Experiment 12

Aim: Managing Users: Types: Super, Owner, Group, Others, Adding user, removing user, working with Passwords, expiry dates using usermod.

Linux Commands to manage Local Accounts

User and Group Management

Since Linux is a multi-user operating system, several people may be logged in and actively working on a given machine at the same time. Security-wise, it is never a good idea to allow users to share the credentials of the same account. In fact, best practices dictate the use of as many user accounts as people needing access to the machine.

At the same time, it is to be expected that two or more users may need to share access to certain system resources, such as directories and files. User and group management in Linux allows us to accomplish both objectives.

There are three types of accounts on a Linux system −

Root account

This is also called **superuser** and would have complete and unfettered control of the system. A superuser can run any commands without any restriction. This user should be assumed as a system administrator.

System accounts

System accounts are those needed for the operation of system-specific components for example mail accounts and the **sshd** accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.

User accounts

User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Linux supports a concept of *Group Account* which logically groups a number of accounts. Every account would be a part of another group account. A Linux group plays important role in handling file permissions and process management.

Managing Users and Groups

There are four main user administration files −

- **/etc/passwd** − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- **/etc/shadow** − Holds the encrypted password of the corresponding account. Not all the systems support this file.
- **/etc/group** − This file contains the group information for each account.

- **/etc/gshadow** − This file contains secure group account information.
- **/etc/sudoers** -(configuration for sudo).

Check all the above files using the **cat** command.

What Is the "/etc/passwd" File?

The first file we will look at, called the "/etc/passwd" file, does not actually store passwords.

At one time, this file stored the hashed passwords of every user on the system. However, this responsibility has been moved to a separate file for security reasons.

Let's look at what *is* in the "/etc/passwd" file:

cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/bin/sh

man:x:6:12:man:/var/cache/man:/bin/sh

. . .

The first thing to note is that this file is accessible by unprivileged users.

Everyone on the system has read privileges to this file. This is why password information was moved out of this file.

Let's look at the format of the file.

How To Read the "/etc/passwd" File

Each line in the file contains the login information of a single user on the system. Some of these users might be created for use by daemons and background services.

Take a look at a single line to see what information it contains:

```
root:x:0:0:root:/root:/bin/bash
```

The fields of information are separated by a colon (:) character. There are seven fields on each line in a typical Linux "/etc/passwd" file:

1. **root:** Account username.
2. **x:** Placeholder for password information. The password is obtained from the "/etc/shadow" file.
3. **0:** User ID. Each user has a unique ID that identifies them on the system. The root user is always referenced by user ID 0.
4. **0:** Group ID. Each group has a unique group ID. Each user has a "primary" group that is used as the group by default. Again, the root group's ID is always 0.
5. **root:** Comment field. This field can be used to describe the user or user's function. This can be anything from contact information for the user, to descriptions of the service the account was made for.
6. **/root:** Home directory. For regular users, this would usually be "/home/username". For root, this is "/root".
7. **/bin/bash:** User shell. This field contains the shell that will be spawned or the command that will be run when the user logs in.

As you add user accounts using commands like "adduser" and "useradd", or as you install more services, this file will grow. New username information will be added to the bottom of this file.

In most cases, you should not have to edit this file by hand. There are tools that manipulate this file and ensure that the proper syntax is maintained.

What Is the "/etc/shadow" File?

The actual password data is stored in a file called "/etc/shadow".

This doesn't actually contain passwords in plain text. Instead, it uses a key derivation function to create a hash. This is what it stores in the file.

A key derivation function is basically an algorithm that will always create a certain hash when given the same input. The same algorithm is run on the password that is given during authentication and this value is compared to the value in this file.

Note that this file, unlike the "/etc/passwd" file, is not readable by unprivileged users.

The root user has read and write permissions, and the "shadow" group, which contains users needed for authentication, has read permissions.

How To Read the "/etc/shadow" File

Open the "/etc/shadow" file by typing:

sudo cat /etc/shadow

root:6mJD3Rsj4$xUa7jru6EEGTXnhwTfTT26/j8M5XiQvUl6UH32cfAWT/6W9iSI5IuIw5OOw4khwrsOHPyMwfCLyayfYiVdhAq0:15952:0:99999:7:::

daemon:*:15455:0:99999:7:::

bin:*:15455:0:99999:7:::

sys:*:15455:0:99999:7:::

sync:*:15455:0:99999:7:::

games:*:15455:0:99999:7:::

man:*:15455:0:99999:7:::

. . .

Like the "/etc/passwd" file, each line defines a user's information and each field is delimited by a colon (:) character.

Note: The asterisk (*) value in the second field on some of the above lines means that the account cannot log in. This is mainly used for services and is intended behavior.

Let's take a look at a single line again:

daemon:*:15455:0:99999:7:::

These are fields defined in the "/etc/shadow" file:

1. daemon: Account username.
2. *: Salt and hashed password. You can see what this looks like with the root entry above. As noted above, the asterisk signifies that this account cannot be used to log in.

3. 15455: Last password change. This value is measured in days from the Unix "epoch", which is January 1, 1970.
4. 0: Days until password change permitted. 0 in this field means there are no restrictions.
5. 99999: Days until password change required. 99999 means that there is no limit to how long the current password is valid.
6. 7: Days of warning prior to expiration. If there is a password change requirement, this will warn the user to change their password this many days in advance.
7. [blank]The last three fields are used to denote days before the account is made inactive, days since the Epoch when the account expires. The last field is unused.

User administration is one of the important tasks of Linux system administrator. Local accounts or users in Linux like operating system are managed by useradd, usermod, userdel, chage passwd, groupadd, groupmod and groupdel commands.

- **useradd** command is used to create new accounts in Linux
- **usermod** command used to modify the existing accounts in linux
- **userdel** command is used to delete local account in linux
- **passwd** command used assign password to local accounts or users.
- **chage** comamnd is used to view & modify users password expiry information
- **groupadd** Adds groups to the system
- **groupmod** Modifies group attributes
- **groupdel** Removes groups from the system

Add an Account

Following is the example that creates an account ***donald***, setting its home directory to ***/home/donald*** and the group as ***developers***. This user would have Korn Shell assigned to it.

```
$ useradd -d /home/donald -g developers -s /bin/ksh donald
```

Before issuing the above command, make sure you already have the *developers* group created using the ***groupadd*** command.

Syntax of 'useradd' command

```
# useradd <options> <username_or_login>
```

Options used in useradd command:

```
-b, --base-dir BASE_DIR       base directory for the home directory of the
                              new account
-c, --comment COMMENT         GECOS field of the new account
-d, --home-dir HOME_DIR       home directory of the new account
-D, --defaults                print or change default useradd configuration
-e, --expiredate EXPIRE_DATE  expiration date of the new account
-f, --inactive INACTIVE       password inactivity period of the new account
-g, --gid GROUP               name or ID of the primary group of the new
                              account
-G, --groups GROUPS           list of supplementary groups of the new
                              account
-h, --help                    display this help message and exit
-k, --skel SKEL_DIR           use this alternative skeleton directory
-K, --key KEY=VALUE           override /etc/login.defs defaults
-l, --no-log-init             do not add the user to the lastlog and
                              faillog databases
-m, --create-home             create the user's home directory
-M, --no-create-home          do not create the user's home directory
-N, --no-user-group           do not create a group with the same name as
                              the user
-o, --non-unique              allow to create users with duplicate
                              (non-unique) UID
-p, --password PASSWORD       encrypted password of the new account
-r, --system                  create a system account
-R, --root CHROOT_DIR         directory to chroot into
-s, --shell SHELL             login shell of the new account
-u, --uid UID                 user ID of the new account
-U, --user-group              create a group with the same name as the user
-Z, --selinux-user SEUSER     use a specific SEUSER for the SELinux user mapping
```

If you do not specify any parameter, then the system makes use of the default values. The **useradd** command modifies the **/etc/passwd**, **/etc/shadow**, and **/etc/group** files and creates a home directory.

2. Create a User with Different Home Directory

By default ‘useradd‘ command creates a user’s home directory under /home directory with username. Thus, for example, we’ve seen above the default home directory for the user ‘tecmint‘ is ‘/home/tecmint‘.

However, this action can be changed by using ‘-d‘ option along with the location of new home directory (i.e. /data/projects). For example, the following command will create a user ‘anusha‘ with a home directory ‘/data/projects‘.

```
[root@tecmint ~]# useradd -d /data/projects anusha
```

You can see the user home directory and other user related information like user id, group id, shell and comments.

```
[root@tecmint ~]# cat /etc/passwd | grep anusha
anusha:x:505:505::/data/projects:/bin/bash
```

3. Create a User with Specific User ID

In Linux, every user has its own UID (Unique Identification Number). By default, whenever we create a new user accounts in Linux, it assigns userid 500, 501, 502 and so on…

But, we can create user's with custom userid with '-u' option. For example, the following command will create a user 'navin' with custom userid '999'.

```
[root@tecmint ~]# useradd -u 999 navin
```

Now, let's verify that the user created with a defined userid (999) using following command.

```
[root@tecmint ~]# cat /etc/passwd | grep navin

navin:x:999:999::/home/navin:/bin/bash
```

NOTE: Make sure the value of a user ID must be unique from any other already created users on the system.

4. Create a User with Specific Group ID

Similarly, every user has its own GID (Group Identification Number). We can create users with specific group ID's as well with -g option.

Here in this example, we will add a user 'tarunika' with a specific UID and GID simultaneously with the help of '-u' and '-g' options.

```
[root@tecmint ~]# useradd -u 1000 -g 500 tarunika
```

Now, see the assigned user id and group id in '/etc/passwd' file.

```
[root@tecmint ~]# cat /etc/passwd | grep tarunika

tarunika:x:1000:500::/home/tarunika:/bin/bash
```

5. Add a User to Multiple Groups

The '-G' option is used to add a user to additional groups. Each group name is separated by a comma, with no intervening spaces.

Here in this example, we are adding a user 'tecmint' into multiple groups like admins, webadmin and developer.

```
[root@tecmint ~]# useradd -G admins,webadmin,developers tecmint
```

Next, verify that the multiple groups assigned to the user with id command.

```
[root@tecmint ~]# id tecmint
```

```
uid=1001(tecmint) gid=1001(tecmint)

groups=1001(tecmint),500(admins),501(webadmin),502(developers)

context=root:system_r:unconfined_t:SystemLow-SystemHigh
```

6. Add a User without Home Directory

In some situations, where we don't want to assign a home directories for a user's, due to some security reasons. In such situation, when a user logs into a system that has just restarted, its home directory will be root. When such user uses su command, its login directory will be the previous user home directory.

To create user's without their home directories, '-M' is used. For example, the following command will create a user 'shilpi' without a home directory.

```
[root@tecmint ~]# useradd -M shilpi
```

Now, let's verify that the user is created without home directory, using ls command.

```
[root@tecmint ~]# ls -l /home/shilpi

ls: cannot access /home/shilpi: No such file or directory
```

7. Create a User with Account Expiry Date

By default, when we add user's with 'useradd' command user account never get expires i.e their expiry date is set to 0 (means never expired).

However, we can set the expiry date using '-e' option, that sets date in YYYY-MM-DD format. This is helpful for creating temporary accounts for a specific period of time.

Here in this example, we create a user 'aparna' with account expiry date i.e. 27th April 2014 in YYYY-MM-DD format.

```
[root@tecmint ~]# useradd -e 2014-03-27 aparna
```

Next, verify the age of account and password with 'chage' command for user 'aparna' after setting account expiry date.

```
[root@tecmint ~]# chage -l aparna
```

Last password change : Mar 28, 2014

Password expires : never

Password inactive : never

Account expires : Mar 27, 2014

Minimum number of days between password change : 0

Maximum number of days between password change : 99999

Number of days of warning before password expires : 7

8. Create a User with Password Expiry Date

The '-f' argument is used to define the number of days after a password expires. A value of 0 inactive the user account as soon as the password has expired. By default, the password expiry value set to -1 means never expire.

Here in this example, we will set a account password expiry date i.e. 45 days on a user 'tecmint' using '-e' and '-f' options.

```
[root@tecmint ~]# useradd -e 2014-04-27 -f 45 tecmint
```

9. Add a User with Custom Comments

The '-c' option allows you to add custom comments, such as user's full name, phone number, etc to /etc/passwd file. The comment can be added as a single line without any spaces.

For example, the following command will add a user 'mansi' and would insert that user's full name, Manis Khurana, into the comment field.

```
[root@tecmint ~]# useradd -c "Manis Khurana" mansi
```

You can see your comments in '/etc/passwd' file in comments section.

```
[root@tecmint ~]# tail -1 /etc/passwd

mansi:x:1006:1008:Manis Khurana:/home/mansi:/bin/sh
```

10. Change User Login Shell:

Sometimes, we add users which have nothing to do with login shell or sometimes we require assigning different shells to our users. We can assign different login shells to a each user with '-s'option.

Here in this example, will add a user 'tecmint' without login shell i.e. '/sbin/nologin' shell.

```
[root@tecmint ~]# useradd -s /sbin/nologin tecmint
```

You can check assigned shell to the user in '/etc/passwd' file.

```
[root@tecmint ~]# tail -1 /etc/passwd

tecmint:x:1002:1002::/home/tecmint:/sbin/nologin
```

11. Add a User with Specific Home Directory, Default Shell and Custom Comment

The following command will create a user 'ravi' with home directory '/var/www/tecmint', default shell /bin/bash and adds extra information about user.

```
[root@tecmint ~]# useradd -m -d /var/www/ravi -s /bin/bash -c "TecMint Owner" -U ravi
```

In the above command '-m -d' option creates a user with specified home directory and the '-s' option set the user's default shell i.e. /bin/bash. The '-c' option adds the extra information about user and '-U' argument create/adds a group with the same name as the user.

12. Add a User with Home Directory, Custom Shell, Custom Comment and UID/GID

The command is very similar to above, but here we defining shell as '/bin/zsh' and custom UID and GID to a user 'tarunika'. Where '-u' defines new user's UID (i.e. 1000) and whereas '-g' defines GID (i.e. 1000).

```
[root@tecmint ~]# useradd -m -d /var/www/tarunika -s /bin/zsh -c "TecMint Technical Writer" -u 1000 -g 1000 tarunika
```

13. Add a User with Home Directory, No Shell, Custom Comment and User ID

The following command is very much similar to above two commands, the only difference is here, that we disabling login shell to a user called 'avishek' with custom User ID (i.e. 1019).

Here '-s' option adds the default shell /bin/bash, but in this case we set login to '/usr/sbin/nologin'. That means user 'avishek' will not able to login into the system.

```
[root@tecmint ~]# useradd -m -d /var/www/avishek -s /usr/sbin/nologin -c "TecMint Sr. Technical Writer" -u 1019 avishek
```

14. Add a User with Home Directory, Shell, Custom Skell/Comment and User ID

The only change in this command is, we used '-k' option to set custom skeleton directory i.e. /etc/custom.skell, not the default one /etc/skel. We also used '-s' option to define different shell i.e. /bin/tcsh to user 'navin'.

```
[root@tecmint ~]# useradd -m -d /var/www/navin -k /etc/custom.skell -s /bin/tcsh -c "No Active Member of TecMint" -u 1027 navin
```

15. Add a User without Home Directory, No Shell, No Group and Custom Comment

This following command is very different than the other commands explained above. Here we used '-M' option to create user without user's home directory and '-N' argument is used that tells the system to only create username (without group). The '-r' arguments is for creating a system user.

```
[root@tecmint ~]# useradd -M -N -r -s /bin/false -c "Disabled TecMint Member" clayton
```

Modify an Account

The **usermod** command enables you to make changes to an existing account from the command line. It uses the same arguments as the **useradd** command, plus the -l argument, which allows you to change the account name.

For example, to change the account name ***donald*** to ***donald20*** and to change home directory accordingly, you will need to issue the following command −

```
$ usermod -d /home/donald20 -m -l mcmohd donald20
```

Once an account is created you can set its password using the **passwd** command as follows −

```
$ passwd donald20
```

```
Changing password for user donald20.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

When you type ***passwd accountname***, it gives you an option to change the password, provided you are a superuser. Otherwise, you can change just your password using the same command but without specifying your account name.

Syntax of usermod command:

```
# usermod <options> <username_or_login>
```

Options used in usermod command

```
-c, --comment COMMENT          new value of the GECOS field
-d, --home HOME_DIR            new home directory for the user account
-e, --expiredate EXPIRE_DATE   set account expiration date to EXPIRE_DATE
-f, --inactive INACTIVE        set password inactive after expiration
                               to INACTIVE
-g, --gid GROUP                force use GROUP as new primary group
-G, --groups GROUPS            new list of supplementary GROUPS
-a, --append                   append the user to the supplemental GROUPS
                               mentioned by the -G option without removing
                               him/her from other groups
-h, --help                     display this help message and exit
-l, --login NEW_LOGIN          new value of the login name
-L, --lock                     lock the user account
-m, --move-home                move contents of the home directory to the
                               new location (use only with -d)
-o, --non-unique               allow using duplicate (non-unique) UID
-p, --password PASSWORD        use encrypted password for the new password
-R, --root CHROOT_DIR          directory to chroot into
-s, --shell SHELL              new login shell for the user account
-u, --uid UID                  new UID for the user account
-U, --unlock                   unlock the user account
-Z, --selinux-user SEUSER      new SELinux user mapping for the user account
```

1. Adding Information to User Account

The '-c' option is used to set a brief comment (information) about the user account. For example, let's add information on 'tecmint' user, using the following command.

```
# usermod -c "This is Tecmint" tecmint
```

After adding information on user, the same comment can be viewed in /etc/passwd file.

```
# grep -E --color 'tecmint' /etc/passwd
tecmint:x:500:500:This is Tecmint:/home/tecmint:/bin/sh
```

2. Change User Home Directory

In the above step we can see that our home directory is under /home/tecmint/, If we need to change it to some other directory we can change it using -d option with usermod command.
For example, I want to change our home directory to /var/www/, but before changing, let's check the current home directory of a user, using the following command.

```
# grep -E --color '/home/tecmint' /etc/passwd
tecmint:x:500:500:This is Tecmint:/home/tecmint:/bin/sh
```

Now, change home directory from /home/tecmint to /var/www/ and confirm the home director after changing.

```
# usermod -d /var/www/ tecmint
# grep -E --color '/var/www/' /etc/passwd
tecmint:x:500:500:This is Tecmint:/var/www:/bin/sh
```

3. Set User Account Expiry Date

The option '-e' is used to set expiry date on a user account with the date format YYYY-MM-DD. Before, setting up an expiry date on a user, let's first check the current account expiry status using the 'chage'(change user password expiry information) command.

```
# chage -l tecmint

Last password change                                    : Nov 02, 2014
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 01, 2014
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

The expiry status of a 'tecmint' user is Dec 1 2014, let's change it to Nov 1 2014 using 'usermod -e' option and confirm the expiry date with 'chage' command.

```
# usermod -e 2014-11-01 tecmint
# chage -l tecmint

Last password change                                    : Nov 02, 2014
Password expires                                        : never
```

```
Password inactive                                     : never
Account expires                                       : Nov 01, 2014
Minimum number of days between password change        : 0
Maximum number of days between password change        : 99999
Number of days of warning before password expires     : 7
```

4. Change User Primary Group

To set or change a user primary group, we use option '-g' with usermod command. Before, changing user primary group, first make sure to check the current group for the user tecmint_test.

```
# id tecmint_test
uid=501(tecmint_test) gid=502(tecmint_test) groups=502(tecmint_test)
```

Now, set the babin group as a primary group to user tecmint_test and confirm the changes.

```
# usermod -g babin tecmint_test
# id tecmint_test
uid=501(tecmint_test) gid=502(babin) groups=502(tecmint_test)
```

5. Adding Group to an Existing User

If you want to add a new group called 'tecmint_test0' to 'tecmint' user, you can use option '-G' with usermod command as shown below.

```
# usermod -G tecmint_test0 tecmint
# id tecmint
```

Note: Be careful, while adding a new groups to an existing user with '-G' option alone, will remove all existing groups that user belongs. So, always add the '-a' (append) with '-G' option to add or append new groups.

6. Adding Supplementary and Primary Group to User

If you need to add a user to any one of the supplementary group, you can use the options '-a' and '-G'. For example, here we going to add a user account tecmint_test0 with the wheel user.

```
# usermod -a -G wheel tecmint_test0
# id tecmint_test0
```

So, user tecmint_test0 remains in its primary group and also in secondary group (wheel). This will make my normal user account to execute any root privileged commands in Linux box.

```
eg : sudo service httpd restart
```

7. Change User Login Name

To change any existing user login name, we can use '-l' (new login) option. In the example below, we changing login name tecmint to tecmint_admin. So the username tecmint has been renamed with the new name tecmint_admin.

```
# usermod -l tecmint_admin tecmint
```

Now check for the tecmint user, It will not be present because we have changed it to tecmint_admin.

```
# id tecmint
```

Check for the tecmint_admin account it will be there with same UID and with existing group what we have added before.

```
# id tecmint_admin
```

8. Lock User Account

To Lock any system user account, we can use '-L' (lock) option, After the account is locked we can't login by using the password and you will see a ! added before the encrypted password in /etc/shadow file, means password disabled.

```
# usermod -L babin
# grep -E --color 'babin' cat /etc/shadow
```

9. Unlock User Account

The '-U' option is used to unlock any locked user, this will remove the ! before the encrypted password.

```
# grep -E --color 'babin' /etc/shadow
```

```
# usermod -U babin
```

Verify the user after unlock.

```
# grep -E --color 'babin' /etc/shadow
```

10. Move User Home Directory to New location

Let's say you've a user account as 'pinky' with home directory '/home/pinky', you want to move to new location say '/var/pinky'. You can use the options '-d' and '-m' to move the existing user files from current home directory to a new home directory.

Check for the account and its current home directory.

```
# grep -E --color 'pinky' /etc/passwd
```

Then list the files which is owned by user pinky.

```
# ls -l /home/pinky/
```

Now we have to move the home directory from /home/pinky to /var/pinky.

```
# usermod -d /var/pinky/ -m pinky
```

Next, verify the directory change.

```
# grep -E --color 'pinky' /etc/passwd
```

Check for the files under '/home/pinky'. Here we have moved the files using -m option so there will be no files. The pinky user files will be now under /var/pinky.

```
# ls -l /home/pinky/
# ls -l /var/pinky/
```

11. Create Un-encrypted Password for User

To create an un-encrypted password, we use option '-p' (password). For demonstration purpose, I'm setting a new password say 'redhat' on a user pinky.

```
# usermod -p redhat pinky
```

After setting password, now check the shadow file to see whether its in encrypted format or un-encrypted.

```
# grep -E --color 'pinky' /etc/shadow
```

Note: Did you see in the above image, the password is clearly visible to everyone. So, this option is not recommended to use, because the password will be visible to all users.

12. Change User Shell

The user login shell can be changed or defined during user creation with useradd command or changed with ‘usermod‘ command using option ‘-s‘ (shell). For example, the user ‘babin‘ has the /bin/bash shell by default, now I want to change it to /bin/sh.

```
# grep -E --color 'babin' /etc/passwd
# usermod -s /bin/sh babin
```

After changing user shell, verify the user shell using the following command.

```
# grep -E --color 'babin' /etc/passwd
```

13. Change User ID (UID)

In the example below, you can see that my user account ‘babin‘ holds the UID of 502, now I want to change it to 888 as my UID. We can assign UID between 0 to 999.

```
# grep -E --color 'babin' /etc/passwd
OR
# id babin
```

Now, let’s change the UID for user babin using ‘-u‘ (uid) option and verify the changes.

```
# usermod -u 888 babin
# id babin
```

14. Modifying User Account with Multiple Options

Here we have a user jack and now I want to modify his home directory, shell, expiry date, label, UID and group at once using one single command with all options as we discussed above.
The user Jack has the default home directory /home/jack, Now I want to change it to /var/www/html and assign his shell as bash, set expiry date as December 10th 2014, add new label as This is jack, change UID to 555 and he will be member of apple group.

Let we see how to modify the jack account using multiple option now.

```
# usermod -d /var/www/html/ -s /bin/bash -e 2014-12-10 -c "This is Jack" -u 555 -aG apple jack
```

Then check for the UID & home directory changes.

```
# grep -E --color 'jack' /etc/passwd
```

Account expire check.

```
# chage -l jack
```

Check for the group which all jack have been member.

```
# grep -E --color 'jack' /etc/group
```

15. Change UID and GID of a User

We can change UID and GID of a current user. For changing to a New GID we need an existing group. Here already there is an account named as orange with GID of 777.

Now my jack user account wants to be assigned with UID of 666 and GID of Orange (777).

Check for the current UID and GID before modifying.

```
# id jack
```

Modify the UID and GID.

```
# usermod -u 666 -g 777 jack
```

Check for the changes

```
# id jack
```

Delete an Account

The **userdel** command can be used to delete an existing user. This is a very dangerous command if not used with caution. There is only one argument or option available for the command **.r**, for removing the account's home directory and mail file.

For example, to remove account *donald20*, issue the following command −

```
$ userdel -r donald20
```

If you want to keep the home directory for backup purposes, omit the **-r** option. You can remove the home directory as needed at a later time.

Syntax of userdel command.

```
# userdel <options> <username_or_login>
```

Options used in userdel command:

```
-f, --force                   force some actions that would fail otherwise
                              e.g. removal of user still logged in
                              or files, even if not owned by the user
-h, --help                    display this help message and exit
-r, --remove                  remove home directory and mail spool
-R, --root CHROOT_DIR         directory to chroot into
-Z, --selinux-user            remove any SELinux user mapping for the user
```

Syntax of chage:

```
# chage <options> <username_or_login>
```

Options used in chage command:

```
-d, --lastday LAST_DAY        set date of last password change to LAST_DAY
-E, --expiredate EXPIRE_DATE  set account expiration date to EXPIRE_DATE
-h, --help                    display this help message and exit
-I, --inactive INACTIVE       set password inactive after expiration
                              to INACTIVE
-l, --list                    show account aging information
-m, --mindays MIN_DAYS        set minimum number of days before password
                              change to MIN_DAYS
-M, --maxdays MAX_DAYS        set maximim number of days before password
                              change to MAX_DAYS
-R, --root CHROOT_DIR         directory to chroot into
-W, --warndays WARN_DAYS      set expiration warning days to WARN_DAYS
```

Syntax of passwd Command:

```
# passwd <username_or_login>
```

How Do You Change Passwords?

Users' passwords can be modified by issuing the "passwd" command.

By default, this command changes the current user's password and does not require special permissions.

```
passwd
```

If you would like to change another user's password, you will need administrative privileges. The following syntax can be used:

```
sudo passwd username
```

You will be prompted for your password for the "sudo" command, and then you will be told to enter and confirm the new password you would like to use.

If you compare the hashed value in the "/etc/shadow" file, you will see that it changes after you issue the passwd command.

Linux Change Group Password

When the -g option is used, the password for the named group is changed. In this example, change password for group sales:

```
# passwd -g sales
```

The current group password is not prompted for. The -r option is used with the -g option to remove the current password from the named group. This allows group access to all members. The -R option is used with the -g option to restrict the named group for all users.

Example 1 Create a local account & assign password.

```
# useradd <username> ; echo -e "<newpassword>\n<newpassword>" | passwd username
```

Let's create a username 'harry' and assign password.

```
[root@linuxtechi ~]# useradd harry ; echo -e "Roxicant@123#\nRoxicant@123#" | passwd harry
Changing password for user harry.
New password: Retype new password: passwd: all authentication tokens updated successfully.
```

```
[root@linuxtechi ~]#
```

Note: When a user is created in Linux followings are updated:

- A home directory is created under '/home/<username>'
- User info is updated in '/etc/passwd' file
- Group Information is stored in '/etc/group'
- Password info is updated in '/etc/shadow' file.
- File for user's email is created under '/var/spool/mail/<username>'

Example 2 Create a user with customize settings

Let's create a user with following options:

UID = 2000

GID = 5000

Comments = 'Admin Account of SAP'

Home Directory = /opt/sap

Shell = /bin/ksh

Username = john

Password = xxxxxx

```
[root@linuxtechi ~]# useradd -u 2000 -g 5000 -c "Admin Account of SAP" -d /opt/sap -s /bin/ksh john

[root@linuxtechi ~]#

[root@linuxtechi ~]# echo -e "Sapcant@123#\nSapcant@123#" | passwd john

Changing password for user john.

New password: Retype new password: passwd: all authentication tokens updated successfully.

[root@linuxtechi ~]#
```

Verify the above settings from /etc/passwd file.

```
[root@linuxtechi ~]# grep john /etc/passwd

john:x:2000:5000:Admin Account of SAP:/opt/sap:/bin/ksh
```

```
[root@linuxtechi ~]#
```

Example 3 Modify the Existing User

Let's make the existing user "harry" part of Secondary group "sap" and change its home directory from '/home/harry' to '/opt/sap' and login shell from '/bin/bash' to '/bin/sh'.

```
[root@linuxtechi ~]# usermod -G sap -d /opt/sap -s /bin/sh harry
[root@linuxtechi ~]#
[root@linuxtechi ~]# grep harry /etc/passwd
harry:x:1000:1000::/opt/sap:/bin/sh
[root@linuxtechi ~]#
```

Example 4 Create a user and force to change the password at first login.

Let's create a user 'mark' with secondary group 'sap', home directory as '/opt/sap' and force him to change his password at the first login.

We can force users to change its password at first login by using command 'chage -d 0 <username>'.

```
[root@linuxtechi ~]# useradd -c "sap user" -G sap -d /opt/data mark
[root@linuxtechi ~]# echo -e "Sapdata@123#\nSapdata@123#" | passwd mark ; chage -d 0 mark
Changing password for user mark.
New password: Retype new password: passwd: all authentication tokens updated successfully.
[root@linuxtechi ~]#
```

Now try to login as mark and see whether user is getting a prompt to change password or not.

Note : Use 'chage -l <username>' command to view the user's password expiry info.

Example 5 Delete a User along with its home directory

userdel command is used to delete local accounts or users in Linux. Let's delete a user linuxtechi along with its related its files (home directory)

```
[root@linuxtechi ~]# userdel -r linuxtechi
[root@linuxtechi ~]# grep linuxtechi /etc/passwd
[root@linuxtechi ~]#
```

Create a Group

We will now understand how to create a group. For this, we need to create groups before creating any account otherwise; we can make use of the existing groups in our system. We have all the groups listed in ***/etc/groups*** file.

All the default groups are system account specific groups and it is not recommended to use them for ordinary accounts. So, following is the syntax to create a new group account −

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

The following table lists out the parameters −

Sr.No.	Option & Description
1	**-g GID** The numerical value of the group's ID
2	**-o** This option permits to add group with non-unique GID
3	**-r** This flag instructs **groupadd** to add a system account
4	**-f** This option causes to just exit with success status, if the specified group already exists. With -g, if the specified GID already exists, other (unique) GID is chosen
5	**groupname** Actual group name to be created

If you do not specify any parameter, then the system makes use of the default values.

Following example creates a *developers* group with default values, which is very much acceptable for most of the administrators.

```
$ groupadd developers
```

Modify a Group

1. To modify a group, use the **groupmod** syntax –

```
$ groupmod -n new_modified_group_name old_group_name
```

2. To change the developers_2 group name to developer, type –

```
$ groupmod -n developer developer_2
```

3. Here is how you will change the financial GID to 545 −

```
$ groupmod -g 545 developer
```

Examples :

1. Change the group "newgroup" to "bettergroup".

```
# groupmod -n bettergroup newgroup
```

2. Change groupid of a group

This command changes the groupid of abc group to 777

```
# groupmod -g 777 abc
```

3. When -o is used with -g option we can give non unique values

```
# groupmod -g 777 -o abc1
```

Delete a Group

We will now understand how to delete a group. To delete an existing group, all you need is the **groupdel command** and the **group name**. To delete the financial group, the command is −

```
$ groupdel developer
```

This removes only the group, not the files associated with that group. The files are still accessible by their owners.

A Note on Superuser Permissions

- **sudo:** run one or more commands as another user (typically with superuser permissions).

- **/etc/sudoers** (configuration for sudo).
- Superuser permissions can be gained either by changing to the root user with the **su** command or using **sudo**. The latter approach is used by default in Ubuntu and derivatives, and is preferred over the former in other distributions as well.

It is also important to note that, as opposed to other Linux flavors, the user that is created when Ubuntu is first installed has superuser privileges out-of-the-box. You can verify whether sudo is installed on your machine by running

```
which sudo
```

on a terminal. If this command returns the absolute path of the associated file (typically **/usr/bin/sudo**), it means that the package is installed. Otherwise, you can install it with

```
apt-get install sudo
```

on Debian, or

```
yum install sudo
```

in CentOS or similar.

Adding a New Regular Account

To begin, let's create a new user named **pluralsight** using Ubuntu and CentOS as representative distributions.
In Ubuntu or derivatives, this is as easy as doing (you will be required to enter your password to run sudo):

```
sudo adduser pluralsight
```

In other distributions, first login as root and do:

```
adduser pluralsight
```

You may be prompted to set the new user's initial password, and other optional information (such as full name, work phone, etc). This will be stored in **/etc/passwd** using colons as field separators. If not, you can assign a password for the newly created account named **pluralsight** with

```
passwd pluralsight
```

and entering it twice. Needless to say, you must preface the above command with sudo if you're using Ubuntu.
When a new user is added, a group with the same name is created automatically. This is called a primary group.

The /etc/sudoers File

Now that we have a regular user account created, we will explain how to utilize it to perform user management tasks.
To grant **plural sight** superuser permissions, we will need to add an entry for it in **/etc/sudoers**. This file is used to indicate which users can run what commands with elevated permissions (most likely as root).

Step 1 - Open /etc/sudoers with visudo

Although **/etc/sudoers** is nothing more and nothing less than a plain text file, it must NOT be edited using a regular text editor. Instead, we will use the `visudo` command. As opposed to other text editors, by utilizing `visudo` we will ensure that 1) no one else can modify the file at the same time, and 2) the file syntax is checked upon saving changes.
To launch `visudo`, just type the command and press Enter. Don't forget to do

```
sudo visudo
```

instead if you're in Ubuntu. In any event, the file will be opened using your default text editor.

Step 2 - Add an Entry in /etc/sudoers for the New User Account

The easiest method to grant superuser permissions for **pluralsight** is by adding the following line at the bottom of **/etc/sudoers**:

```
pluralsight ALL=(ALL) ALL
```

Let's explain the syntax of this line:

- First off, we indicate which user this rule refers to (**pluralsight**).
- The first **ALL** means the rule applies to all hosts using the same **/etc/sudoers** file. Nowadays, this means the current host since the same file is not shared across other machines.
- Next, **(ALL) ALL** tells us that **pluralsight** will be allowed to run all commands as any user. Functionally speaking, this is equivalent to **(root) ALL**.

Step 3 (Optional): Create Command Aliases

An alternative to using the wide permissions outlined above, we can restrict the list of commands that can be executed by a given user by grouping them into sets known as *aliases*.
For example, we may want to allow user **pluralsight** to only use `adduser` and `usermod`, but not other commands. To do so, we can either list the commands one by one (using the corresponding absolute paths) at the end of the same entry:

```
pluralsight    ALL=(root) /usr/sbin/adduser, /usr/sbin/usermod
```

or define an *alias* (which we can name as we wish as long as it's all upper case, for example **USERMANAGEMENT**):

```
Cmnd_Alias USERMANAGEMENT = /usr/sbin/adduser, /usr/sbin/usermod
pluralsight    ALL=(root) USERMANAGEMENT
```

While the latter requires two lines, it is often preferred instead of the former because it contributes to keep **/etc/sudoers** cleaner. In any event, **pluralsight** will not be able to execute any other commands as root other than those specified above.

For more information on the available options in /etc/sudoers, refer to man sudoers.
While saving, visudo will alert you if a syntax error is found in the file, and indicate the line where the error is found so that you can identify it more easily.

Switching Users

If no errors are found while saving the recent changes in **/etc/sudoers**, we'll be ready to start using **pluralsight** to perform user management tasks. To do so, use the `su` command to change to that account. Note that from this point, there is no need to use the root account if you're in CentOS or similar.
Additionally, the `-l` option will allow providing an environment like what the user would expect if he or she had logged in directly:

```
su -l pluralsight
```