

Linux Activity

Learning Outcomes:

Understand some key concepts of Linux operating system.

1. Users, Groups.
2. File permissions and authorizations.
3. Linux basic commands.

Linux Introduction: Linux is a multi-user operating system and have good working mechanism to provide security at the file system level.

To verify the linux operating system details:

`$lsb_release -a`

```
[manasm@ ~]$ lsb_release -a
LSB Version: :core-4.1-amd64:core-4.1-noarch:cxx-4.1-amd64:cxx-4.1-noarch:desktop-4.1-amd64:desktop-4.1-noarch:languages-4.1-amd64:languages-4.1-noarch:printing-4.1-amd64:printing-4.1-noarch
Distributor ID: CentOS
Description: CentOS Linux release 7.2.1511 (Core)
Release: 7.2.1511
Codename: Core
```

User and Group Permissions:

As any modern operating system Linux also has the facility of multi-tasking , another key feature of Linux is it also multi user, means more than one user have access to the system at the same time. With this feature in place it also has a mechanism to protect the contents of each user from one another i.e. user and group permissions.

Super User:

Root is the super user in a Linux system, which has access and privilege to do anything on the system.

To prevent any potential damage in place of root sudo is used, it will allow to a user to have administrative privilege without actually being a root user.

A sample of the sudo command is as follows:

`$sudo <command>`

Groups:

Groups are collections of zero or more users. A user belongs to a default group, and can also be a member of any of the other groups on a server.

An easy way to view all the groups and their members is to look in the `/etc/group` file on a server. We won't cover group management in this class, but you can run this command if you are curious about your groups:

```
$ cat /etc/group
```

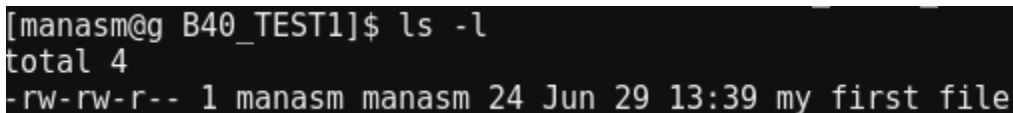
Ownership and Permissions:

The most common way to view the permissions of a file is to use `ls` with the long listing option, e.g. `ls -l myfile`. If you want to view the permissions of all of the files in your current directory, run the command without an argument, like this:

```
$ls -l
```

Hint: If you are in an empty home directory, and you haven't created any files to view yet, you can follow along by listing the contents of the `/etc` directory by running this command: `ls -l /etc`

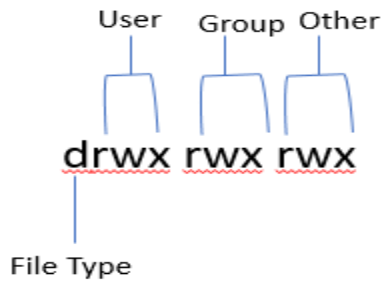
Here is an example screenshot of what the output might look like, with labels of each column of output:



```
[manasm@g B40_TEST1]$ ls -l
total 4
-rw-rw-r-- 1 manasm manasm 24 Jun 29 13:39 my_first_file
```

-- mode -- -Owner- -Group- -Last Modified- -file name-

We will try to understand what the different types of permissions are.



File Type:

In Linux files are of two types , normal and special, and they can be identified by the first character of the permission field.

If the first character is a hyphen “-” then that is a normal file, or regular file which can be used to save your contents or data.

Special files can be identified by a non-hyphen character in the first place.

Below we have listed some of the special files.

File types in a long list

Symbol	Meaning
-----	-----
-	Regular file
d	Directory
l	Link
c	Special file
s	Socket
p	Named pipe
b	Block device

File Permissions:

From the above diagram we know that the first field is the permission field.

Let us try to understand it in detail.

```
ls -l
-rw-rw-r-- 1 user group 4096 Jan 10 10:10 file1
drwxr-xr-x 2 user group 4096 Jan 10 10:10 dir1
```

So the first field if it is hyphen we know that it is a regular file and if it has some other character in place they are special file.

Then the next three places are permission for the **owner** of the file.

- - - :- These three triads are for read or (r), write or (w) and execute or (x).

So, from the above screen shot we know that the owner of the file has read and write permissions.

- - - :- Next three triads are for the **Group** permission, that means the group to which the owner of the file belongs to can have what type of access on this file.

- - - :- Then the next three triads are for the **Other**, for those users who is not the owner or belongs to the owner group. So, it will be safe to say all the other users in the system.

How Read, Write, Execute works:

Below we have mentioned how these three functionalities works.

Read

File: - Allows to view the contents of the file.

Directory: - To view the names of the files in the directory.

Write

File: - Change the contents of the file or delete the file.

Directory: - Delete the directory, modify the files inside it.

Execute

File: - User can run that file, basically if it is a executable file such as a python file or a shell script file.

Directory: - User can access the metadata of the files inside the directory.

Chmod Command:

Chmod command can be used to change the permission for the files and directories.

We can do that by giving octal representation for various access modes.

For example, we can give 4 for read, 2 for write and 1 for execute.

Below we have given some example.

This file has below permissions.

```
-rw-rw-r-- 1 manasm manasm 25 Jun 29 13:46 my_first_file
```

Now to give user the execute permission we can give the following command.

```
chmod 764 my_first_file
```

```
-rwxrw-r-- 1 manasm manasm 25 Jun 29 13:46 my_first_file
```

Linux Commands:

1. **pwd**: It will print the present working directory
directory]

Usage: pwd

2. To change the directory to the current user desktop: **cd**

Usage: cd <location>

cd ~ : Home directory

cd - : Previous directory to the current home directory.

cd .. : Parent directory of the current one.

cd / : System's root directory

When we give a / before to any path then the root of that is assumed as the start point else the current directory is assumed as the start point.

Try cd /home/<your-id>

Try cd home

Try cd /home

3. Create Directory: **mkdir**

Usage: mkdir <location of the directory>

Ex: mkdir test

`mkdir -m=rwx myfile` : Create directory with user defined access.

4. To list the files in a given directory: **ls**

Usage: `ls <location>`

`ls -a`: To show the hidden files

`ls -l`: long list format

`ls -lh`: file size in readable format

`ls -lhS`: files in descending order in terms of size

5. Create file using vi editor: **vi**

Usage: `vi <filename>` <Enter>

Ex: `vi testTxt.txt`

Esc i <This brings you to insert mode>

Type some text to be persistent on the file. Save and quit from vi prompt.

Esc :wq!

This brings you back to terminal

Type "ls" to see if the file is created.

6. view the content of the file: **cat**

Usage: `cat <filename>`

Ex: `cat testTxt.txt`

7. Print all the history of commands you have given: **history**

Usage: `history`

8. create a file if it does not exist, if exists only change the last access time stamp not the content: **touch**

Usage: `touch <filename>`

Ex: `touch testTxt.txt`

9. Copy file from one place to another: **cp**

Usage: `cp testTxt.txt copyFile.txt`

`ls` <Displays both testTxt.txt and copyFile.txt>

10. Move file from one location to another, source file will be removed, it can be used to rename the file also: **mv**

Usage: `mv <source> <destination>`

Ex: mv copyFile.txt newName.txt

11. Remove a file: **rm**

Usage: rm <filename>

12. Remove the directory and files recursively.

: **rm -r**

Usage: rm -r directory name

Ex: create a directory that can be deleted

mkdir testDel

cd testDel

touch file1.txt

cd ..

rm -r testDel

13. Clear terminal or screen: **clear** [Alternatively Ctrl + L can be used]

Usage: clear

14. find the file: **find**

Usage: find <location> -name "name of entity"

15. Number of newlines, words, and bytes in files: **wc**

Usage: wc <options> <filename>

Ex: wc -c testTxt.txt //print the byte counts

wc -l testTxt.txt //prints the number of lines

wc -w //print the number of words

16. Unique lines in a file: **uniq**

Usage: uniq [option] [input [output]]

Ex: Create a file, myFile.txt with the following content

This is a line.

This is a line.

This is a line.

This is also a line.

This is also a line.

This is also also a line

uniq myFile.txt

uniq -c myFile.txt //Count

uniq -d myFile.txt //Only print duplicated

17. To execute commands as a super user: **su** or **sudo**

Usage: su -user

18. Disk space utilization: **df -h**

Usage: df <options>

19. Name of the user who logged in: **who**

Usage: who

Note how does "who" defer in its output "who am i" or "whoami"

20. Last 10 lines of a file to console: **tail**

Usage: tail <Options> filename

Ex: tail myFile.txt

21. First 10 lines of a file to console: **head**

Usage: head <option> <file>

Ex: head -15 myFile.txt //Displays first 15 lines of output

22. Text search: **grep**

Usage: grep <pattern> <filename>

Ex: grep "test" myFile.txt

23. Manual for all commands available: **man /whatis**

Usage: man <command name>

Ex: man ls

24. File permissions: Understanding the user permissions on files
chmod

Example: -

-rwxrw-r-- 1 manasm manasm 25 Jun 29 16:34 my_first_file_copy

Now we will change the file permission to only read access.

chmod 400 my_first_file_copy

-r----- 1 manasm manasm 25 Jun 29 16:34 my_first_file_copy

Give full access to user and group (i.e read, write and execute)
on a specific file.

\$ chmod ug+rwx my_first_file_copy

Summary:

In this session, we learnt about:

Basic commands in Linux Operating System, and Key Concepts such as

Users and Groups

File Permissions and Authorizations