User and Groups

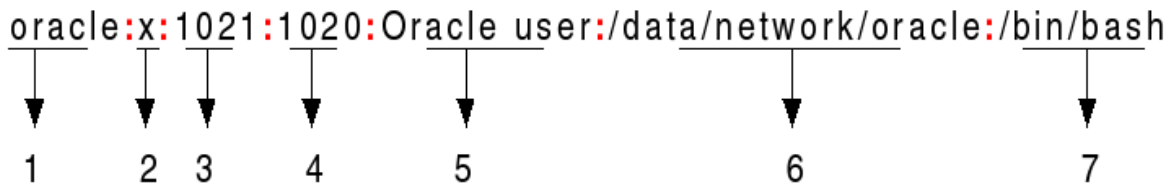
/ETC/PASSWD:: EXPLAINED

/etc/passwd file stores essential information, which is required during login i.e. user account information. /etc/passwd is a text file, that contains a list of the system's accounts, giving for each account some useful information like user ID, group ID, home directory, shell, etc. It should have general read permission as many utilities, like ls use it to map user IDs to user names, but write access only for the superuser (root).

UNDERSTANDING FIELDS IN /ETC/PASSWD

The /etc/passwd contains one entry per line for each user (or user account) of the system. All fields are separated by a colon (:) symbol. Total seven fields as follows.

Generally, passwd file entry looks as follows

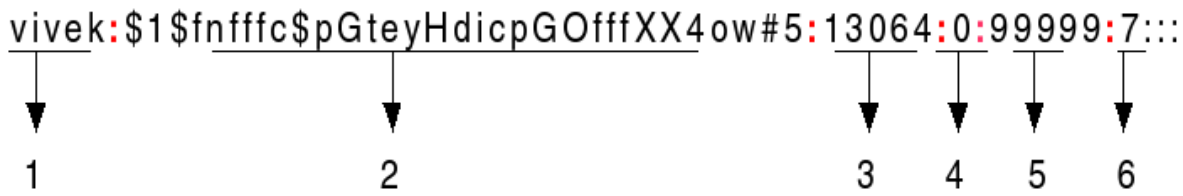


1. **Username**: It is used when user logs in. It should be between 1 and 32 characters in length.
2. **Password**: An x character indicates that encrypted password is stored in /etc/shadow file.
3. User ID (UID): Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups.
4. **Group ID (GID)**: The primary group ID (stored in /etc/group file)
5. **User ID Info**: The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. This field use by finger command.
6. **Home directory**: The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes /
7. **Command/shell**: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. Please note that it does not have to be a shell.

Note: UID for root is 0 , if any user have UID to 0 then user is treated as equivalent to root.

/ETC/SHADOW :: EXPLAINED

/etc/shadow file stores actual password in encrypted format for user's account with additional properties related to user password i.e. it stores secure user account information. All fields are separated by a colon (:) symbol. It contains one entry per line for each user listed in /etc/passwdfile Generally, shadow file entry looks as follows



/ETC/SHADOW FILE FIELDS EXPLAINED HERE

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 (last changed): 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

The last 6 fields provides password aging and account lockout features (you need to use **chage** command to setup password aging). According to man page of shadow - the password field must be filled. The encrypted password consists of 13 to 24 characters from the 64 character alphabet a through z, A through Z, 0 through 9, \. and /. Optionally it can start with a "$" character. This

means the encrypted password was generated using another (not DES) algorithm. For example if it starts with "1" it means the MD5-based algorithm was used.

What is the best way to edit /etc/passwd, shadow, and group files?

The best way to edit /etc/passwd, or shadow or group file is to use vipw command. Traditionally (under UNIX and Linux) if you use vi to edit /etc/passwd file and same time a user try to change a password while root editing file, then the user's change will not entered into file. To avoid this problem and to put a lock while editing file, use vipw and vigr command which will edit the files /etc/passwd and /etc/group respectively. If you pass -s option to these command, then they will edit the shadow versions of those files i.e. /etc/shadow and /etc/gshadow, respectively.

The main purpose of locks is to prevent file corruption. Do not use vi or other text editor to edit password file. Syntax:

- vipw -s : Edit /etc/passwd file
- vigr -s : Edit /etc/group file

Where,

- -s : Secure file editing

An example

Login as a root user:

vipw -s

On other terminal login as normal user (for example vivek) and issue command passwd to change vivek's password:

```
$ passwd

(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: Authentication token lock busy

As you see it returned with an error "passwd: Authentication token
lock busy"

This will avoid /etc/shadow file corruption.
```

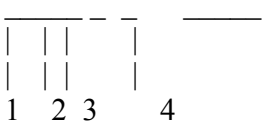
/etc/group :: Explained

/etc/group is a text file which defines the groups to which users belong under Linux and UNIX operating system. Under Unix / Linux multiple users can be categorized into groups. Unix file system permissions are organized into three classes, user, group, and others. The use of groups allows additional abilities to be delegated in an organized fashion, such as access to disks, printers, and other peripherals. This method, amongst others, also enables the Superuser to delegate some administrative tasks to normal users.

/ETC/GROUP FILE

It stores group information or defines the user groups i.e. it defines the groups to which users belong. There is one entry per line, and each line has the following format (all fields are separated by a colon (:)

```
cdrom:x:24:vivek,student13,raj
```



Where,

1. **group_name**: It is the name of group. If you run ls -l command, you will see this name printed in the group field.
2. **Password**: Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups.
3. **Group ID (GID)**: Each user must be assigned a group ID. You can see this number in your /etc/passwd file.
4. **Group List**: It is a list of user names of users who are members of the group. The user names, must be separated by commas.

More About User Groups

Users on Linux and UNIX systems are assigned to one or more groups for the following reasons:

- To share files or other resource with a small number of users
- Ease of user management
- Ease of user monitoring
- Group membership is perfect solution for large Linux (UNIX) installation.
- Group membership gives you or your user special access to files and directories or devices which are permitted to that group

/etc/login.defs Explained

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to
the
#   home directory.  If you _do_ define both, MAIL_DIR takes
precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be
used.
#       PASS_MIN_DAYS   Minimum number of days allowed between
password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a
password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
UID_MAX                 60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  1000
GID_MAX                 60000

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is overridden with the -m flag on
```

```
# useradd command line.
CREATE_HOME     yes

# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK        077

# This enables userdel to remove user groups if no members exist.
#
USERGROUPS_ENAB yes

# Use MD5 or DES to encrypt password? Red Hat use MD5 by default.
MD5_CRYPT_ENAB yes
```

/ETC/DEFAULT/USERADD EXPLAINED

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

/ETC/SKEL EXPLAINED

```
[root@localhostskel]# ls -la
total 80
drwxr-xr-x   4 root root  4096 Jan  2 08:27 .
drwxr-xr-x 137 root root 12288 Jan  2 08:27 ..
-rw-r--r--   1 root root    33 Jan 22  2009 .bash_logout
-rw-r--r--   1 root root   176 Jan 22  2009 .bash_profile
-rw-r--r--   1 root root   124 Jan 22  2009 .bashrc
-rw-r--r--   1 root root   515 May 24  2008 .emacs
drwxr-xr-x   3 root root  4096 Jan  1 06:13 .kde
drwxr-xr-x   4 root root  4096 Jan  1 06:17 .mozilla
-rw-r--r--   1 root root   658 Sep 22  2009 .zshrc
```

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 '**openpath**' is '**/home/openpath**'.

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@openpath ~]# 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@openpath ~]# cat /etc/passwd | grepanusha

anusha:x:505:505::/data/projects:/bin/bash
```

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@openpath ~]# useradd -u 999 navin
```

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

```
[root@openpath ~]# cat /etc/passwd | grepopenpath

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.

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@openpath ~]# useradd -u 1000 -g 500 tarunika
```

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

```
[root@openpath ~]# cat /etc/passwd | greptarunika

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

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 '**openpath**' into multiple groups like **admins**, **webadmin** and **developer**.

```
[root@openpath ~]# useradd -G admins,webadmin,developersopenpath
```

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

```
[root@openpath ~]# idopenpath

uid=1001(openpath) gid=1001(openpath)
groups=1001(openpath),500(admins),501(webadmin),502(developers)
context=root:system_r:unconfined_t:SystemLow-SystemHigh
```

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@openpath ~]# useradd -M shilpi
```

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

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

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

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@openpath ~]# 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@openpath ~]# 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
```

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 'openpath' using '**-e**' and '**-f**' options.

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

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, **ManisKhurana**, into the comment field.

[root@openpath ~]# useradd -c "MansiKhurana" mansi

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

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

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

Change User Login Shell:

Sometimes, we add users which has nothing to do with login shell or sometimes we require to assign 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 '**openpath**' without login shell i.e. '**/sbin/nologin**' shell.

```
[root@openpath ~]# useradd -s /sbin/nologinopenpath
```

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

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

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

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

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

```
[root@openpath ~]# useradd -m -d /var/www/vivek -s /bin/bash -c
"Openpath Owner"  vivek
```

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.

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@openpath ~]# useradd -m -d /var/www/tarunika -s /bin/zsh -c
"Openpath Technical Writer" -u 1000 -g 1000 tarunika
```

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

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

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@openpath ~]# useradd -M -N -r -s /bin/false -c "Disabled
Openpath Member" clayton
```

For more information and options about useradd, run '**useradd**' command on the terminal to see available options

Change attribute of already created user

Options of Usermod

The '**usermod**' command is simple to use with lots of options to make changes to an existing user. Let us see how to use usermod command by modifying some existing users in Linux box with the help of following options.

1. **-c** = We can add comment field for the useraccount.
2. **-d** = To modify the directory for any existing user account.
3. **-e** = Using this option we can make the account expiry in specific period.
4. **-g** = Change the primary group for a User.
5. **-G** = To add a supplementary groups.
6. **-a** = To add anyone of the group to a secondary group.
7. **-l** = To change the login name from openpath to openpath_admin.
8. **-L** = To lock the user account. This will lock the password so we can't use the account.
9. **-m** = moving the contents of the home directory from existing home dir to new dir.
10. **-p** = To Use un-encrypted password for the new password. (NOT Secured).
11. **-s** = Create a Specified shell for new accounts.
12. **-u** = Used to Assigned UID for the user account between 0 to 999.
13. **-U** = To unlock the user accounts. This will remove the password lock and allow us to use 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 '**openpath**' user, using the following command.

```
# usermod -c "This is Openpath" openpath
```

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

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

openpath:x:500:500:This is Openpath:/home/openpath:/bin/sh
```

Add Information to User

2. Change User Home Directory

In the above step we can see that our home directory is under **/home/openpath/**, 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/openpath' /etc/passwd

openpath:x:500:500:This is Openpath:/home/openpath:/bin/sh
```

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

```
# usermod -d /var/www/ openpath
# grep -E --color '/var/www/' /etc/passwd

openpath:x:500:500:This is Openpath:/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 openpath

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 '**openpath**' 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 openpath
# chage -l openpath
```

```
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**' withusermod command. Before, changing user primary group, first make sure to check the current group for the user **openpath_test**.

```
# idopenpath_test

uid=501(openpath_test) gid=502(openpath_test)
groups=502(openpath_test)
```

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

```
# usermod -g babinopenpath_test
# idopenpath_test

uid=501(openpath_test) gid=502(babin) groups=502(openpath_test)
```

5. Adding Group to an Existing User

If you want to add a new group called '**openpath_test0**' to '**openpath**' user, you can use option '**-G**' with usermod command as shown below.

```
# usermod -G openpath_test0 openpath
# idopenpath
```

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 **openpath_test0** with the **wheel** user.

```
# usermod -a -G wheel openpath_test0
# idopenpath_test0
```

So, user **openpath_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 **openpath** to **openpath_admin**. So the username **openpath** has been renamed with the new name **openpath_admin**.

```
# usermod -l openpath_adminopenpath
```

Now check for the **openpath** user, It will not be present because we have changed it to **openpath_admin**.

```
# id openpath
```

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

```
# id openpath_admin
```

Change User Login Name

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
```

Check for the locked account.

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

Lock User Account

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
```

Unlock User Account

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 it's 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/
```

Move User Home Directory

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
```

Create Unencrypted User Password

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/shbabin
```

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

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

Change User Login Shell

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
```

Change User UID

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
```

Using Multiple Options with usermod

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 want 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
```

/etc/profile and /etc/bashrc

What is /etc/profile used for?

If you have been using Linux for a while you are probably familiar with the .profile or .bash_profile files in your home directory. These files are used to set environmental items for a users shell. Items such as umask, and variables such as PS1 or PATH.

The /etc/profile file is not very different however it is used to set system wide environmental variables on users shells. The variables are sometimes the same ones that are in the .bash_profile, however this file is used to set an initial PATH or PS1 for all shell users of the system.

/etc/profile.d

In addition to the setting environmental items the /etc/profile will execute the scripts within /etc/profile.d/*.sh. If you plan on setting your own system wide environmental variables it is recommended to place your configuration in a shell script within /etc/profile.d.

What is /etc/bashrc used for?

Like .bash_profile you will also commonly see a .bashrc file in your home directory. This file is meant for setting command aliases and functions used by bash shell users.

Just like the /etc/profile is the system wide version of .bash_profile. The /etc/bashrc for Red Hat and /etc/bash.bashrc in Ubuntu is the system wide version of .bashrc.

Interestingly enough in the Red Hat implementation the /etc/bashrc also executes the shell scripts within /etc/profile.d but only if the users shell is a Interactive Shell (aka Login Shell)

When are these files used?

The difference between when these two files are executed are dependent on the type of login being performed. In Linux you can have two types of login shells, Interactive Shells and

Non-Interactive Shells. An Interactive shell is used where a user can interact with the shell, i.e. your typical bash prompt. Whereas a non-Interactive shell is used when a user cannot interact with the shell, i.e. a bash scripts execution.

The difference is simple, the /etc/profile is executed only for interactive shells and the /etc/bashrc is executed for both interactive and non-interactive shells. In fact in Ubuntu the /etc/profile calls the /etc/bashrc directly.

Interactive Shell vs Non-Interactive Shell

To show an example of an interactive shell vs a non-interactive shell I will add a variable into both /etc/profile and /etc/bash.bashrc on my Ubuntu system.

/etc/profile

```
# grep TEST /etc/profile
export TESTPROFILE=1
```

/etc/bash.bashrc

```
# grep TEST /etc/bash.bashrc
export TESTBASHRC=1
```

Interactive Shell

The below example is showing an interactive shell, in this case both the /etc/profile and /etc/bash.bashrc was executed.

```
# su -
# env | grep TEST
TESTBASHRC=1
TESTPROFILE=1
```

Non-Interactive Shell

In this example we are running a command through SSH that is non-interactive; because this is a non-interactive shell only the /etc/bash.bashrc file is executed.

```
# sshlocalhost "env | grep TEST"
root@localhost's password:
TESTBASHRC=1
```

How to force user to change password in next login:

```
chage -d0 vikas
```

```
vikas:$6$shEeh82b$Mawd6C7Q72q2fSkf3c1/p6aSsipIwiPl4WLcTWUezJziVZ9oPeT2
Ds9rQ00g/gLs.s5QeIN4iS78LdMStbWDo0:0:0:99999:7:::
```

Run a command when log out

Edit/append the following in your$HOME/.bash_logout (BASH) or $HOME/.logout (CSH/TCSH) file:

```
## clear the screen ##
clear
## Disk info ##
du -s $HOME

## Delete mysqlcmdfile ##
/bin/rm $HOME/.mysql_history

## add rest of the stuff here for bash
```

Add user to Single Group

```
Groupadd purchase

useradd –g purchase ravi
```

Add user to multiple groups

You can add the user to multiple secondary groups using single command

```
# useradd-G admin,dba,deepak
```

Verify

```
# groups deepak

deepak : deepak admin dba
```

Bulk User Creation:

/root/create-bulkuser.sh

```
#!/bin/bash
for i in `cat unpw.csv`; do
  UN=`echo $i | cut -f1 -d','`
  PW=`echo $i | cut -f2 -d','`
```

```
  PASS=`opensslpasswd -crypt $PW`
echouseradd -p $PW $UN
echo $UN
echo $PW
echo $PASS
useradd -p $PASS $UN
done
```

/root/unpw.csv

musariq1,123
ravi1,123

/etc/shadow

musariq1:RMyKQVc1pVdCM:16811:0:99999:7:::
ravi1:5ttyUK1LepGFA:16811:0:99999:7:::

CHECK SUCCESSFUL & UNSUCCESSFUL USER LOGIN ATTEMPTS IN LINUX

For Linux System admins it is very important to know successful & unsuccessful user login attempts on their Linux boxes. In this post we will discuss the commands that will help Linux system admins to determine successful & unsuccessful user login attempts.

LAST COMMAND :

The last command shows the history the successful user login attempts & system reboot details by reading the file /var/log/wtmp. This file capture all login and logout sessions including login time, duration a user stayed logged in &tty(terminal) where the user's session took place. To display all user login, logout & system reboot activities, type the '**last**' command on terminal without any arguments. Example is shown below :

```
root@mail2 ~]# last
rootpts/0        117.206.178.226  Sun Nov 30 10:47   still logged in
rootpts/0        117.206.178.226  Sat Nov 29 22:47 - 22:50  (00:03)
rootpts/1        117.206.178.226  Sat Nov 29 22:17 - 22:46  (00:29)
rootpts/0        117.206.183.48   Wed Nov 26 21:35 - 21:50  (00:14)
rootpts/0        117.206.185.124  Tue Nov 25 23:23 - 23:24  (00:01)
```

.....

TO DISPLAY ONLY SYSTEM REBOOT DETAILS:

```
[root@mail2 ~]# last reboot
```

```
reboot   system boot  2.6.32-431.23.3. Sun Sep  7 02:07 - 10:49
(84+09:41)
reboot   system boot  2.6.32-431.23.3. Sun Sep  7 01:58 - 02:07
(00:08)
reboot   system boot  2.6.32-431.17.1. Sat Sep  6 12:13 - 01:58
(13:44)
```

```
wtmp begins Sat Sep  6 12:13:56 2014
```

There is another command that lists more detailed information on recent logins and reboots.This command is **utmpdump** and is executed the following way:

```
[root@mail2 ~]# utmpdump /var/log/wtmp
```

```
[root@mail2 ~]# utmpdump /var/log/wtmp
Utmp dump of /var/log/wtmp
[2] [00000] [~~  ] [reboot  ] [~           ] [2.6.32-431.17.1.el6.x86_64] [0.0.0.0        ] [Sat Sep 06 12:13:56 2014 CDT]
[1] [00051] [~~  ] [runlevel] [~           ] [2.6.32-431.17.1.el6.x86_64] [0.0.0.0        ] [Sat Sep 06 12:13:56 2014 CDT]
[6] [01522] [v/hv] [LOGIN   ] [/dev/hvc0   ] [                    ] [0.0.0.0        ] [Sat Sep 06 12:15:24 2014 CDT]
[6] [01523] [1   ] [LOGIN   ] [tty1        ] [                    ] [0.0.0.0        ] [Sat Sep 06 12:15:24 2014 CDT]
[6] [01525] [2   ] [LOGIN   ] [tty2        ] [                    ] [0.0.0.0        ] [Sat Sep 06 12:15:24 2014 CDT]
[6] [01527] [3   ] [LOGIN   ] [tty3        ] [                    ] [0.0.0.0        ] [Sat Sep 06 12:15:24 2014 CDT]
.....................
```

LASTB COMMAND :

The lastb command display the information of bad login attempts or unsuccessful login attempts by reading the file /var/log/btmp. This file keeps the track of all unsuccessful login attempt activities inlcluding login name, time & the tty (terminal) where the attempt was made. To display all unsuccessful login attempts, type the '**lastb**' command on the terminal without any arguments. Example is shown below.

```
[root@mail2 ~]# lastb
adminssh:notty    125.161.19.132   Sun Nov 30 09:49 - 09:49  (00:00)
adminssh:notty    125.161.19.132   Sun Nov 30 09:48 - 09:48  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
rootssh:notty    61.174.49.105    Sun Nov 30 09:33 - 09:33  (00:00)
```

....

LASTLOG COMMAND :

The lastlog command displays information of most recent logins of all users or a given user by reading the file /var/log/lastlog.

```
[root@mail2 ~]# lastlog
```

```
Username         Port     From             Latest
rootpts/0    117.206.178.226  Sun Nov 30 10:47:03 -0600 2014
bin                                       **Never logged in**
daemon                                    **Never logged in**
adm                                       **Never logged in**
lp                                        **Never logged in**
sync                                      **Never logged in**
shutdown                                  **Never logged in**
..............
```

User essential plus

Useradd command explained

Creating user with custom home directory

```
#useradd –b /rhcetestdata rhcetestuser3
Or
#useradd –d /rhcetestdata/rhcetestuser3  rhcetestuser3
```

Creating user without home directory

```
#useradd –M rhcetestuser4
```

Note: Regardless home directory is created or not, its entry will be placed in /etc/passwd. This default behavior allows administrators to, if require, create a home directory for user in future.

The useradd command options cheat sheet

The useradd command supports several options and arguments. From them, important options with default arguments are explained in following table.

Short option	Full option	Description	Default

-b	--base-dir	Defines the absolute path of the base directory to place users' home directories.	Default is **/home** directory.
-c	--comment	Sets a description for user account. Usually it is used to store user's full name. Use quotes, If description contains white spaces or multiple words for example "Sanjay Kumar Goswami".	Default is blank. If we don't use this option, description will set to blank.
-d	--home-dir	Sets the name of user's home directory.	Default is login name. If this option is not used, user's home directory name will be same as his username.
-D	--default	Instead of creating new user, this option force command to use supplied information as the default setting for any new accounts that will be created.	Default value of each option is explained individually along with their respective options in this row.
-e	--expiredate	Specifies a date in YYYY-MM-DD format after that account will be disabled automatically.	Default is blank. If no expiry date is specified through this option, account will never disable.
-f	--inactive	Sets number of days after a password expires before disabling account permanently.	Default is blank. If this option is not used, account will not be disabled, even password is expired.
-g	--gid	Sets primary group for user. In order to use this option, group must already exist in the **/etc/group** file.	By default, a new group named same as username is created and user is added in this group.
-G	--groups	Set secondary groups for user. In order to specify multiple groups' names, use comma (for example, -G wheel,developer,programmer).	By default, user is not added in any secondary group.
-k	--skel	Set the location of skeleton directory. The skeleton directory contains initial configuration files and login scripts. These files are copied to user's home	Default is **/etc/skel** directory. By default this directory contains three bash shell

		directory at the time of account creation.	files; **.bash_profile**, **.bashrc** and **.bash_logout**.
-m	--create-home	Creates user's home directory and copy initial configuration files and scripts from skeleton directory. Check user's home directory in the base directory.	Base directory is the directory which we set with **–b** option. If it doesn't exist, create a new directory there and copy initial configuration files from skeleton directory.
-M	--no-create-home	Don't create home directory for user.	Prevents command from creating user's home directory
-n	--no-user-group	Don't create primary group for user.	Stop command from creating primary group for user.
-o	--non-unique	Creates a user account which uses the UID of an existing user. When two users share a common UID, both get identical rights.	Force command to create a duplicate user account.
-r	--system	Creates a system or service account	Force command to create an account which has UID below 1000
-s	--shell	Defines the absolute path of command shell for this account.	Default is **/bin/bash**.
-u	--user-group	Specify the UID of user account.	Default is the next available UID from **/etc/passwd** file. For example last used UID in **/etc/passwd** file is 1050, then UID 1051 will be used for this account.

Disable Root Account in Linux

Change root User's Shell

The simplest method to disable root user login is to change its shell from /bin/bash or /bin/bash (or any other shell that permits user login) to /sbin/nologin, in the /etc/passwd file, which you can open for editing using any of your favorite command line editors as shown.

```
#  vim /etc/passwd
```

Change the line:

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

to

```
root:x:0:0:root:/root:/sbin/nologin
```

Change root User Shell

From now on, when root user logs in, he/she will get the message “This account is currently not available.” This is the default message, but, you can change it and set a custom message in the the file **/etc/nologin.txt**.

This method is only effective with programs that require a shell for user login, otherwise, **sudo, ftp and email clients can access the root account**.

Disable root Login via Console Device (TTY)

The second method uses a PAM module called pam_**securetty**, which permits root access only if the user is logging in on a “secure” TTY, as defined by the listing in **/etc/securetty**.

The above file allows you to specify which TTY devices the root user is allowed to login on, emptying this file prevents root login on any devices attached to the computer system.

To create an empty file, run.

```
# mv /etc/securetty /etc/securetty.orig
# touch /etc/securetty
# chmod 600 /etc/securetty
```

This method has some limitations, it only affects programs such as login, display managers (i.e gdm, kdm and xdm) and other network services that launch a TTY. Programs such as su, sudo, ssh, and other related openssh tools will have access to the root account.

Disable SSH Root Login

The commonest way of accessing remote servers or VPSs is via SSH and to block root user login under it, you need to edit the /etc/ssh/sshd_config file.

```
# vim /etc/ssh/sshd_config
```

Then uncomment (if it is commented) the directive **PermitRootLogin** and set its value to no as shown in the screenshot.

Disable Root Login in SSh

Once you are done, save and close the file. Then restart the sshd service to apply the recent change in configurations.

```
# systemctl restart sshd
```

OR

```
# service sshd restart
```

As you may already know, this method only affects openssh tools set, programs such as ssh, scp, sftp will be blocked from accessing the root account.

Restrict root Acess to Services Via PAM

Pluggable Authentication Modules (PAM in short) is a centralized, pluggable, modular, and flexible method of authentication on Linux systems. PAM, through the **/lib/security/pam_listfile.so** module, allows great flexibility in limiting the privileges of specific accounts.

The above module can be used to reference a list of users who are not allowed to log in via some target services such as login, ssh and any PAM aware programs.

In this case, we want to disable root user access to a system, by restricting access to login and sshd services. First open and edit the file for the target service in the **/etc/pam.d/** directory as shown.

```
# vim /etc/pam.d/login
```

OR

```
# vim /etc/pam.d/sshd
```

Next, add the configuration below in both files.

```
auth    required        pam_listfile.so
\onerr=succeed  item=user  sense=deny
file=/etc/ssh/deniedusers
```

When you are done, save and close each file. Then create the plain file **/etc/ssh/deniedusers** which should contain one item per line and not world readable.

Add the name root in it, then save and close it.

```
# vim /etc/ssh/deniedusers
```

Also set the required permissions on this.

```
# chmod 600 /etc/ssh/deniedusers
```

This method only affect programs and services that are PAM aware. You can block root access to the system via ftp and email clients and more.

How to Lock User Accounts After Failed Login Attempts

You can configure the above functionality in the **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth** files, by adding the entries below to the auth section.

```
auth    required pam_faillock.so preauth silent audit deny=3
unlock_time=600
auth    [default=die]  pam_faillock.so authfail audit deny=3
unlock_time=600
```

Where:

- **audit –** enables user auditing.
- **deny** – used to define the number of attempts (3 in this case), after which the user account should be locked.
- **unlock_time** – sets the time (300 seconds = 5 minutes) for which the account should remain locked.

Note: ***The order of these lines is very important, wrong configurations can cause all user accounts to be locked.***

The auth section in both files should have the content below arranged in this order:

```
auth        required      pam_env.so
auth        required      pam_faillock.so preauth silent audit deny=3
unlock_time=300
auth        sufficient    pam_unix.so  nullok  try_first_pass
auth        [default=die]  pam_faillock.so  authfail  audit  deny=3
unlock_time=300
auth        requisite     pam_succeed_if.so uid >= 1000 quiet_success
auth        required      pam_deny.so
```

Now open these two files with your choice of editor.

 # vi /etc/pam.d/system-auth

 # vi /etc/pam.d/password-auth

The default entries in auth section both files looks like this.

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000 quiet
auth        required      pam_deny.so
```

After adding the above settings, it should appear as follows.

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        required      pam_faillock.so preauth silent audit deny=3
unlock_time=300
auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
```

```
auth        [default=die]  pam_faillock.so  authfail  audit  deny=3
unlock_time=300

auth        requisite     pam_succeed_if.so uid >= 1000 quiet

auth        required      pam_deny.so
```

Then add the following highlighted entry to the account section in both of the above files.

```
account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     required      pam_permit.so
account     required      pam_faillock.so
```

How to Lock Root Account After Failed Login Attempts

To lock the root account after failed authentication attempts, add the even_deny_root option to the lines in both files in the auth section like this.

```
auth        required      pam_faillock.so preauth silent audit deny=3
even_deny_root unlock_time=300

auth        [default=die]  pam_faillock.so  authfail  audit  deny=3
even_deny_root unlock_time=300
```

Once you have configured everything. You can restart remote access services like sshd, for the above policy to take effect that is if users will employ ssh to connect to the server.

```
# systemctl restart sshd  [On SystemD]

# service sshd restart    [On SysVInit
```

How to Test SSH User Failed Login Attempts

From the above settings, we configured the system to lock a user's account after 3 failed authentication attempts.

In this scenario, the user vikas is trying to switch to user mohan, but after 3 incorrect logins because of a wrong password, indicated by the "Permission denied" message, the user mohan's account is locked as shown by "authentication failure" message from the fourth attempt.

Test User Failed Login Attempts

The root user is also notified of the failed login attempts on the system, as shown in the screen shot below.

Failed Login Attempts Message

How to View Failed Authentication Attempts

You can see all failed authentication logs using the faillock utility, which is used to display and modify the authentication failure log.

You can view failed login attempts for a particular user like this.

```
# faillock --user mohan
```

View User Failed Login Attempts

To view all unsuccessful login attempts, run faillock without any argument like so:

```
# faillock
```

To clear a user's authentication failure logs, run this command.

```
# faillock --user aaronkilik --reset
```

OR

fail --reset #clears all authentication failure records

Lastly, to tell the system not to lock a user or user's accounts after several unsuccessful login attempts, add the entry marked in red color, just above where pam_faillock is first called under the auth section in both files (/etc/pam.d/system-auth and /etc/pam.d/password-auth) as follows.

Simply add full colon separated usernames to the option user in.

```
auth  required      pam_env.so
auth   [success=1 default=ignore] pam_succeed_if.so user in vikas:mohan
auth   required      pam_faillock.so preauth silent audit deny=3 unlock_time=600
auth   sufficient    pam_unix.so  nullok  try_first_pass
```

```
auth    [default=die]  pam_faillock.so  authfail  audit  deny=3  unlock_time=600
auth    requisite      pam_succeed_if.so uid >= 1000 quiet_success
auth    required       pam_deny.so
```

TMOUT – Auto Logout Linux Shell When There Isn't Any Activity

How often do you leave a Linux system idle after login; a situation which can be referred to as an 'idle session', where you are not attending to the system by running commands or any administration tasks.

However, this normally presents a great security risk, especially when your logged on as the superuser or with an account that can gain root privileges and in the event that someone with malicious intend gains physical access to your system, he or she can executes some destructive commands or do what ever they want to achieve on it, in the shortest time possibles.

To enable automatic user logout, we will be using the TMOUT shell variable, which terminates a user's login shell in case there is no activity for a given number of seconds that you can specify.

To enable this globally (system-wide for all users), set the above variable in the /etc/profile shell initialization file.

```
# vi /etc/profile
```

Add the following line.

```
TMOUT=120
```

Save and close the file. From now on, a user will be logged out after 120 seconds (2 minutes), if he or she is not attending to the system.

Note that users can configure this in their own shell initialization file **~/.profile.** This means that once that particular user has no activity on the system for the specified second, the shell automatically terminates, thus logging out that user.

Set SSH Root Login Email Alerts

mailx (Mail Client) to be installed on the server to send the emails. you can install mailx client using one of the following commands.

```
# yum install mailx
```

Now login as root user and go to root's home directory by typing cd /root command.

```
# cd /root
```

Next, add an entry to the .bashrc file. This file sets local environment variables to the users and does some login tasks. For example, here we setting a an email login alert.

Open .bashrc file with vi or nano editor. Please remember .bashrc is a hidden file, you won't see it by doing ls -l command. You've to use -a flag to see hidden files in Linux.

```
# vi .bashrc
```

Add the following whole line at the bottom of the file. Make sure to replace "ServerName" with a hostname of your Server and change "your@yourdomain.com" with a your email address.

```
echo 'ALERT - Root Shell Access (ServerName) on:' `date` `who` | mail -s "Alert: Root Access from `who | cut -d'(' -f2 | cut -d')' -f1`" your@yourdomain.com
```

Save and close the file and logout and log back in. Once you login via SSH, a **.bashrc** file by default executed and sends you an email address of the root login alert.

Sample Email Alert>>

```
ALERT - Root Shell Access (Samba Server) on: Thu Nov 28 16:59:40 IST 2021 openpath pts/
```