Linux Administration

Add New user

- useradd command
- It edits /etc/passwd, /etc/shadow, /etc/group and /etc/gshadow files for the newly created User account.
- Creates and populate a home directory for the new user.
- Sets permissions and ownerships to home directory.
- Basic syntax of command is:

```
useradd [options] username
```

Add New user

#useradd user1

- When we add a new user in Linux with ‘useradd‘ command it gets created in locked state and to unlock that user account, we need to set a password for that account with ‘passwd‘ command.

#passwd user1

Changing password for user user1.

New UNIX password:

Retype new UNIX password:

passwd: all authentication tokens updated successfully.

- Once a new user created, entry will get added in ‘/etc/passwd‘ file.

/etc/passwd

- The entry in /etc/passwd contains a set of fields, delimiter is :

- Username: User login name. It should be between 1 to 32 characters long.
- Password: User password (or x character) stored in /etc/shadow file in encrypted format.
- User ID (UID): Every user must have a User ID (UID) User Identification Number. By default UID 0 is reserved for root user and UID’s ranging from 1-99 are reserved for other predefined accounts. Further UID’s ranging from 100-999 are reserved for system accounts and groups.
- Group ID (GID): The primary Group ID (GID) Group Identification Number stored in /etc/group file.
- User Info: It is optional and allow to define extra information about the user. For example, user full name. This field is filled by ‘finger’ command.
- Home Directory: The absolute location of user’s home directory.
- Shell: user’s shell i.e. /bin/bash.

Options

- -d ----- to specify home folder name
- -u -------for assigning specific user id
- -g -------to assign specific group id
- -G -------option is used to add a user to additional groups. Each group name is separated by a comma
- -e --------create a user with expiry date
- User password will expire after 45 days from data of creation

useradd -e 2001-07-27 -f 45 user1

Add New user

- 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.
- example, the following command will create a user ‘navin‘ with custom userid ‘999‘.
- useradd -u 999 rajan

Add New user

To see user information

#id user1

Groups in Linux

- There are two kinds of groups:
- Primary Group:
  - This is the group applied to user when user log in
  - In most user cases it has the same name as user login name.
- The primary group is used by default when creating new files (or directories), modifying files, or executing commands.
- Secondary Groups :
  - These are groups you are a member of beyond your primary group.
- A list of all currently available groups can be found in the /etc/group file.

Creating groups

- Create or Delete a Group in Linux: groupadd and groupdel
- Using the groupadd command, we can create a new group: group1.

#groupadd group1

#adduser myuser group1 -------adduser is symbolic link to useradd command

- To remove group1 from the Linux system

#groupdel group1

- To delete the user's home directory along with the user account itself, type this command as root:

#userdel -r username

Add *a existing user* to existing group

- **To add *a existing user* to existing group using**
- **Use usermod command using the following options**
- **-a ------- add the user**
- **-G -------- to the secondary group(s).**

- **Add existing user user1 to project1 secondary group**

 # usermod -a -G project1 user1

- **To change user1 user's primary group to project2**

 # usermod -g project2 user1

Delete user

- To delete user
- #userdel username

Change group or ownership of file

- ***Chown or chgrp***: change individual or group ownership of a file or directory

#chgrp [OPTION]... GROUP FILE...

- To change existing group of file1 to root

#chgrp root file1

How chgrp deals with symbolic links

- By default, the chgrp command affects the original file of a symbolic link.
- This means that any change made with chgrp doesn't get applied to the symbolic link, but to the file it's referring to instead.
- For example, consider the symbolic link 'file1symlink' which links to 'file1’.
- Both 'file1' and 'file1symlink' have their owner and group set to ‘user1'.

- Now, if you try changing the group of the symbolic link file, you'll see that 'file1symlink' will remain unaffected, but group ownership of 'file1' gets changed instead.

#chgrp root file1symlink

- However, if you want, you can change this behaviour by using the -h command line option.

- chgrp -h root file1symlink

- This command changes the group of the symbolic link
- chgrp result

How to make chgrp checks group information from a reference file?

- If you want, you can ask the chgrp command to pick group information from a file, instead of manually specifying group name on the command line.
- This feature can be accessed through the **--reference** command line option, which requires you to specify the name of the reference file.
- *chgrp --reference=[ref-file-name] FILENAME*
- For example:
- *chgrp --reference=file1 file2*
- The above command will make the group ownership of file1 same as that of file2.

- How to ask chgrp to make changes recursively
- In case - while dealing with directories and subdirectories
- you want to make recursive changes
- use the -R option.

- chgrp -R GROUPNAME DIRECTORY-OR-PATH

chown command

- chown command changes the user and/or group ownership of given file. The syntax is:

- chown owner-user file
- chown owner-user:owner-group file
- chown owner-user:owner-group directory
- chown options owner-user:owner-group file

- change file ownership to user1 user and list the permissions

chown user1 demo.txt

ls -l demo.txt

- The owner is set to user1 followed by a colon and a group onwership is also set to group1 group

chown user1:group1 demo.txt

ls -l demo.txt

- To change only the group of file.
- the colon and following group-name group1 are given, but the owner is omitted, only the group of the files is changed:

chown :group2 demo.txt

- # ls -l demo.txt

-rw-r--r-- 1 user1 group2 0 Aug 2 05:50 demo.txt

- Please note that if only a colon is given, or if new owner is empty, neither the owner nor the group is changed:
- # chown : demo.txt

- To change the owner of /mydir to “root”
- # chown root /mydir

- change owner to root and group to “gr1”

chown root:gr1 /mydir

- Change the owner of /mydir and subfiles to “root”
- # chown -R root /mydir

- Where,

- -R – Recursively change ownership of directories and their contents.

- use chgrp when chown can also be used to change groups
- One could argue that if the chown command can also be used to tweak group-related information, then why chgrp is required in the first place? Well, firstly, chgrp is simple to use compared to chown when all you need to do is to change the group of a file/directory. And secondly, since chown is also capable of doing other stuff, one would not want to accidentally make changes while working on, say, a production server.