## 1. Concept: User and Group Management in Linux:

The concept of Groups and Users is pretty straightforward. Everything (or better say, every process) in Linux runs under specific user and uses that user's permissions for its proper execution. To further extend the permissions of a group (or collection) of users, the User Group concept was introduced. We know that each file, should be owned by a User. Now, another user may or may not be able to read/edit/execute that file, depending on that file's permissions and the group of the user.

In simpler words, if we want to run a process, then it has to run under some user. Any user should be a part of a group or a set of groups. For example, when first install Linux and create the primary user, then we give a username, which becomes the User's login. A group with same name as the username is created and is assigned as the primary group of the user. The user is also assigned to other groups depending on what the user is supposed to do.

My user name is swashata and my primary group is "swashata". Other than that, I might be added to the following groups as well.

```
swashata adm cdrom sudo dip plugdev lpadmin sambashare
```

Therefore, I can also apply sudo command, have administrator rights, can use sambashare and so on.

What a group can do, solely depends on the model of an application. Most of the system applications like, Apache, SambaShare etc creates groups and allows user only their own group to execute them.

## 2. Standard Linux Files

Everything in Linux is stored in a file, Groups and Users are no exceptions. We can view the following file to quickly view the current status of users and groups.

/etc/group File – Group Information

```
[Group name]:[Group password]:[GID]:[Group members]
```

- **[Group name]** is the name of group.
- An x in **[Group password]** indicates group passwords are not being used.

- **[GID]**: same as in /etc/passwd.
- **[Group members]**: a comma separated list of users who are members of **[Group name]**.

/etc/passwd File – User Information:

It holds 7 fields delimited by colon(:).

```
[username]:[x]:[UID]:[GID]:[Comment]:[Home directory]:[Default shell]
```

- Fields **[username]** and **[Comment]** are self-explanatory.
- The x in the second field indicates that the account is protected by a shadowed password (in /etc/shadow), which is needed to logon as [username].
- The **[UID]** and **[GID]** fields are integers that represent the User IDentification and the primary Group IDentification to which **[username]** belongs, respectively.
- The **[Home directory]** indicates the absolute path to [username]'s home directory, and
- The **[Default shell]** is the shell that will be made available to this user when he or she logins the system.

/etc/shadow File – User Login Information:

The shadow file holds the password of the user and other login credentials. It has 8 columns delimited by colon(:) which holds the following information.

1. **User name:** It is your login name.
2. **Password:** It your encrypted password. The password should be minimum 6-8 characters long including special characters/digits
3. **Last password change (lastchanged):** Days since Jan 1, 1970 that password was last changed.
4. **Minimum:** The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password.
5. **Maximum:** The maximum number of days the password is valid (after that user is forced to change his/her password).
6. **Warn:** The number of days before password is to expire that user is warned that his/her password must be changed.
7. **Inactive:** The number of days after password expires that account is disabled.
8. **Expire:** Days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used.

## 3. Adding a new Group

Now, let us see how we can actually create and manage users and groups. Obviously, we shall use terminal commands to do the necessary. Also, these concepts are for System Administrators (SA). So, we will need su privilege to execute any of the commands. For general Linux system, we can start a root session by typing su on the terminal, whereas in debian or Ubuntu, we can either type sudo then the command or can start a root session by entering **sudo su**.

To add a normal group named mygroup we would execute the command:

```
groupadd mygroup
```

Now, if we do a

```
cat /etc/group
```

Then the output will be something like this:

```
utempter:x:121:
rtkit:x:122:
saned:x:123:
swashata:x:1000:
sambashare:x:124:swashata
winbindd_priv:x:125:
gdm:x:126:
mygroup:x:1001:
```

Where we can see our group. Note that the group ID 1001 is automatically assigned to the group. There are a few useful parameters as well.

| Parameter | Usage | Example |
|---|---|---|
| -g | Used to define the group ID of the group we are creating. | `groupadd -g 2000 mygroupgid` |
| -p | Specifies a password for the group.<br>You must use the **openssl passwd -crypt** command to first encrypt the password | `openssl passwd -crypt pass`<br>`groupadd -p encrypted_pass` |

## 4. Adding a new User

To create a user named “myuser” we shall use the following command:

```
useradd -c "My User" -d "/home/myuser" -s "/bin/bash" -m myuser
```

Now let us see what are the most useful parameters for the command and also what the parameters above did.

| Parameter | Usage | Example |
|---|---|---|
| -d | The path of the home directory of the user. If the path does not exist, then it is not necessarily created by default | `useradd -d "/home/myuser" myuser` |
| -m | Creates the user’s home directory if it is not present. Also, copies everything from skeleton if it is specified. | `useradd -m -c "My User" myuser`<br><br>`The simplest way to create users with all default settings.` |

| Parameter | Usage | Example |
|---|---|---|
| -g | Primary group ID or name. If not specified, a new group is created with same name as the login name of the user and the corresponding ID is assigned. | `useradd -g 100 myuser`<br><br>The GID 100 corresponds to a group named "users". myuser will be assigned to that group. |
| -G | List of supplementary groups which the user is also a member of. We can have multiple groups separated by comma. Without -a replaces the existing secondary groups. | `useradd -g 100 -G adm,sudo myuser`<br><br>Will make the user myuser also a member of sudo and adm group and use corresponding privileges. |
| -s | Specifies the default login shell of the user. | `useradd -d "/home/myuser" -s "/bin/bash" myuser`<br><br>The login shell will be bash. |
| -c | To add comments. It is mainly used as the Name of the user. | `useradd -c "My User" myuser`<br><br>"My User" will come as the name on the login window. |
| -u | Assign an ID value manually to the user. Has to be non-negative integer and unique. We do not however, use this option for system users. So, typically the range starts from value greater than 999. | `useradd -u 2000 -m myuser` |

## 5. Assigning Passwords to Users

We have created an account for a new user, assigned primary group and supplementary groups, etc. But, in order to login to the account, we will need to specify the password as well.

```
passwd user
```

We can also specify an encrypted password when we create the user, by typing:

```
useradd -p "encrypted_pass" myuser

Alike the groups, we have to execute before "openssl
passwd -crypt pass" to get the "encrypted_pass"
```

## 6. Modifying existing Groups and Users

**Groups**

Now, that we have learnt about creating groups and users, we might want to modify them as well. Luckily, the commands for modification are very straightforward and accepts all the parameters from the groupadd or useradd commands. Let us take a quick look.
To change a group name from “mygroup” to “yourgroup” and to change the ID from “OLD_ID” to “NEW_ID”, we would simply use:

```
groupmod –g new_id –n yourgroup mygroup
```

The only new parameter introduced here is **-n**. It defines the new name. All of the other parameters of groupadd holds true.

**Users**

To change the login of “myuser” to “youruser” and name to “Your User” and ID to 3000 and append to the adm group we would use this:

```
usermod –l youruser –c "Your User" –u 3000 –G adm –a myuser
```

Two new introduced parameters are:

1. **-l**: Specify the new login name.
2. **-a:** If used with -G, then all the groups that are not in the current list of user’s supplementary groups will be appended to the existing ones. Otherwise, the user will be removed from the groups which is not listed (with -G).

## 7. Deleting existing Groups and Users

**Groups**

To delete a group “mygroup” we simply execute:

```
groupdel mygroup
```

Note that, if the group is a primary group of a user, then we need to delete the user first before deleting the group. If the group is a supplementary group of some users, then the group will be deleted safely (ie, it will also remove users from the group automatically).

**Users**

To simply delete an user “myuser” we would do:

```
userdel myuser
```

This will delete the user but will not remove its home directory and other files. In addition, user will not be deleted and a warning will be shown if s/he is currently logged in.

| Parameter | Usage | Example |
|---|---|---|
| -f | This option forces the removal of the user account, even if the user is still logged in. It also forces userdel to remove the user's home directory and mail spool, even if another user uses the same home directory or if the mail spool is not owned by the specified user. | `userdel -f myuser` |
| -r | All files and directories inside the user's home directory will be removed along with mail spool | `userdel -r myuser` |

## 8. Other useful commands

Although useradd and userdel will work for Ubuntu or other debian system, but it is recommended to use the following commands instead:

- For adding users/groups: **adduser**, **addgroup**
- For deleting users/group: **deluser**, **delgroup**

They are friendlier front ends to the low level tools like useradd, groupadd and usermod programs, by default choosing Debian policy conformant UID and GID values, creating a home directory with skeletal configuration, running a custom script, and other features.

For example, a basic run of `adduser myuser` will do:

1. Create the user named username.
2. Create the user's home directory (default is /home/username and copy the files from /etc/skel into it.
3. Create a group with the same name as the user and place the user in it.
4. Prompt for a password for the user.
5. Prompt for additional information on the user.

More commands:

- **groups:** You can find out which groups a user belongs to by typing: `groups username`
- **id:** Prints user and group information for the specified USERNAME, or (when USERNAME omitted) for the current user. Example: `id username`
- **who:** Print information about users who are currently logged in.
- **whoami:** Print information about users who are logged in the terminal you have currently opened.