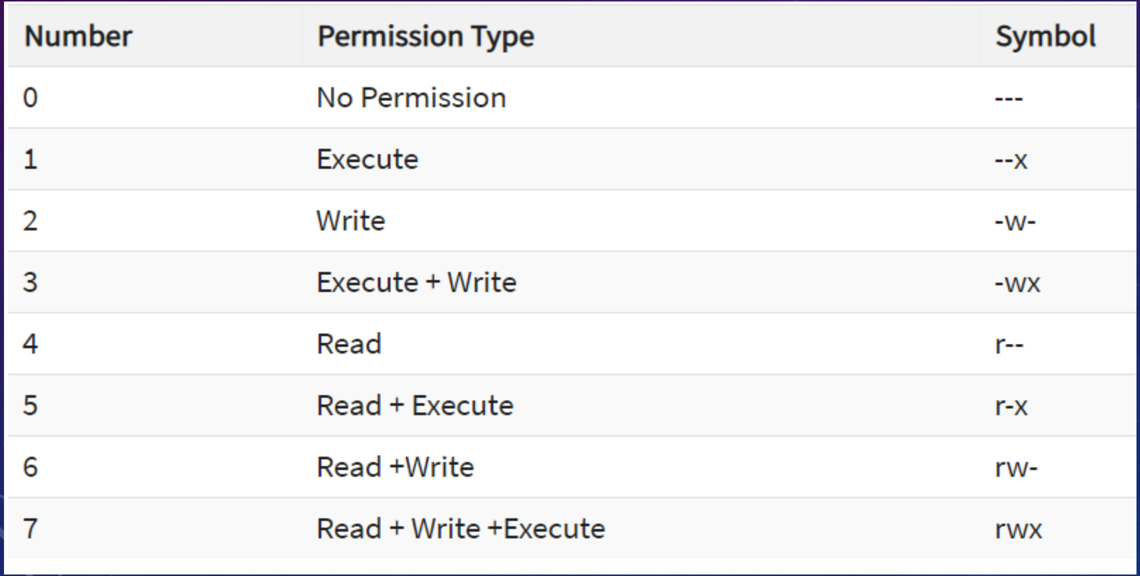


Absolute(Numeric) Mode

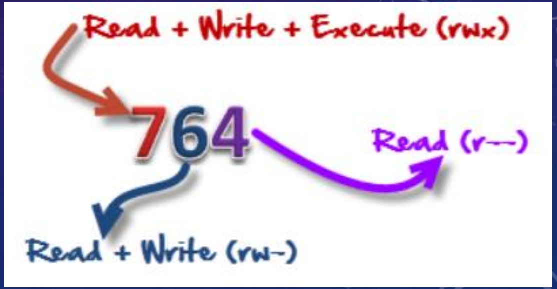
In this mode, file **permissions are not represented as characters but a three-digit octal number**.

Syntax: chmod ??? <filename>

where ? Is from 0 to 7



Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx



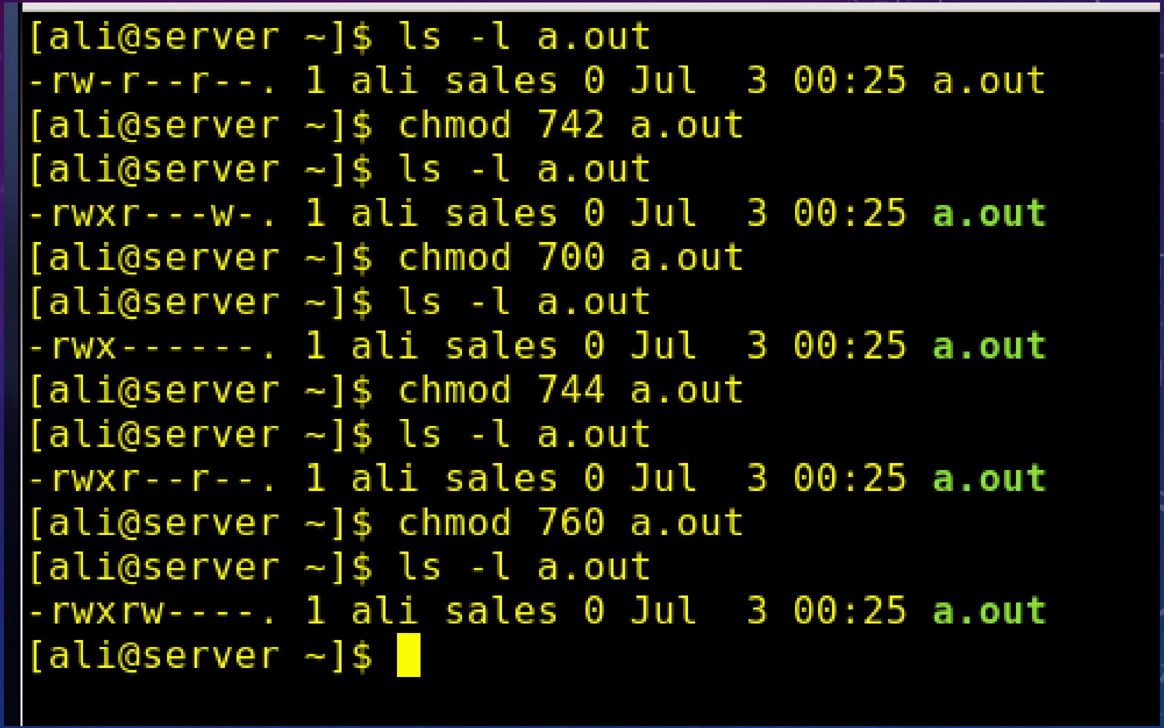
- Owner can read, write and execute
- Usergroup can read and write
- World can only read

Examples

`chmod 742 a.out`

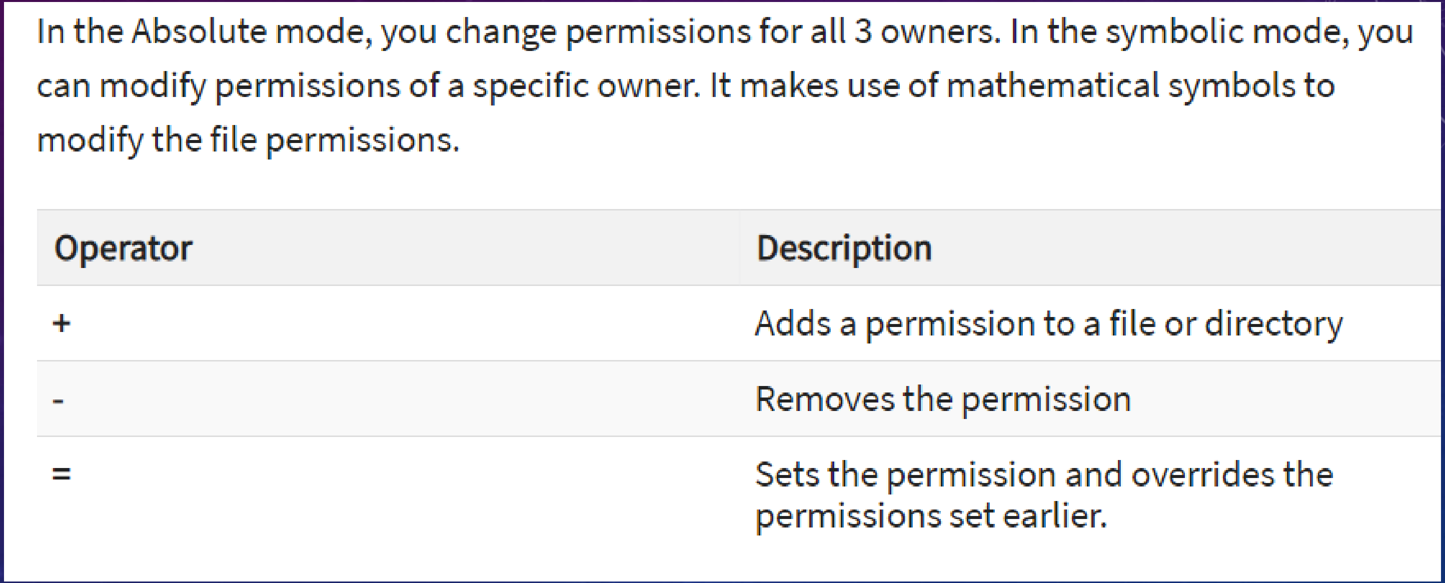
7 – owner permission
4 – group permission
2 – others permission

7 = rwx
4 = r
2 = w



```
[ali@server ~]$ ls -l a.out
-rw-r--r--. 1 ali sales 0 Jul  3 00:25 a.out
[ali@server ~]$ chmod 742 a.out
[ali@server ~]$ ls -l a.out
-rwxr---w-. 1 ali sales 0 Jul  3 00:25 a.out
[ali@server ~]$ chmod 700 a.out
[ali@server ~]$ ls -l a.out
-rwx------. 1 ali sales 0 Jul  3 00:25 a.out
[ali@server ~]$ chmod 744 a.out
[ali@server ~]$ ls -l a.out
-rwxr--r--. 1 ali sales 0 Jul  3 00:25 a.out
[ali@server ~]$ chmod 760 a.out
[ali@server ~]$ ls -l a.out
-rwxrw----. 1 ali sales 0 Jul  3 00:25 a.out
[ali@server ~]$
```

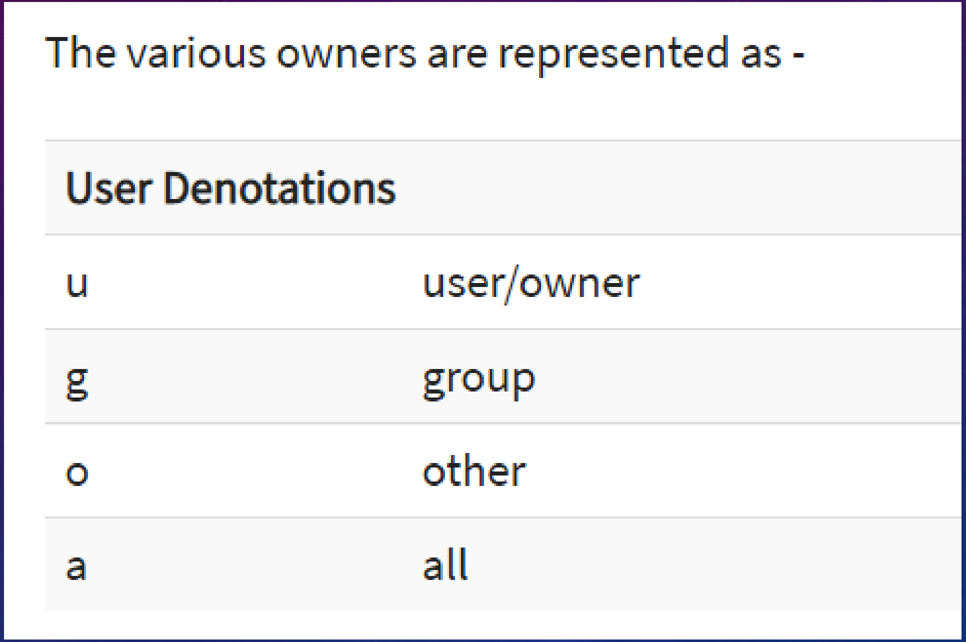
Symbolic Mode



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 file permissions.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission
=	Sets the permission and overrides the permissions set earlier.

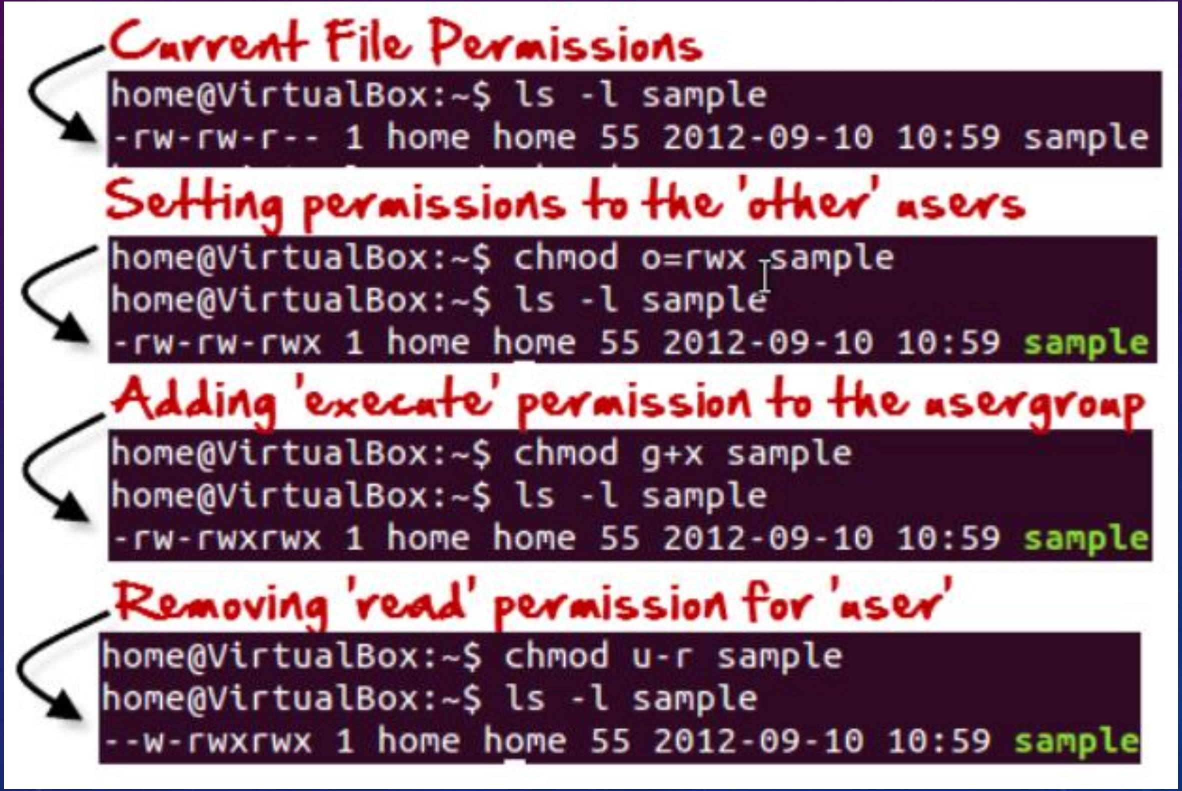
Symbolic Mode



The various owners are represented as -

User Denotations	
u	user/owner
g	group
o	other
a	all

Symbolic Mode


Current File Permissions

```
home@VirtualBox:~$ ls -l sample
-rw-rw-r-- 1 home home 55 2012-09-10 10:59 sample
```

Setting permissions to the 'other' users

```
home@VirtualBox:~$ chmod o=rwx sample
home@VirtualBox:~$ ls -l sample
-rw-rw-rwx 1 home home 55 2012-09-10 10:59 sample
```

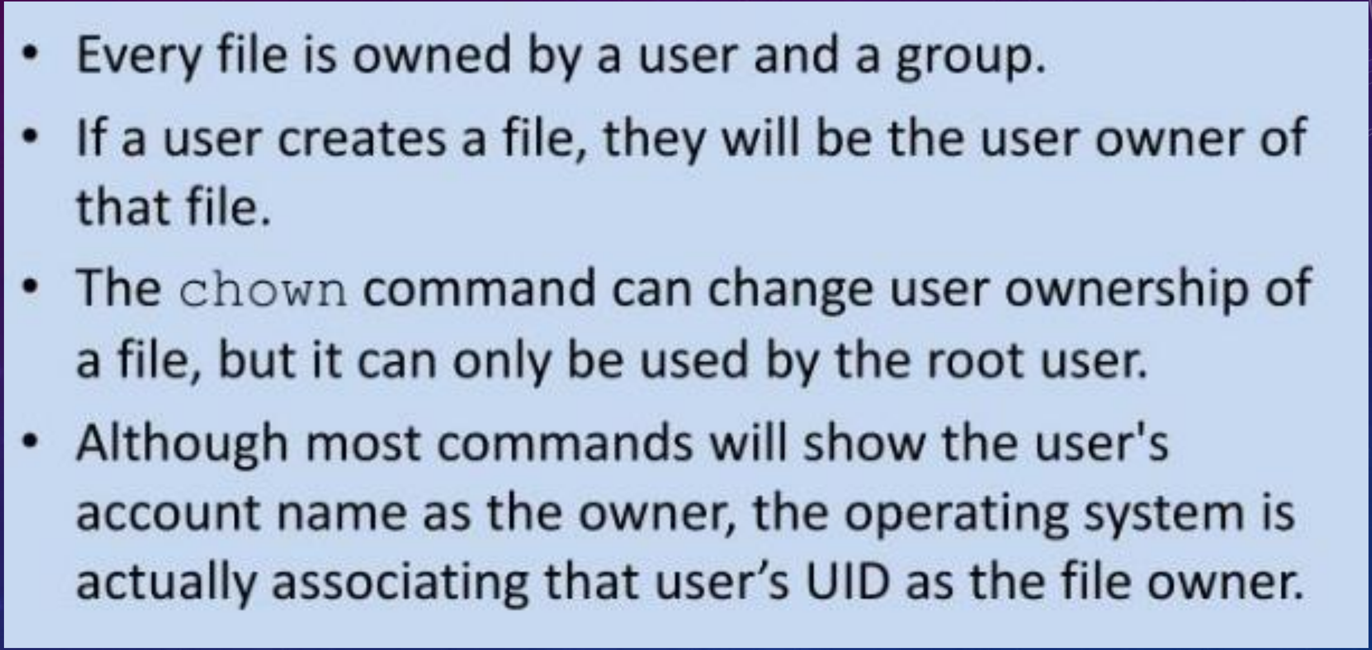
Adding 'execute' permission to the usergroup

```
home@VirtualBox:~$ chmod g+x sample
home@VirtualBox:~$ ls -l sample
-rw-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

Removing 'read' permission for 'user'

```
home@VirtualBox:~$ chmod u-r sample
home@VirtualBox:~$ ls -l sample
--w-rwxrwx 1 home home 55 2012-09-10 10:59 sample
```

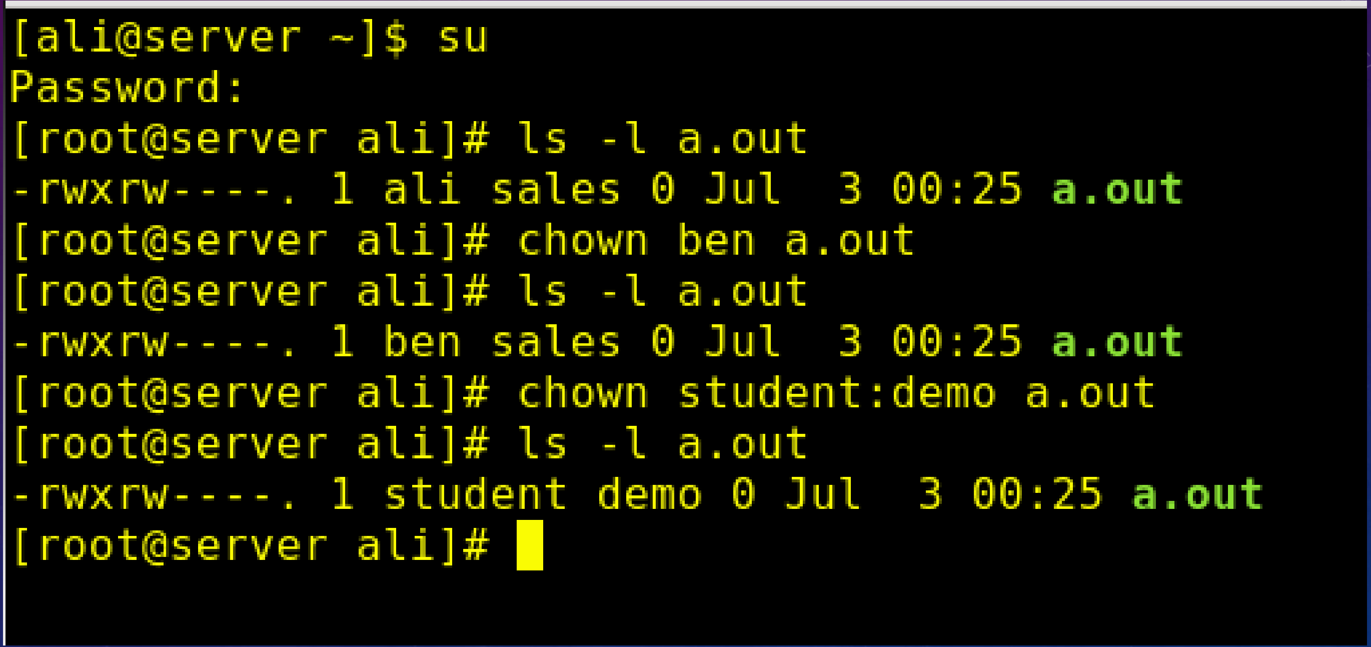
File Ownership



- Every file is owned by a user and a group.
- If a user creates a file, they will be the user owner of that file.
- The `chown` command can change user ownership of a file, but it can only be used by the root user.
- Although most commands will show the user's account name as the owner, the operating system is actually associating that user’s UID as the file owner.

Change Owner: `chown`

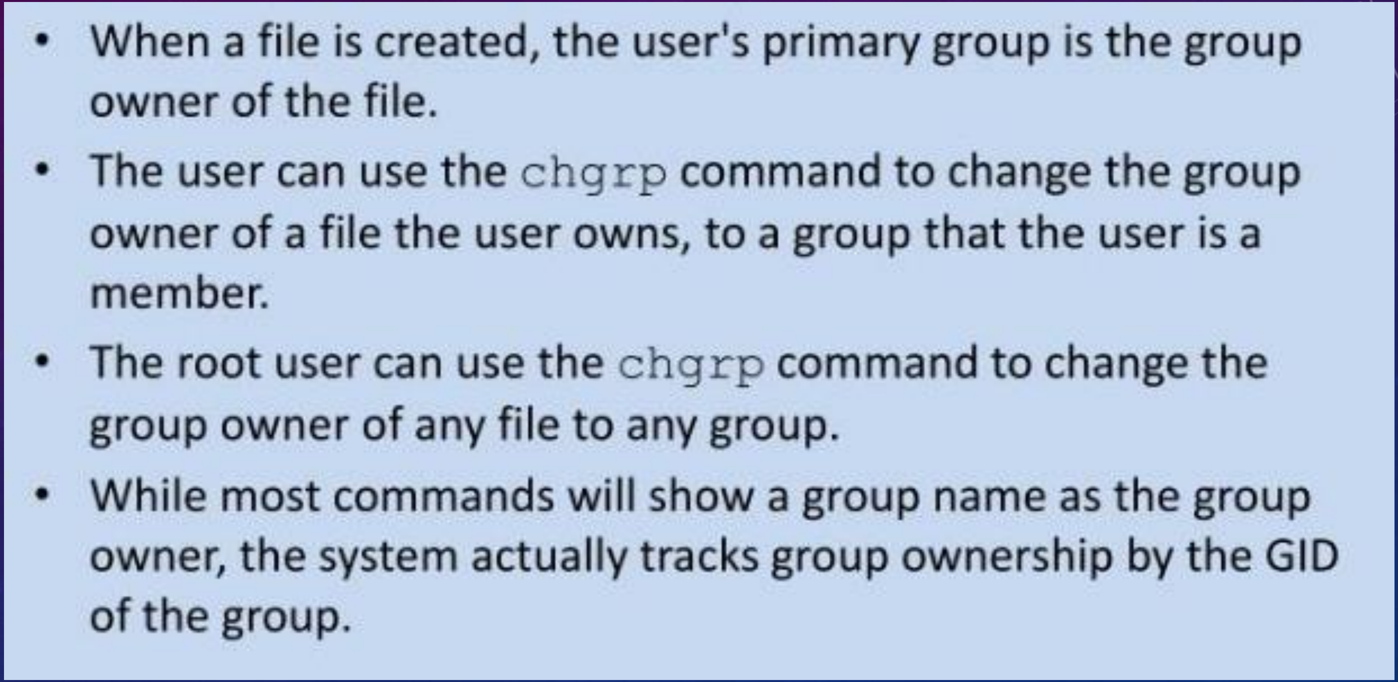
- Syntax: `chown [owner]:[group] <filename>`



```
[ali@server ~]$ su
Password:
[root@server ali]# ls -l a.out
-rwxrw----. 1 ali sales 0 Jul  3 00:25 a.out
[root@server ali]# chown ben a.out
[root@server ali]# ls -l a.out
-rwxrw----. 1 ben sales 0 Jul  3 00:25 a.out
[root@server ali]# chown student:demo a.out
[root@server ali]# ls -l a.out
-rwxrw----. 1 student demo 0 Jul  3 00:25 a.out
[root@server ali]# 
```

Only root can change ownership of file

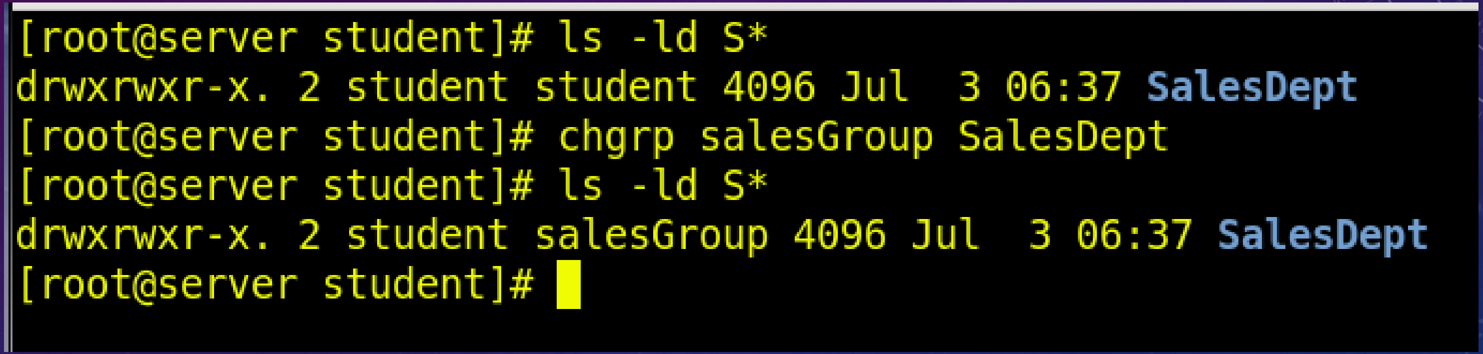
Group Ownership



- When a file is created, the user's primary group is the group owner of the file.
- The user can use the `chgrp` command to change the group owner of a file the user owns, to a group that the user is a member.
- The root user can use the `chgrp` command to change the group owner of any file to any group.
- While most commands will show a group name as the group owner, the system actually tracks group ownership by the GID of the group.

Change Group: `chgrp`

- Syntax: `chgrp [options] <groupname> <filename>`



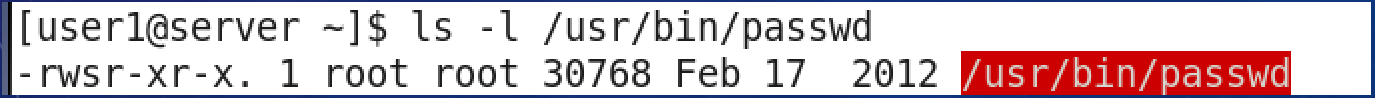
```
[root@server student]# ls -ld S*
drwxrwxr-x. 2 student student 4096 Jul  3 06:37 SalesDept
[root@server student]# chgrp salesGroup SalesDept
[root@server student]# ls -ld S*
drwxrwxr-x. 2 student salesGroup 4096 Jul  3 06:37 SalesDept
[root@server student]# 
```

- Changes the group ownership of the SalesDept directory from student to salesGroup
- Cannot have 2 groups owning the same file.
- No nested groups in Linux. One group cannot be sub-group of other

setuid & setgid

- **setuid** and **setgid**
  - short for **set user ID** upon execution and **set group ID** upon execution, respectively
  - Unix access rights flags that allow users to run an executable with the permissions of the executable's owner or group.

- setuid are needed for tasks that require higher privileges than those which common users have.
- When setuid bit is set, an executable when launched does not run with the privileges of the user who launched it, but with that of the file owner instead.
- For example, if an executable has the setuid bit set on it, and it's owned by root, when launched by a normal user, it will run with root privileges.
  - Example: passwd and ping command
  - # **chmod u+s <filename>**



```
[user1@server ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 Feb 17  2012 /usr/bin/passwd
```

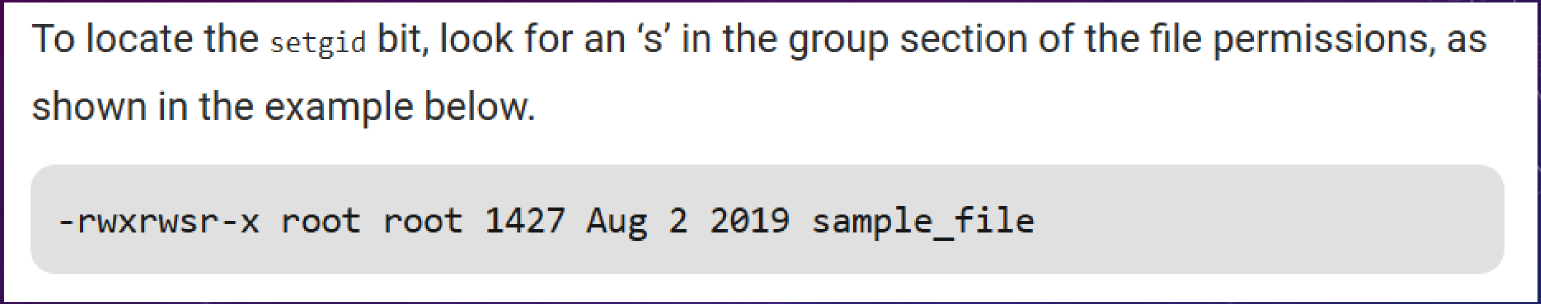
setgid

- setgid most of the time is used to create a collaborative directory
  - When a file is created in a directory with the setgid bit set, it belongs to the same group as the group directory, rather the creator’s primary group
- The setgid affects both files as well as directories.
- When used on a file, it executes with the privileges of the group of the user who owns it instead of executing with those of the group of the user who executed it.

`setgid` on Directories

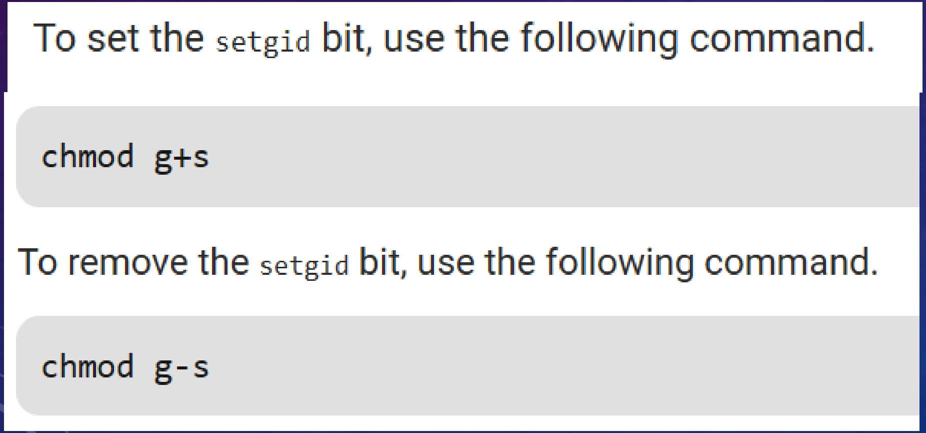
- The setgid flag, when set on a directory, have an entirely different meaning.
  - `chmod g+s <directoryname>`
    - Causes new files and subdirectories created within it to **inherit the groupID** of the folder, rather than the primary groupID of the user who created the file.
    - Newly created subdirectories inherit the setgid bit.
    - existing entities (files/subdirectories) will not be affected.
    - This is used for file sharing since they can be now modified by all the users who are part of the group of the parent directory.

setgid



To locate the `setgid` bit, look for an ‘s’ in the group section of the file permissions, as shown in the example below.

```
-rwxrwsr-x root root 1427 Aug 2 2019 sample_file
```



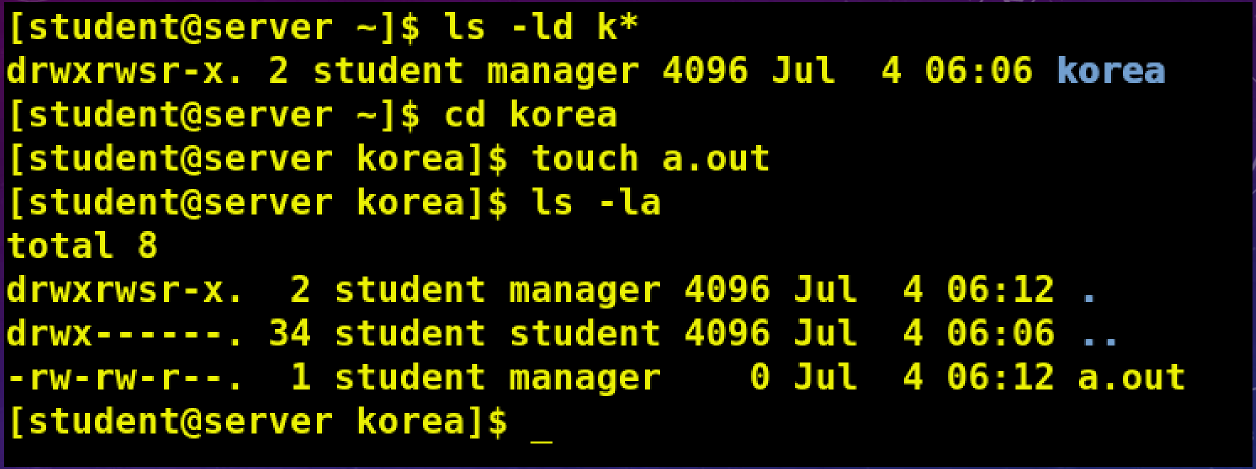
To set the `setgid` bit, use the following command.

```
chmod g+s
```

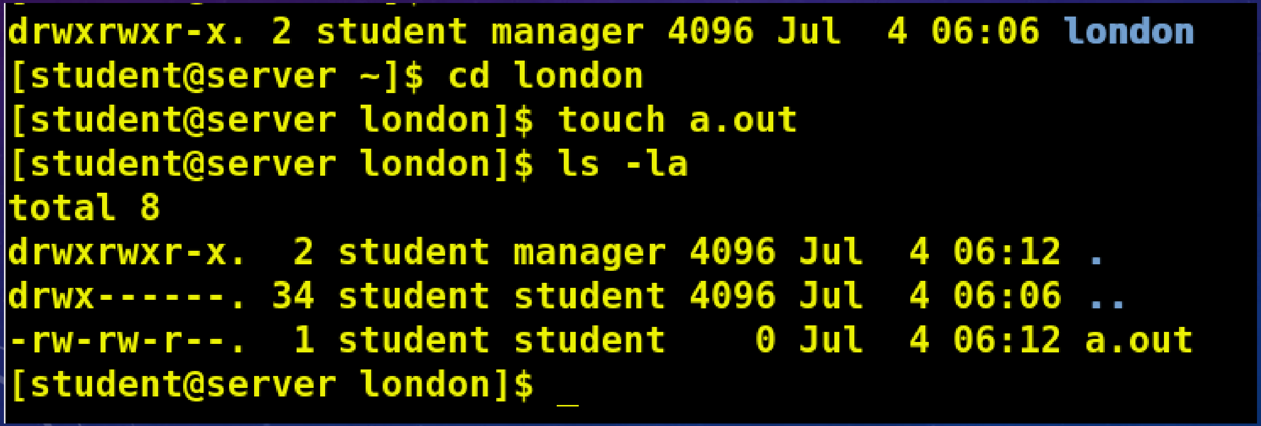
To remove the `setgid` bit, use the following command.

```
chmod g-s
```

Demo



```
[student@server ~]$ ls -ld k*
drwxrwsr-x. 2 student manager 4096 Jul  4 06:06 korea
[student@server ~]$ cd korea
[student@server korea]$ touch a.out
[student@server korea]$ ls -la
total 8
drwxrwsr-x.  2 student manager 4096 Jul  4 06:12 .
drwx------. 34 student student 4096 Jul  4 06:06 ..
-rw-rw-r--.  1 student manager    0 Jul  4 06:12 a.out
[student@server korea]$ _
```



```
drwxrwxr-x. 2 student manager 4096 Jul  4 06:06 london
[student@server ~]$ cd london
[student@server london]$ touch a.out
[student@server london]$ ls -la
total 8
drwxrwxr-x.  2 student manager 4096 Jul  4 06:12 .
drwx------. 34 student student 4096 Jul  4 06:06 ..
-rw-rw-r--.  1 student student    0 Jul  4 06:12 a.out
[student@server london]$ _
```

How to set SUID and SGID

- The setuid and setgid bits are normally set with the command chmod by setting the high-order octet
  - 4 (for setuid) or 2 (for setgid) or 6 (for both).
- For example: chmod 6711
  - set both the setuid and setgid bit
  - make the file read/write/executable for the owner
  - executable by the group
  - executable by others.
- All chmod flags are octal, and the least significant bit of the high-order octet is for the sticky bit.

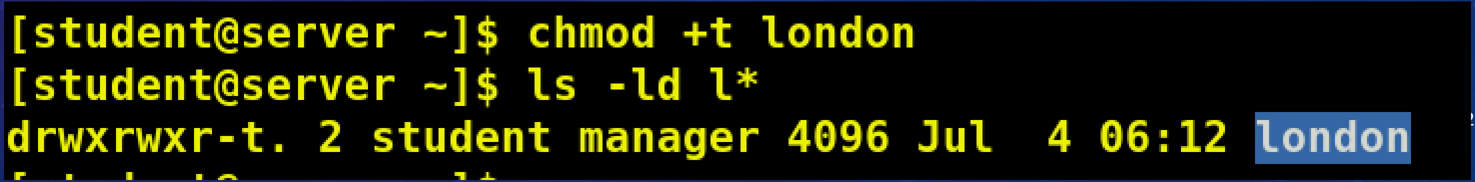
SUID Security Issues

- When a executable file has been given the SUID attribute, normal users on the system who have permission to execute this file gain the privileges of the user who owns the file (commonly root) within the created process.
- When root privileges have been gained within the process, the application can then perform tasks on the system that regular users normally would be restricted from doing.
- Risky when program is not carefully designed. Users may exploit this and gain permanent elevated privileges.
- Typical attack: buffer overflow attack.
  - Successful buffer overrun attacks on vulnerable applications allow the attacker to execute arbitrary code under the rights of the process being exploited.

Sticky Bit

- **sticky bit** is on directories:
  - If not set, users with write permissions to a directory can delete any file in that directory regardless of that files permissions or ownership.
  - When set, only the item's owner, the directory's owner, or the root can rename or delete a file under the directory.
  - Typically it is set on the /tmp to prevent ordinary users from deleting or moving other users' files.

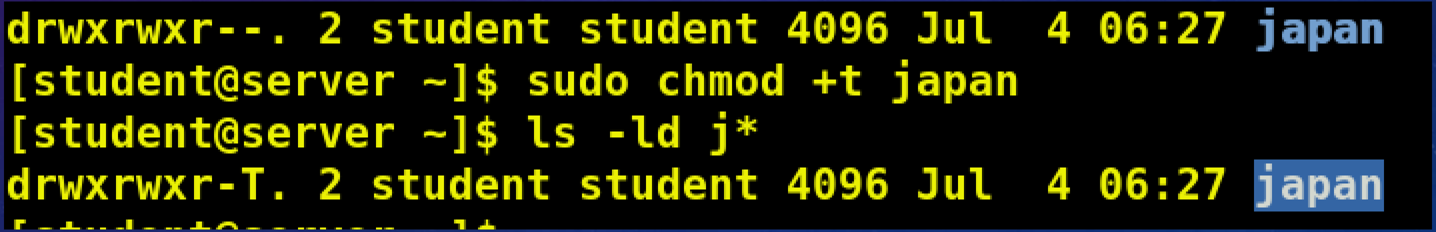
  - # **chmod o+t <directoryname>** or
  - # **chmod 1xxx <directoryname>**



```
[student@server ~]$ chmod +t london
[student@server ~]$ ls -ld l*
drwxrwxr-t. 2 student manager 4096 Jul  4 06:12 london
```

Sticky Bit

- In Unix notation, the sticky bit is represented by the letter **t** in the final character-place.
  - e.g. drwxrwxrwt
- If the sticky-bit is set on a directory without the execution bit set for the *others* category, it is indicated with a capital **T**:



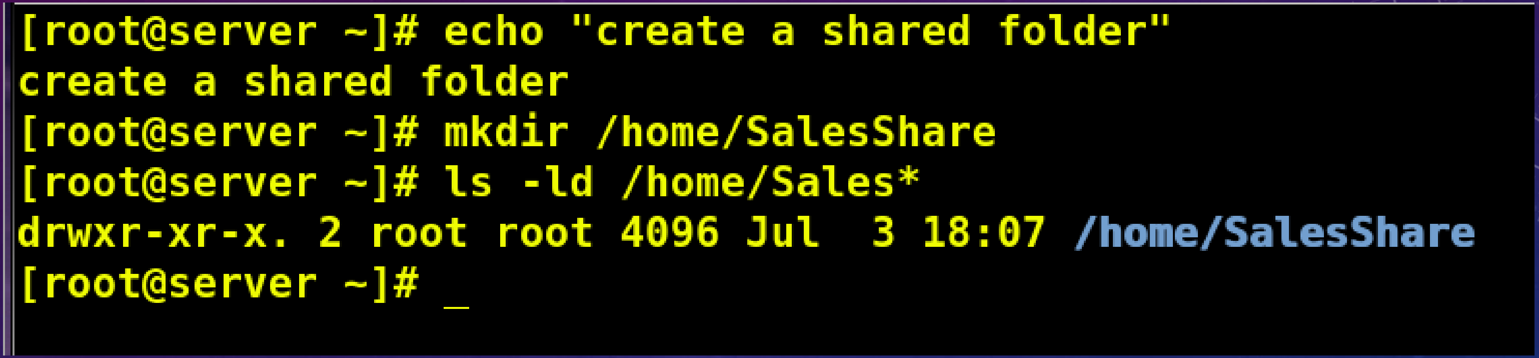
```
drwxrwxr--. 2 student student 4096 Jul  4 06:27 japan
[student@server ~]$ sudo chmod +t japan
[student@server ~]$ ls -ld j*
drwxrwxr-T. 2 student student 4096 Jul  4 06:27 japan
```

Shared Groups

- When multiple users need access the same set of directories of files, we need to create shared folders to be used by the users.
- In Linux the concept of users and groups can give certain level of permissions that will enable them to share data.
- Below are the steps how to create the shared folders where users can and update the files individually.
  1) Create the folder to be shared
  2) Create a user group for sharing purpose
  3) Set permissions of group
  4) Add users to the group

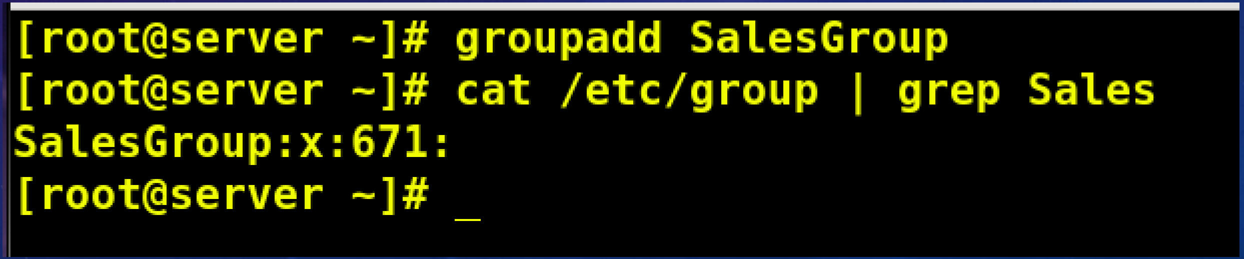
Example:

Step 1: Create a folder for sharing



```
[root@server ~]# echo "create a shared folder"
create a shared folder
[root@server ~]# mkdir /home/SalesShare
[root@server ~]# ls -ld /home/Sales*
drwxr-xr-x. 2 root root 4096 Jul  3 18:07 /home/SalesShare
[root@server ~]# _
```

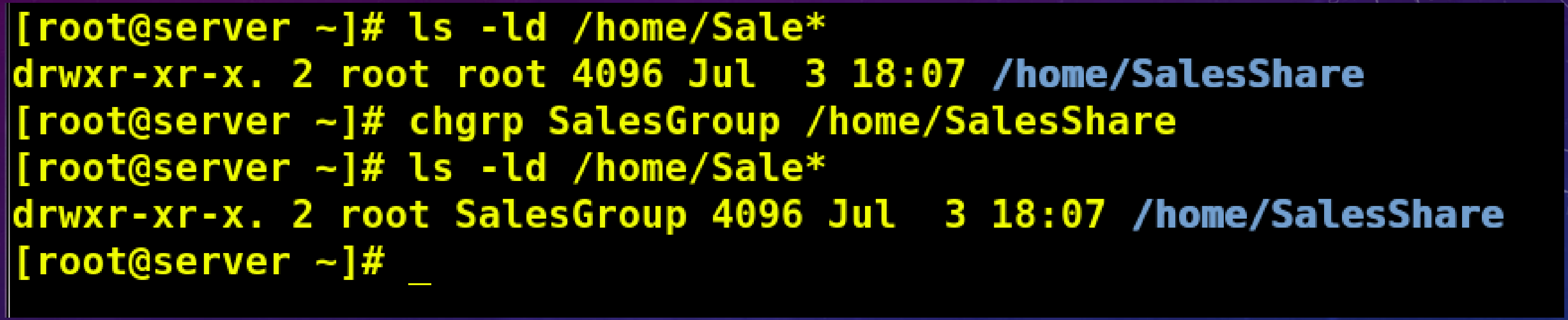
Step 2: Create a group for sharing



```
[root@server ~]# groupadd SalesGroup
[root@server ~]# cat /etc/group | grep Sales
SalesGroup:x:671:
[root@server ~]# _
```

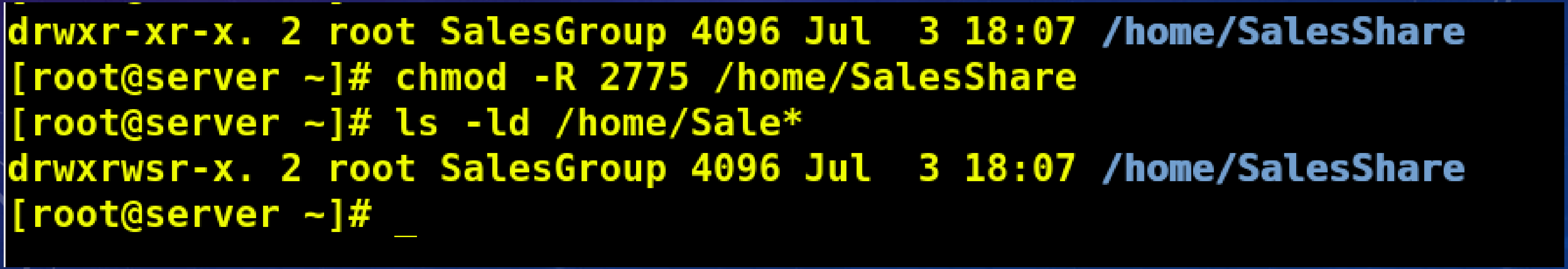
Example:

Step 3a: Change group ownership of shared folder to (sharing) group



```
[root@server ~]# ls -ld /home/Sale*
drwxr-xr-x. 2 root root 4096 Jul  3 18:07 /home/SalesShare
[root@server ~]# chgrp SalesGroup /home/SalesShare
[root@server ~]# ls -ld /home/Sale*
drwxr-xr-x. 2 root SalesGroup 4096 Jul  3 18:07 /home/SalesShare
[root@server ~]# _
```

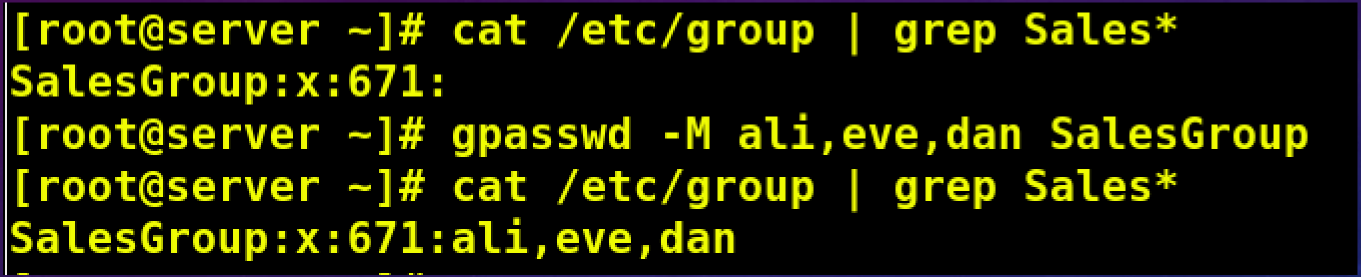
Step 3b: Set the Sticky bit of Group for directory and all its sub-directories (or chmod g+ws)



```
drwxr-xr-x. 2 root SalesGroup 4096 Jul  3 18:07 /home/SalesShare
[root@server ~]# chmod -R 2775 /home/SalesShare
[root@server ~]# ls -ld /home/Sale*
drwxrwsr-x. 2 root SalesGroup 4096 Jul  3 18:07 /home/SalesShare
[root@server ~]# _
```

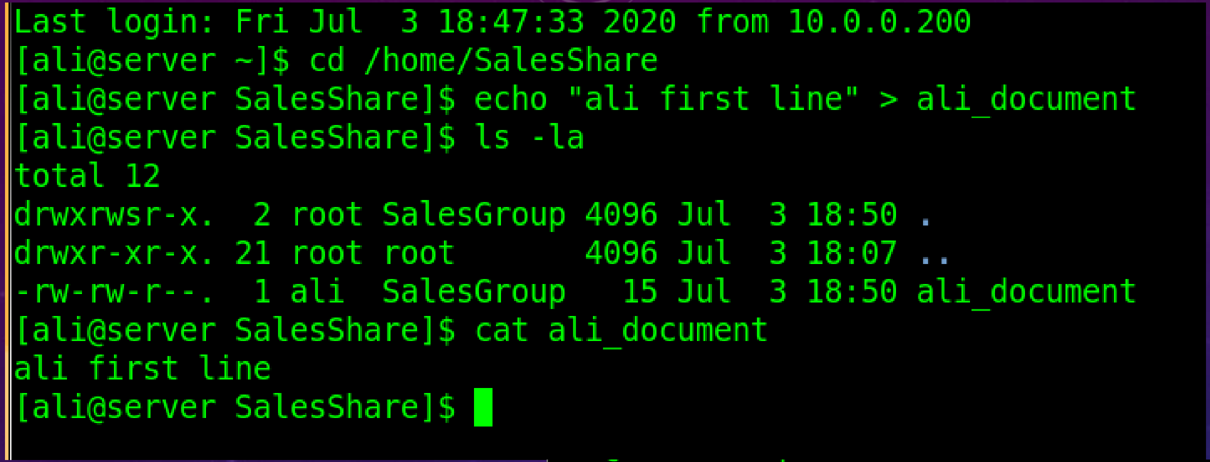
Example:

Step 4: Add users to the sharing group



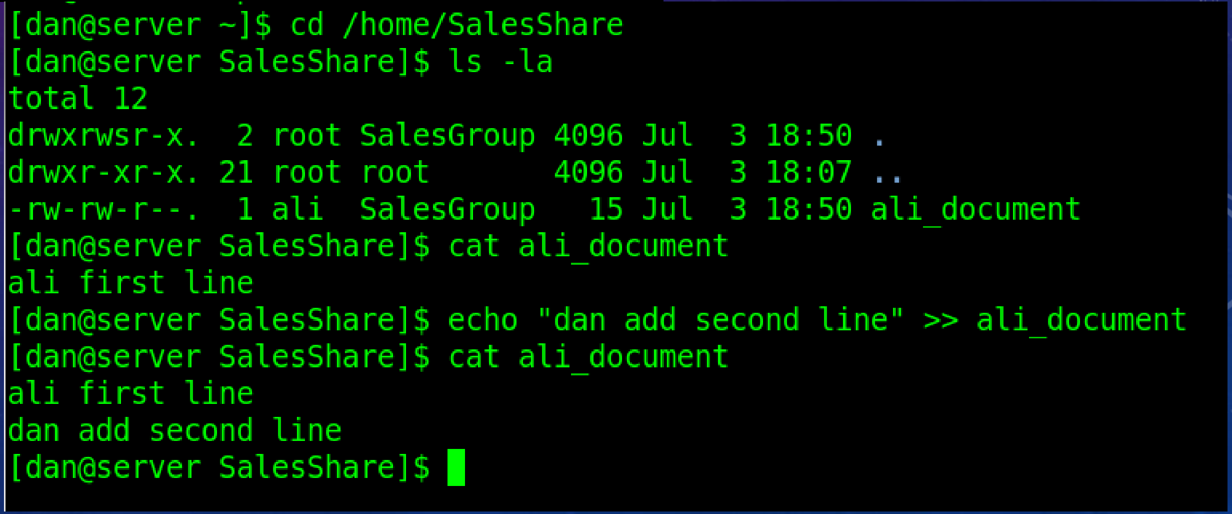
```
[root@server ~]# cat /etc/group | grep Sales*
SalesGroup:x:671:
[root@server ~]# gpasswd -M ali,eve,dan SalesGroup
[root@server ~]# cat /etc/group | grep Sales*
SalesGroup:x:671:ali,eve,dan
```

Conclusion: users ali, eve, and dan can create documents in the /home/SalesShare folder and edit each other files too.



```
Last login: Fri Jul  3 18:47:33 2020 from 10.0.0.200
[ali@server ~]$ cd /home/SalesShare
[ali@server SalesShare]$ echo "ali first line" > ali_document
[ali@server SalesShare]$ ls -la
total 12
drwxrwsr-x.  2 root SalesGroup 4096 Jul  3 18:50 .
drwxr-xr-x. 21 root root       4096 Jul  3 18:07 ..
-rw-rw-r--.  1 ali  SalesGroup   15 Jul  3 18:50 ali_document
[ali@server SalesShare]$ cat ali_document
ali first line
[ali@server SalesShare]$ 
```

ali can create a file in the shared folder. Note the owner and group.



```
[dan@server ~]$ cd /home/SalesShare
[dan@server SalesShare]$ ls -la
total 12
drwxrwsr-x.  2 root SalesGroup 4096 Jul  3 18:50 .
drwxr-xr-x. 21 root root       4096 Jul  3 18:07 ..
-rw-rw-r--.  1 ali  SalesGroup   15 Jul  3 18:50 ali_document
[dan@server SalesShare]$ cat ali_document
ali first line
[dan@server SalesShare]$ echo "dan add second line" >> ali_document
[dan@server SalesShare]$ cat ali_document
ali first line
dan add second line
[dan@server SalesShare]$ 
```

dan can also access the shared folder and edit ali’s file.

The Quota System

- Quota allows you to specify limits on two aspects of disk storage:
  - the number of inodes (data structures that contain information about files in UNIX file systems). This enable control over number of files.
  - the number of disk blocks

- Quota is handled on a per user, per file system basis. If there is more than one file system which a user is expected to create files, then quota must be set for each file system separately

- Disk quota can be configure for user or group.

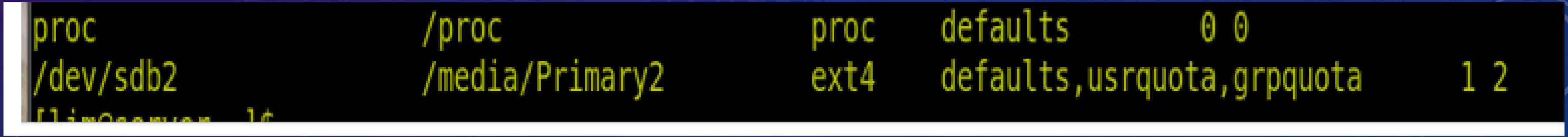
Disk Quota Steps

1. Modify the /etc/fstab file
2. Unmount and mount the file system
3. Run the quotacheck program
4. Run edquota to set up the user quotas

Configuring the Quota System

Step 1: Modify the /etc/fstab file

- For a partition to implement quotas, it must be mounted with the usrquota or grpquota options.
- Edit **/etc/fstab**, adding the **usrquota** option to the /media/Primary2 partition :



```
proc                    /proc                   proc    defaults        0 0
/dev/sdb2               /media/Primary2         ext4    defaults,usrquota,grpquota      1 2
```

Configuring the Quota System
Step 2: Remount the file system.

```
[root@station ~] # mount -o remount, rw /media/Primary2
```

However, if the file system is in use, you have to reboot the system

Configuring the Quota System
Step 3: Run the quotacheck program

- ❑ Create the quota databases.
- ❑ The disk usage database is stored within a partition's top-level directory in specially named binary files, aquota.user and aquota.group
- ❑ Use **quotacheck –cugfm /<directory>** to create a new user and group quota file
- ❑ Start and stop quotas with **quotaon** and **quotaoff** :

```
[root@stationX ~] # quotaon /<directory>

Example:
# quotaon /media/Primary2
```

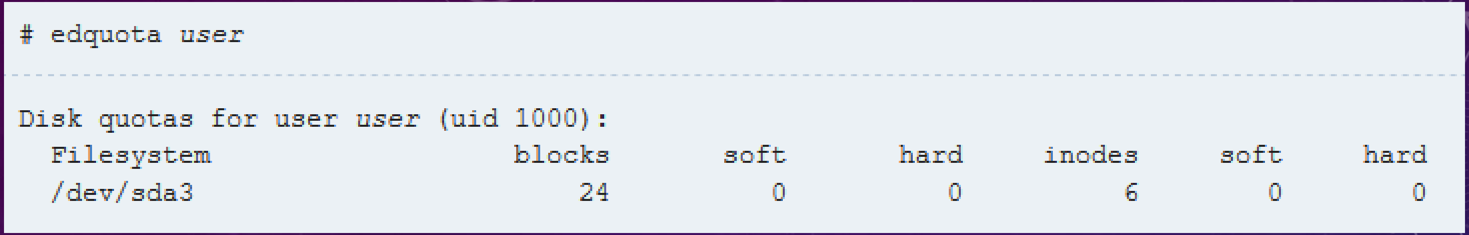
Configuring the Quota System
Step 4: Run edquota to set up the user quotas

- Edit quotas in nano editor: **edquota username**
- Edit from command:
  - **setquota** username BlkSoft BlkHard FileSoft FileHard filesys
  - Example

```
# setquota eddy 1024 2048 40 50 /media/Primary2
```

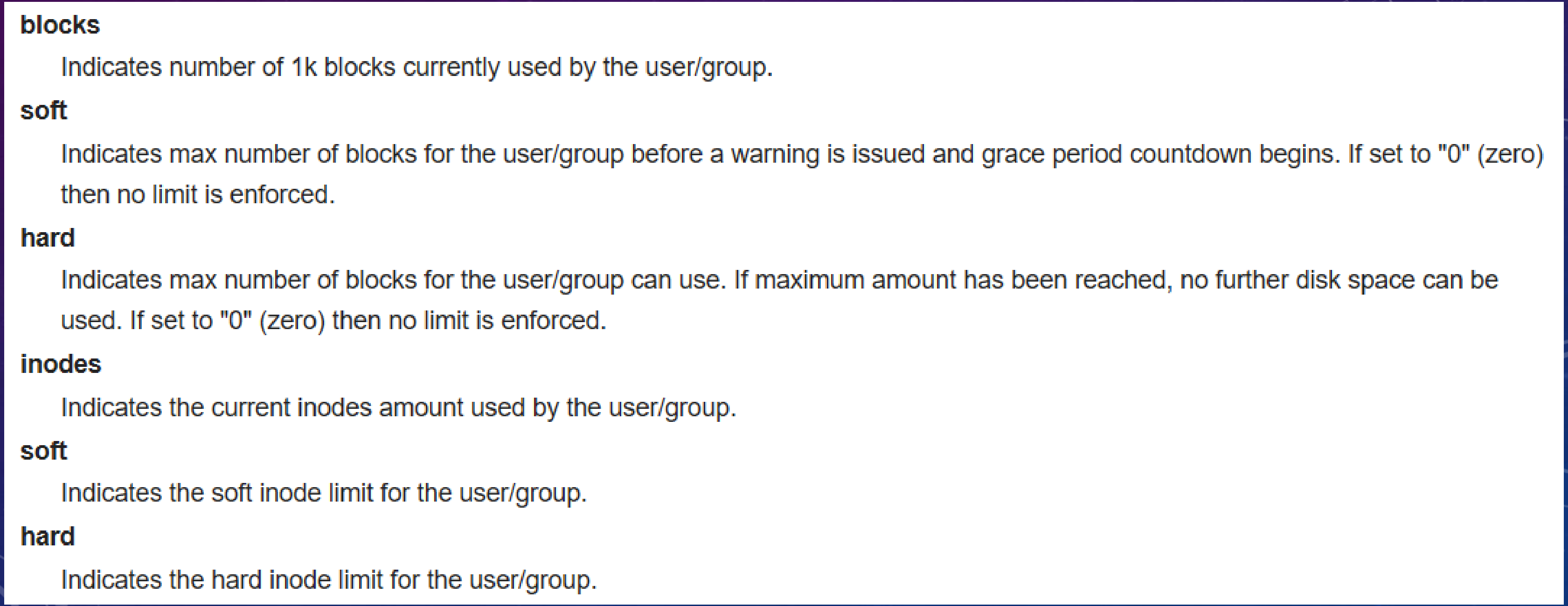
setquota *name block-softlimit block-hardlimit inode-softlimit inode-hardlimit filesystem…*

- Reporting
  - User inspection: **quota**
  - Quota overviews: **repquota -a**
  - Miscellaneous utilities: **warnquota**



```
# edquota user

Disk quotas for user user (uid 1000):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/sda3                        24          0          0          6        0        0
```



blocks

Indicates number of 1k blocks currently used by the user/group.

soft

Indicates max number of blocks for the user/group before a warning is issued and grace period countdown begins. If set to "0" (zero) then no limit is enforced.

hard

Indicates max number of blocks for the user/group can use. If maximum amount has been reached, no further disk space can be used. If set to "0" (zero) then no limit is enforced.

inodes

Indicates the current inodes amount used by the user/group.

soft

Indicates the soft inode limit for the user/group.

hard

Indicates the hard inode limit for the user/group.

Summary

- Linux is a multi-user system and uses permissions and ownerships for security
- There are 3 types of users – User, Group and Other
- There are 3 types of permissions – Read (r), Write (w), Execute (x)
- setuid, setgid & sticky bit – special file permissions
- Disk quota can be implemented to control users usage of disk storage.