

USER AND GROUP ADMINISTRATION

Some Important Points related to Users:

- ➔ Users and groups are used to control access to files and resources
- ➔ Users login to the system by supplying their username and password
- ➔ Every file on the system is owned by a user and associated with a group
- ➔ Every process has an owner and group affiliation, and can only access the resources its owner or group can access.
- ➔ Every user of the system is assigned a unique user ID number (the UID)
- ➔ Users name and UID are stored in **/etc/passwd**
- ➔ User's password is stored in **/etc/shadow** in encrypted form.
- ➔ Users are assigned a home directory and a program that is run when they login (Usually a shell)
- ➔ Users cannot read, write or execute each other's files without permission.


Types of users In Linux and their attributes:

TYPE	EXAMPLE	USER ID (UID)	GROUP ID (GID)	HOME DIRECTORY	SHELL
Super User	Root	0	0	/root	/bin/bash
System User	ftp, ssh, apache, nobody	1 to 499	1 to 499	/var/ftp , etc	/sbin/nologin
Normal User	Visitor, ktuser, etc	500 to 60000	500 to 60000	/home/user name	/bin/bash

Whenever a user is created in Linux things created by default:-

- ➔ A home directory is created(/home/username)
- ➔ A mail box is created(/var/spool/mail)
- ➔ unique UID & GID are given to user

/etc/passwd

 root@master-server:~

```
[root@master-server ~]# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
[root@master-server ~]#
```

The above fields are

root = name ; x = link to password file i.e. /etc/shadow; 0 or 1 = UID (user id); 0 or 1 = GID (group id)

root or bin = comment (brief information about the user); /root or /bin = home directory of the user

/bin/bash or /sbin/nologin = shell

➔ /etc/shadow

```
root@master-server:~  
[root@master-server ~]# tail -1 /etc/shadow  
dev:$6$EGQmuhM1l8n4J72W$cFdjCLQJnBMbulKXQWxprz1X3coxPuMkZFylagPjCMGZfZCw10nYrdLsoK..QLEZVfTcHR.Qd0CrLvDutsZo/:20205:0:999  
99:7:::  
[root@master-server ~]#
```

The fields are as follows,

1. root = User name ;
2. :\$1fdsfsgsdfsdskffefje = Encrypted password ;
3. 1475
4. 7 = Days since that password was last changed.;
5. 0 = Days after which password must be changed.;
6. 99999 = Days before password is to expire that user is warned.
7. A reserved field.

Let's create a user with default attributes:

#useradd <username>

#useradd yallareddy

```
root@master-server:~  
[root@master-server ~]# useradd yallareddy  
[root@master-server ~]# id yallareddy  
uid=501(yallareddy) gid=501(yallareddy) groups=501(yallareddy)  
[root@master-server ~]#
```

```
root@master-server:~  
[root@master-server ~]# tail -1 /etc/passwd  
yallareddy:x:501:501::/home/yallareddy:/bin/bash  
[root@master-server ~]#
```

Note: If you use command #useradd the above fields are automatically created.

Let's create a user with our own attributes:

- ➔ Create a user with following attributes
- ➔ Name = Yalla Reddy
- ➔ Uid=505
- ➔ Home Dir = /home/yallareddy
- ➔ Comment= Linux Administrator
- ➔ `#useradd yallareddy -u 505 -g 505 -d /home/yallareddy -c linux administrator`

Note: Before creating the check below commands

`#id <username>`

`#cat /etc/passwd | grep <username>`

```
root@master-server:~  
[root@master-server ~]# id yallareddy  
id: yallareddy: No such user  
[root@master-server ~]#  
[root@master-server ~]# cat /etc/passwd | grep -i yallareddy  
[root@master-server ~]#  
[root@master-server ~]#
```

So now user is not present so we can create a user by using below command

- ➔ `#useradd yallareddy -u 505 -g 505 -d /home/yallareddy -c linux administrator`

Before creating the user , create a group also by using below command

`#groupadd -g <gid> groupname`

```
root@master-server:~  
[root@master-server ~]# groupadd -g 505 linuxgroup  
[root@master-server ~]#
```

Now execute useradd command

- ➔ `#useradd yallareddy -u 505 -g 505 -d /home/yallareddy -c linux administrator`

root@master-server:~

```
[root@master-server ~]# useradd yallareddy
[root@master-server ~]# passwd yallareddy
Changing password for user yallareddy.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@master-server ~]#
```

Modifying the user's attribute:

- ➔ After creating a user if we need to modify the attributes of user like changing uid, changing secondary group id or adding a comment, locking or unlocking the user account, can be done by following command
- ➔ Syntax. # usermod <options> <username>
- ➔ all the options which are used with useradd command can be used and also the following,
- ➔ -l to change login name
- ➔ -L to LOCK account
- ➔ -U to UNLOCK account
- ➔ ex. # usermod -l newname oldname (changing the name of the user)
- ➔ ex. # usermod -L newname to lock the user account
- ➔ ex. # usermod -U newname to unlock the user account
- ➔ Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.

```
root@master-server:~
[root@master-server ~]# cat /etc/passwd | grep yallareddy
yallareddy:x:501:501::/home/yallareddy:/bin/bash
[root@master-server ~]# usermod -L yallareddy
[root@master-server ~]# cat /etc/passwd | grep yallareddy
yallareddy:x:501:501::/home/yallareddy:/bin/bash
[root@master-server ~]# cat /etc/shadow | grep yallareddy
yallareddy:!!$6$jNjVHmni$bGfO7WfdMm7RsIFSGArqNykvOB.1mrsLb.P.5bWs7LPMORlsXpUuKyV2yBKuwTH3GJOHQtbmpJeUVT0rHnkva.:20213:0:999
99:7:::
[root@master-server ~]#
```

Locking and unlocking a user account::


- ➔ #usermod -L < user name>

```
root@master-server:~
[root@master-server ~]# cat /etc/shadow | grep yallareddy
yallareddy:!!$6$jNjVHmni$bGfO7WfdMm7RsIFSGArqNykvOB.1mrsLb.P.5bWs7LPMORlsXpUuKyV2yBKuwTH3GJOHQtbmpJeUVT0rHnkva.:20213:0:999
99:7:::
[root@master-server ~]#
[root@master-server ~]# usermod -U yallareddy
[root@master-server ~]# cat /etc/shadow | grep yallareddy
yallareddy:$6$jNjVHmni$bGfO7WfdMm7RsIFSGArqNykvOB.1mrsLb.P.5bWs7LPMORlsXpUuKyV2yBKuwTH3GJOHQtbmpJeUVT0rHnkva.:20213:0:999
99:7:::
➔ [root@master-server ~]#
```

The password parameters:

- ➔ For any user we can set the parameters for the password, like min and max password age, password expiration warnings and a/c expiration date etc.

- ➔ `#chage -l <username>`

 root@master-server:~

```
[root@master-server ~]# chage -l yallareddy
Last password change           : May 05, 2025
Password expires               : never
Password inactive              : never
Account expires                : never
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
```

- ➔ [root@master-server ~]# █

- ➔ **Last password change:** When the password was change last time.

- ➔ **Password expires:** Password expiry date

- ➔ **Password inactive:** After password expiry grace period before the account gets locked

- ➔ **Account expires:** Date on which the account expires.

- ➔ **Minimum number of days b/w password change:** once the password is changed, it cannot be changed until a min period of specified date. [0] means never.

- ➔ **Max number of days b/w password change:** After changing the password how long it will be valid for.


- ➔ **Number of days of warning before password expires:** start of warnings to change the password, no. of days before the password expires.

Changing the password parameters:

`#chage <username>`

`#chage <option> <value><username>`

To set never expire password `#chage -M -1 <username>`

 root@master-server:~

```
[root@master-server ~]# chage -M -1 yallareddy
[root@master-server ~]# chage -l yallareddy
Last password change           : May 05, 2025
Password expires               : never
Password inactive              : never
Account expires                : never
Minimum number of days between password change : 0
Maximum number of days between password change : -1
Number of days of warning before password expires : 7
[root@master-server ~]# █
```

Deleting a User:

To delete a user the syntax used is `#userdel <username>`

`#userdel` it will only delete the user but home directory will be there. To delete the user with its home directory use the following command.

`#userdel -r < user name >`

PART-II GROUP ADMINISTRATION:

GROUPS:

- ➔ Users are assigned to groups with unique group ID numbers (the GID)
- ➔ The group name and GID are stored in `/etc/group`
- ➔ Each user is given their own private group
- ➔ They can also be added to their groups to gain additional access
- ➔ All users in a group can share files that belong to the group
- ➔ Each user is a member of at least one group, called a primary group. In addition, a user can be a member of an unlimited number of secondary groups. Group membership can be used to control the files that a user can read and edit. For example, if two users are working on the same project you might put them in the same group so they can edit a particular file that other users cannot access.
- ➔ A user's primary group is defined in the `/etc/passwd` file and Secondary groups are defined in the `/etc/group` file.
- ➔ The primary group is important because files created by this user will inherit that group affiliation.

Creating a Group with default options :

`#groupadd <groupname>`


Creating a group with user specified group id (GID)

`#groupadd <options> <name of the group>`

`#groupadd -g 500 linuxgroup`

Adding and Removing Members to a Group:

- ➔ Adding the members to the group is to add users to the group. To add the members to the group the syntaxes are
- ➔ To add single user to the group
- ➔ `#usermod -aG <groupname> < user name>`
- ➔ `#usermod -aG redhatgroup yallareddy`

 root@master-server:~

```
[root@master-server ~]# cat /etc/group | grep redhat
redhatgroup:x:506:
[root@master-server ~]# usermod -aG redhatgroup yallareddy
[root@master-server ~]# cat /etc/group | grep redhat
redhatgroup:x:506:yallareddy
[root@master-server ~]#
```

➔

CONTROLLING ACCESS TO FILES :

Special Permissions or Advanced Permission:

- ➔ There are three special permissions that can be assigned to a file or directory apart from basic file permissions(rwx), they are
- ➔ **SUID – SET USER ID**
- ➔ **SGID – SET GROUP ID**
- ➔ **STICKY BIT**

Permission	Symbolic Form	Numeric Form	Syntax
SETUID	s or S	4	#chmod u+s or #chmod 4766
SETGID	s or S	2	#chmod g+s or #chmod 2766
STICKY BIT	t or T	1	#chmod o+t or chmod 1766

→ **Note:** Where **s= setuid + execute** permission and **S= setuid only**. Same is for **SGID** and also for **sticky bit**.

SUID – SET USER ID:

Change user ID on execution. If SETUID bit is set, when the file will be executed by a user, the process will have the same rights as the owner of the file being executed. Many of the system commands are the best example for SUID, basically the owner of the commands will be root, but still a normal user can execute it.

Example:

→ By default ping command is having suid, so all users can run that command but if suid is removed and a normal user wants to user execute it, then it will show '**operation not permitted**'

root@master-server:~

```
[root@master-server ~]# which ping
/bin/ping
[root@master-server ~]# ls -l /bin/ping
-rwsr-xr-x. 1 root root 40760 Mar 22 2011 /bin/ping
[root@master-server ~]# ping google.com
PING google.com (142.251.43.46) 56(84) bytes of data.
64 bytes from bkk02s01-in-f14.1e100.net (142.251.43.46): icmp_seq=1 ttl=128 time=18.7 ms
^C
--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 567ms
rtt min/avg/max/mdev = 18.745/18.745/18.745/0.000 ms
→ [root@master-server ~]#
```

Note: observe that in the permissions “-rwsr-xr-x” it contains an “s”, which means SUID is placed.

- Let's remove suid on Ping command and logged in as normal user and check the results
- **#chmod 755 /bin/ping**
- Suid value is 4
- Remove the suid and switch with normal user like below

yallareddy@master-server:~

```
[root@master-server ~]# ls -l /bin/ping
-rwsr-xr-x. 1 root root 40760 Mar 22 2011 /bin/ping
[root@master-server ~]#
[root@master-server ~]# chmod 755 /bin/ping
[root@master-server ~]# ls -l /bin/ping
-rwxr-xr-x. 1 root root 40760 Mar 22 2011 /bin/ping
[root@master-server ~]# su - yallareddy
[yallareddy@master-server ~]$ ping google.com
ping: icmp open socket: Operation not permitted
[yallareddy@master-server ~]$
```

SGID – SET GROUP ID:

Note: sgid value is 2

- Set group ID, used on executable files to allow the file to be run as if logged into the group (like SUID but uses file group permissions) SGID can also be used on a directory so that every file created in that directory will have the directory group owner rather than the group owner of the user creating the file.
- Example
- When a directory is created and its group is set to some group. Now if SGID is applied to it, and the group member creates files and directory inside it, then it will get the same group rather than getting user's primary group

STICKY BIT:

If sticky bit is applied on a file or directory, then only root and owner of that file or directory can delete it. Even if others are having full permissions they cannot delete the file or directory.

Note: By default /tmp is having the stickbit permissions

Stickybit value is 1.

root@master-server:~

```
[root@master-server ~]# ls -ld /tmp/
drwxrwxrwt. 17 root root 4096 May 5 22:22 /tmp/
[root@master-server ~]#
[root@master-server ~]#
```

dev@master-server:/tmp

```
[root@master-server ~]# su - yallareddy
[yallareddy@master-server ~]$ cd /tmp/
[yallareddy@master-server tmp]$ touch yalla-file1
[yallareddy@master-server tmp]$ ls -l yalla-file1
-rw-rw-r--. 1 yallareddy yallareddy 0 May  6 00:20 yalla-file1
[yallareddy@master-server tmp]$ pwd
/tmp
[yallareddy@master-server tmp]$ whoami
yallareddy
[yallareddy@master-server tmp]$ chmod 777 yalla-file1
[yallareddy@master-server tmp]$ whoami
yallareddy
[yallareddy@master-server tmp]$ ls -l yalla-file1
-rwxrwxrwx. 1 yallareddy yallareddy 0 May  6 00:20 yalla-file1
[yallareddy@master-server tmp]$ logout
```

Now switch to any other user try to delete a file. Other user cannot delete the file even though the file has full permissions.

```
[yallareddy@master-server tmp]$ logout
[root@master-server ~]# su - dev
[dev@master-server ~]$ cd /tmp/
[dev@master-server tmp]$ ls -l yalla-file1
-rwxrwxrwx. 1 yallareddy yallareddy 0 May  6 00:20 yalla-file1
[dev@master-server tmp]$ rm -rf yalla-file1
rm: cannot remove `yalla-file1': Operation not permitted
[dev@master-server tmp]$
```

Access Control List (ACL):

- ➔ Define more fine-grained discretionary access rights for files and directories.
- ➔ Often, you want to share files among certain groups and specific users. It is a good practice to designate a directory for that purpose. You want to allow those groups and users to read, and write files in that directory, as well as create new files into the directory. Such special permissions can be given using ACL.
- ➔ Now check the default permission and acl permission on /acl
- ➔ To check the acl permission syntax is
- ➔ `getfacl <options> <dir/filename>`
- ➔ `#getfacl /acl`

root@master-server:~

```
[root@master-server ~]# getfacl /acl
getfacl: Removing leading '/' from absolute path names
# file: acl
# owner: root
# group: root
user::rwx
group::r-x
other::r-x

[root@master-server ~]#
```

➔ Now let's assign full permission to the directory and then apply acl on it, so that we can analyze how acl will work.

➔ #chmod 777 /acl

root@master-server:~

```
[root@master-server ~]# ls -ld /acl/
drwxr-xr-x. 2 root root 4096 May  6 00:32 /acl/
[root@master-server ~]#
[root@master-server ~]# chmod 777 /acl/
[root@master-server ~]# ls -ld /acl/
drwxrwxrwx. 2 root root 4096 May  6 00:32 /acl/
[root@master-server ~]#
```

- ➔
- ➔ The syntax to apply acl is
- ➔ #setfacl <options> < argument > < file or directory name >
- ➔ The options are,
- ➔ -m Modifies an ACL
 - ➔ -x Removes an ACL
 - ➔ -R Recurses into subdirectories
- ➔ The possible arguments are
- ➔ u: user g: group • o: others

To assign read and execute permission to a particular user the syntax could be

#setfacl -m u: <username>:<permissions> <file/dir name>

#setfacl -m u:yallareddy:rwx /acl

root@master-server:~


```
[root@master-server ~]# getfacl /acl
getfacl: Removing leading '/' from absolute path names
# file: acl
# owner: root
# group: root
user::rwx
group::rwx
other::rwx

[root@master-server ~]#
[root@master-server ~]# setfacl -m u:yallareddy:rwx /acl
[root@master-server ~]#
[root@master-server ~]# getfacl /acl
getfacl: Removing leading '/' from absolute path names
# file: acl
# owner: root
# group: root
user::rwx
user:yallareddy:rwx
group::rwx
mask::rwx
other::rwx

[root@master-server ~]#
```

Removing acl for a particular user:

```
#setfacl -x u:<username><dirname>
#setfacl -x u:yallareddy /acl
```

 root@master-server:~


```
[root@master-server ~]# setfacl -x u:yallareddy /acl
[root@master-server ~]# getfacl /acl
getfacl: Removing leading '/' from absolute path names
# file: acl
# owner: root
# group: root
user::rwx
group::rwx
mask::rwx
other::rwx

[root@master-server ~]# █
```

Removing all ACL permissions from a file or directory

#setfacl -b <dirname>

#setfacl -b /acl

 root@master-server:~

```
[root@master-server ~]# setfacl -b /acl/
[root@master-server ~]# getfacl /acl/
getfacl: Removing leading '/' from absolute path names
# file: acl/
# owner: root
# group: root
user::rwx
group::rwx
other::rwx

[root@master-server ~]# █
```

#####END of User and group administration

