Lab 2 - Linux Basics (2)

Introduction

The purpose of this lab is to give you a deeper introduction to the Linux OS through different topics and commands.

Users

Users are accounts that can be used to login into a system. Each user is identified by a **unique identification number** or **UID** by the system. All the information of users in a system are stored in **/etc/passwd** file. The hashed passwords for users are stored in **/etc/shadow** file.

Users can be divided into two categories on the basis of the level of access:

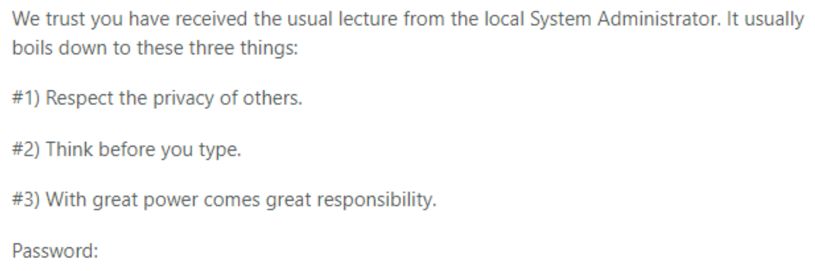
1. Superuser/root/administrator : Access to all the files on the system.
2. Normal users : Limited access.

What is the root user?

The root is the username or account that by default has access to all commands and files on a Linux or other Unix-like operating system. It is also referred to as the root account, root user, and the superuser.

What is the “sudo” Command?

Short for “**SuperUser Do**”, It lets non-privileged users access and modify files that require low-level permissions. Often you will use this command to access root from your regular user account.



```
We trust you have received the usual lecture from the local System Administrator. It usually
boils down to these three things:

#1) Respect the privacy of others.

#2) Think before you type.

#3) With great power comes great responsibility.

Password:
```

Everybody loves Uncle Ben, right?

Permissions and Privileges of the root user

Root privileges are the powers that the root account has on the system. The root account is the most privileged on the system and has absolute power over it (i.e., complete access to all files and commands). Among root's powers are the ability to modify the system in any way desired and to grant and revoke access permissions (i.e., the ability to read, modify and execute specific files and directories) for other users, including any of those that are by default reserved for root.

The permissions system in Unix-like operating systems is set by default to prevent access by ordinary users to critical parts of the system and to files and directories belonging to other users. This is because it is very easy to damage a Unix-like system with root access. However, an important principle of Unix-like operating systems is the provision of maximum flexibility to configure the system, and thus the root user is fully empowered.

When a new user is created, by default system takes following actions:

- Assigns UID to the user.
- Creates a home directory /home/.
- Sets the default shell of the user to be /bin/sh.
- Creates a private user group, named after the username itself.
- Contents of /etc/skel are copied to the home directory of the new user.
- .bashrc, .bash_profile and .bash_logout are copied to the home directory of new user.These files provide environment variables for this user's session.

Description of contents of /etc/passwd File

This file is readable by any user but only root as read and write permissions for it. This file consists of the following colon separated information about users in a system:

1. Username field
2. Password field
    - An `x` in this field denotes that the encrypted password is stored in the /etc/shadow file.
3. The user ID number (UID)
4. User's group ID number (GID)
5. Additional information field such as the full name of the user or comment (GECOS)
6. Absolute path of user's home directory
7. Login shell of the user

Syntax:
[username]:[password]:[UID]:[GID]:[GECOS]:[home_dir]:[shell_path]

Description of contents of the /etc/shadow File

This file is readable and writable by only by root user. This file consists of the following colon separated information about password of users in a system:

1. User name field
2. Password field
3. Contains an encrypted password.
    - A blank entry, {:: }, indicates that a password is not required to login into that user's account.
    - An asterisk, {:*:}, indicates the account has been disabled.
4. Last Password Change
    - This field denotes the number of days since the date of last password change counted since UNIX time (1-Jan-1970).
5. The minimum number of days after which the user can change his password.
6. Password validity
    - Denotes the number of days after which the password will expire.
7. Warning period
    - Denotes the number of days before the password expiry date, from which the user will start receiving warning notification for password change.
8. Account validity
    - Denotes the number of days after which the account will be disabled, once the password is expired.
9. Account disability
    - This field denotes the number of days since which the account had been disabled counted from UNIX time (1-Jan-1970).

Syntax:

```
[username]:[enc_pwd]:[last_pwd_change]:[pwd_validity]:[warn_date]:[acc_validity]:[acc_disablity]
```

How to add a User?

We can add a user using the command: useradd

Example:

```
sudo useradd -m username
```

hint: use man useradd to show available parameters to the command useradd.

Groups

A Linux group is a collection of users. The main purpose of the groups is to define a set of privileges like read, write, or execute permission for a given resource that can be shared among the users within the group. Users can be added to an existing group to utilize the privileges it grants.

There are two types of groups that a user can belong to:
Primary or login group – is the group that is assigned to the files that are created by the user. Usually, the name of the primary group is the same as the name of the user. Each user must belong to exactly one primary group.
Secondary or supplementary group - used to grant certain privileges to a set of users. A user can be a member of zero or more secondary groups.

How to find out which groups a user belongs to?

There are multiple ways to find out the groups a user belongs to.

The primary user's group is stored in the /etc/passwd file and the supplementary groups, if any, are listed in the /etc/group file.

To list the groups the current user belongs to, use the "groups" command

How to add a user to an existing group?

Using the command "usernmod"
```
sudo usermod -a -G groupname username
```

How to Remove a user from an existing group?

Using the command "gpasswd"
```
sudo gpasswd -d username groupname
```

How to Create a group?

```
sudo groupadd groupname
```

How to change a user's primary group?

```
sudo usermod -g groupname username
```

File Permissions

Every file in Unix has the following attributes:
Owner permissions − The owner's permissions determine what actions the owner of the file can perform on the file.
Group permissions − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
Other (world) permissions − The permissions for others indicate what action all other users can perform on the file.

The Permission Indicators

While using **ls -l** command, it displays various information related to file permission as follows

```
$ls -l /home/ayoussry
-rwxr-xr--  1 ayoussry   users 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 ayoussry   users 1024  Nov 2 00:10  mydir
```

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory. The first character indicates whether this entry is for a file '-' or a directory 'd'.

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) −

- The first three characters (2-4) represent the permissions for the file's owner. For example, **-rwxr-xr--** represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rwxr-xr--** represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rwxr-xr--** represents that there is **read (r)** only permission.

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 have been described below

Read
Grants the capability to read, i.e., view the contents of the file.

Write
Grants the capability to modify, or remove the content of the file.

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

Read

Access to a directory means that the user can read the contents. The user can look at the **filenames** inside the directory.

Write

Access means that the user can add or delete files from the directory.

Execute

Executing a directory doesn't really make sense, so think of this as a traverse permission. A user must have **execute** access to the **bin** directory in order to execute the **ls** or the **cd** command.

Changing Permissions

To change the file or the directory permissions, you use the **chmod** (change mode) command. There are two ways to use chmod — the symbolic mode and the absolute mode.

Using chmod in Symbolic Mode

The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.

+	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).

Here's an example using **testfile**. Running **ls -l** on the testfile shows that the file's permissions are as follows

```
$ls -l testfile
-rwxrwxr--  1 ayoussry   users 1024  Nov 2 00:10  testfile
```

Then each example **chmod** command from the preceding table is run on the testfile, followed by **ls –l**, so you can see the permission changes

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx  1 ayoussry   users 1024  Nov 2 00:10  testfile
$chmod u-x testfile
$ls -l testfile
-rw-rwxrwx  1 ayoussry   users 1024  Nov 2 00:10  testfile
$chmod g = rx testfile
$ls -l testfile
-rw-r-xrwx  1 ayoussry   users 1024  Nov 2 00:10  testfile
```

Here's how you can combine these commands on a single line

```
$chmod o+wx,u-x,g = rx testfile
$ls -l testfile
-rw-r-xrwx  1 ayoussry   users 1024  Nov 2 00:10  testfile
```

Changing Owners and Groups

While creating an account on Unix, it assigns a **owner ID** and a **group ID** to each user. All the permissions mentioned above are also assigned based on the Owner and the Groups.
Two commands are available to change the owner and the group of files −

- **chown** − The **chown** command stands for **"change owner"** and is used to change the owner of a file.
- **chgrp** − The **chgrp** command stands for **"change group"** and is used to change the group of a file.

Changing Ownership

The chown command changes the ownership of a file. The basic syntax is as follows

```
$ chown user filelist
```

The value of the user can be either the **name of a user** on the system or the **user id (uid)** of a user on the system.

The following example will help you understand the concept

```
$ chown ayoussry testfile
```

Changes the owner of the given file to the user **ayoussry**.
NOTE − The super user, root, has the unrestricted capability to change the ownership of any file but normal users can change the ownership of only those files that they own.

Changing Group Ownership

The **chgrp** command changes the group ownership of a file. The basic syntax is as follows

```
$ chgrp group filelist
```

The value of group can be the **name of a group** on the system or **the group ID (GID)** of a group on the system.
Following example helps you understand the concept

```
$ chgrp special testfile
```

Changes the group of the given file to **special** group.

More Commands

ls ls -R will list all the files in the sub-directories as well ls -a will show the hidden files ls -al will list the files and directories with detailed information like the permissions, size, owner, etc.	This command will display the contents of your current working directory.
locate	You can use this command to locate a file, just like the search command in Windows.
find	Similar to the locate command, using find also searches for files and directories. The difference is, you use the find command to locate files within a given directory.
grep	This command lets you search through all the text in a given file. ***grep blue notepad.txt*** will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully.
head	The head command is used to view the first lines of any text file. By default, it will show the first ten lines
tail	The tail command is used to view the last lines of any text file. By default, it will show the last ten lines
diff	Short for difference, the diff command compares the contents of two files line by line. After analyzing the files, it will output the lines that do not match. ***diff file1.ext file2.ext***
sort	Whenever you find the need to sort out a file in an alphabetical or reverse manner, utilize this command.
history	The history command lets you check the history of your terminal sessions, it will print out the bash history of your terminal session
alias	it lets them replace a word by another string in files directly from the terminal.