CHAPTER-6 MANAGE LOCAL USERS AND GROUPS

Describe User and Group Concepts

User

A user account provides security boundaries between different people and programs that can run commands. Users have user names to identify them to human users and for ease of working. Internally, the system distinguishes user accounts by the unique identification number, the user ID or UID, which is assigned to them.

User accounts are of the following main types:

The superuser : The superuser account administers the system. The superuser name is root and the account has a UID of 0. The superuser has full system access.

System users: The system user accounts are used by processes that provide supporting services. These processes, or daemons, usually do not need to run as the superuser. They are assigned non-privileged accounts to secure their files and other resources from each other and from regular users on the system. Users do not interactively log in with a system user account.

Regular users: Most users have regular user accounts for their day-to-day work. Like system users, regular users have limited access to the system.

Basic command to views details related about users

DESCRIPTION	COMMANDS / OPTIONS
Important information about who is currently using the computer	w
To display information about currently logged-in users	Who
To show information about the currently logged-in user	**Syntax:** id [Options] [UserName] **-g :** Print only the effective group id. **-G :** Print all Group ID's. **-n :** Prints name instead of number. **-r :** Prints real ID instead of num. **-u :** Prints only the effective user ID. **Example:** [user@host ~]$id –G UserName ↵

List the content of file folder	**Syntax:** ls [option] [file/directory] **-l** long list **-a** hidden file **-R** Recursive **-t** Sort files and directories last modify **-r** reverse order **-S** Sort files and directories by their sizes **-i** inode **-h** file sizes in human-readable format **Example:** [user@host ~]$ ls -l↵

The mapping of **usernames to UIDs** is defined in databases of account information. By default, systems use **the /etc/passwd file to store** information **about local users**. Each line in the **/etc/passwd file contains information** about one user. The file is **divided into seven colon-separated fields**. An example of a line from /etc/passwd follows:

```
[user01@host ~]$ cat /etc/passwd
...output omitted...
user01:x:1000:1000:User One:/home/user01:/bin/bash
```

user01: The username for this user.
x: The user's encrypted password was historically stored here; this is now a placeholder.
1000: The UID number for this user account.
1000: The GID number for this user account's primary group. Groups are discussed later in this section.
User One: A brief comment, description, or the real name for this user.
/home/user01: The user's home directory, and the initial working directory when the login shell starts.
/bin/bash: The default shell program for this user that runs at login. Some accounts use the /sbin/nologin shell to disallow interactive logins with that account.

What Is a Group?

A group is a collection of users that need to share access to files and other system resources. Groups can be used to grant access to files to a set of users instead of to a single user.
Groups have group names for easier recognition. Internally, the system distinguishes groups by the unique identification number, the **group ID or GID**, which is assigned to them. The mapping of group names to GIDs is defined in identity management databases of group account information. By default, **systems use the /etc/group file to store information** about local groups.

Each line in **the /etc/group file** contains information about one group. Each group entry is divided **into four colon-separated fields**. An example of a line from /etc/group follows:

```
[user01@host ~]$ cat /etc/group
...output omitted...
group01:x:10000:user01,user02,user03
```

group01: Name for this group.
x: Obsolete group password field; this is now a placeholder.
10000: The GID number for this group (10000).
user01, user02, user03: A list of users that are members of this group as a supplementary group.

Primary Groups and Supplementary Groups

Every user has exactly **one primary group**. For **local users, this group is listed by GID** in the **/etc/passwd file**. The primary group owns files that the user creates.

When **creating a regular user, a group is created with the same name** as the user, to be the **primary group for the user**. The user is the only member of this User Private Group. This group membership design simplifies the management of file permissions, to have user groups segregated by default. Users might also have supplementary groups. **Membership in supplementary groups is stored in the /etc/group file.**

Gain Superuser Access

Switch to the superuser account to manage a Linux system, and grant other users superuser access through the **sudo command.**
The root account on Linux is **roughly equivalent to the local Administrator account on Microsoft Windows**. In Linux, most system administrators log in to the system as an unprivileged user and use various tools to temporarily gain root privileges.

Switch User Accounts

The **su command**, users can **switch to a different user account**. If you run the su command from a regular user account with another user account as a parameter, then you must provide the password of the account to switch to. When the root user runs the su command, you do not need to enter the user's password.

DESCRIPTION	COMMANDS / OPTIONS
Switch to user account	**Syntax:** su [options] userName **Example:** [user@host ~]$ su UserName ↵ **su** The command su starts a non-login shell, **-** starts a login shell **-c** executing the specific command as another user with a login shell ensuring the user's environment is fully uploaded.

DESCRIPTION	COMMANDS / OPTIONS
sudo command	**Syntax:** sudo [options] command **Example:** [user@host ~]$ sudo mkdir dir1 ↵ **-i** To the root account and runs that user's default shell (usually bash) and associated interactive login scripts. **-s** To run the shell without the interactive scripts

Configure sudo

The /etc/sudoers file is the main configuration file for the sudo command. To avoid problems if multiple administrators try to edit the file at the same time, you can edit it only with the special visudo command. The visudo editor also validates the file, to ensure no syntax errors.

For example, the following line from the **/etc/sudoers file** enables sudo access for wheel group members:

```
%wheel        ALL=(ALL:ALL)       ALL
```

- The **%wheel string is the user or group** that the **rule applies to. The % symbol before** the word **wheel specifies a group**.

- The **ALL=(ALL:ALL) command specifies** that on any host with this file (the first ALL), users in the **wheel group can run commands as any other user** (the second ALL) and any **other group (the third ALL) on the system**.

- The **final ALL command specifies that users in the wheel group** can run any command.

By default, the **/etc/sudoers** file also includes the contents of any files in the **/etc/sudoers.d** directory as part of the configuration file. With this hierarchy, you can add sudo access for a user by putting an appropriate file in that directory.

- To enable **full sudo access** for the **user01 user**, you can **create the /etc/sudoers.d/user01** file with the following content:

```
user01        ALL=(ALL)       ALL
```

- To enable **full sudo access for the group01 group**, you can **create the /etc/sudoers.d/ group01** file with the following content:

```
%group01        ALL=(ALL)       ALL
```

- To enable **users in the games group to run the id command as the operator user**, you can **create the /etc/sudoers.d/games file** with the following content:

```
%games ALL=(operator) /bin/id
```

- You can also set up sudo **to allow a user to run commands as another user without entering their password,** by using the NOPASSWD: ALL command:

```
ansible        ALL=(ALL)       NOPASSWD: ALL
```

Manage Local User Accounts

Create Users from the Command Line

The **useradd username** command creates a user called username. It sets up the user's home directory and account information, and creates a private group for the user called username. At this point, a valid password is not set for the account, and the user cannot log in until a password is set.

The **useradd --help** command **displays the basic options to override the defaults**. In most cases, you can **use the same options with the usermod command to modify an existing user**. The **/etc/login.defs file sets some of the default options for user accounts**, such as the **range of valid UID numbers** and **default password aging rules**. The values in this file affect only newly created user accounts. A change to this file does not affect existing users.

DESCRIPTION	COMMANDS / OPTIONS
To create user account	**Syntax:** useradd [options] userName **Example:** [root@host ~]# useradd Jon ↲ **-d** To give a home directory path for new users **-u** To create a new user with a custom UID **-g** specific group ID **-M** To create a user without a home directory **-e** To set an expiry date for a user account (YY-MM-DD) **Example:** [user@host ~]$ sudo useradd -e 2020-05-30 Jon ↲ -c To add a comment or description for a user -s To create a user with a different login shell (/bin/sh) To set login sell (/sbin/nologin) To disable login sell **-p** To set an unencrypted password for the user **Example:** [user@host ~]$ sudo useradd -p password Jon ↲

DESCRIPTION	COMMANDS / OPTIONS
To modify user account	**Syntax:** usermod [options] userName **Example:** [root@host ~]# usermod Jon ↵ **-a** To append with G options **-c** To add a comment or description for a user **-d** To Specify a home directory for the user account **-g** Specify the primary group for the user account. **Example:** [root@host ~]# usermod –g GroupName Jon ↵ **-G** supplementary group **Example:** [root@host ~]# usermod -aG GroupName Jon ↵ **-L** Lock the user account **-U** Unlock user account **-m** Move the user's home directory to a new location. You must use it with the -d option. **-e** To change the expiry date of a user) **Example:** [user@host ~]$ sudo usermod -e 2020-05-30 Jon ↵ **-s** To create a user with a different login shell (/bin/sh) To set login sell (/sbin/nologin) To disable login sell
Delete Users from the Command Line	**Syntax:** userdel [options] UserName **-r** delete user with home directory The **userdel username** command **removes the username** user from **/etc/passwd**, but **leaves the user's home directory intact**. The userdel -r username command removes the user from /etc/passwd and deletes the user's home directory.

Manage Local Group Accounts

The groupadd command creates groups. Without options, the groupadd command uses the next available GID from the range specified by the GID_MIN and GID_MAX variables in the /etc/login.defs file. By default, the command assigns a GID value greater than any other existing GIDs, even if a lower value becomes available.

DESCRIPTION	COMMANDS / OPTIONS
To create group	**Syntax:** groupadd [options] GroupName **Example:** [root@host ~]# groupadd Developer ↵ **-r** System group **-g** GID **-U** list of user members of this group **-f** force -p Password
Modify Existing Groups	**Syntax:** groupmod [options] GroupName **Example:** [root@host ~]# groupmod Developer ↵ **-g** The group ID of the given GROUP will be changed to GID **-n** The name of group will change into new_name. **Example:** [root@host ~]# groupmod -n group_new group_old ↵ **-p** This gives the encrypted password. **-R** Apply changes in the CHROOT_DIR directory and use the configuration files from the CHROOT_DIR directory. **-U** list of user members of this group
To delete group	**Syntax:** groupdel [options] GroupName **Example:** [root@host ~]# groupdel Developer ↵

Compare Primary and Supplementary Group Membership

A user's **primary group** is the group that is **viewed** on the user's account in the **/etc/passwd** file. A **user** can only **belong to one primary group at a time**.

A user's **supplementary groups are the additional groups** configured **for the user** and **viewed** on the user's entry in the **/etc/group** file. A user **can belong to as many supplementary groups** as is necessary **to implement file access and permission effectively**.

Temporarily Change Your Primary Group

Only a user's primary group is used for new file creation attributes. However, you can temporarily switch your primary group to another group, but you can only choose from supplementary groups to which you already belong. You might switch if you are about to create a number of new files, manually or scripted, and want them to have a different group assigned as owner as they are being created.

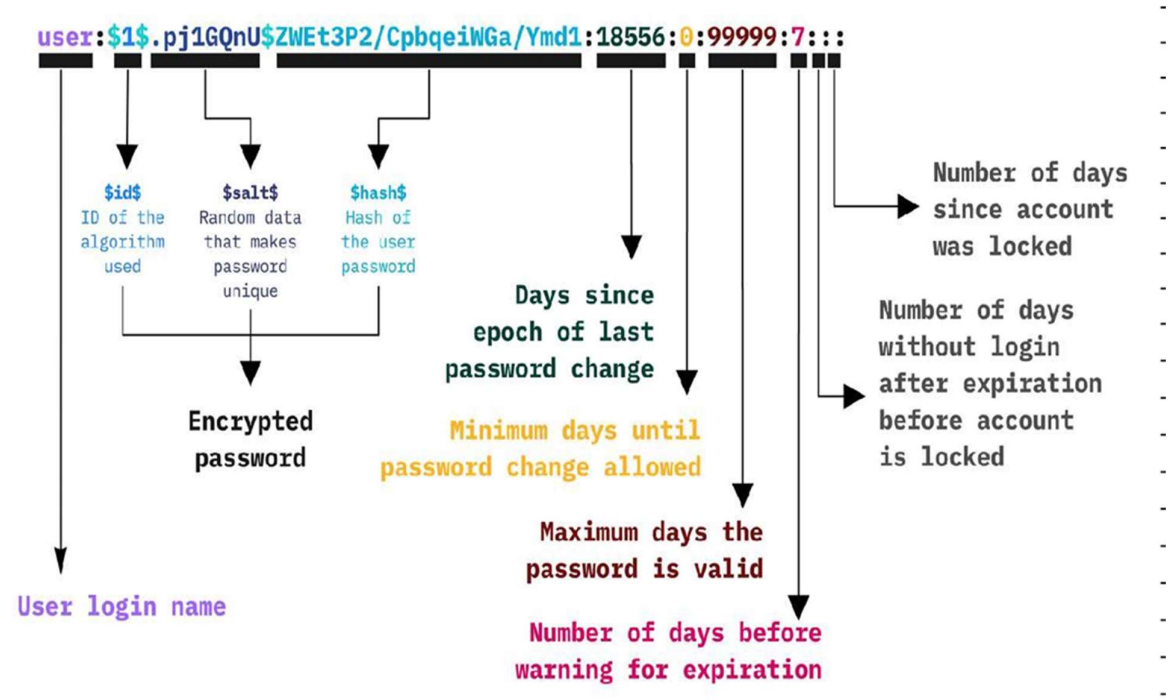
Use the newgrp command to switch your primary group, in this shell session. You can switch between any primary or supplementary group to which you belong, but only one at a time can be primary. Your primary group will return to the default if you log out and log in again. In this

Example: the group called group01 temporarily becomes this user's primary group.

```
[user03@host ~]# id
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
[user03@host ~]$ newgrp group01
[user03@host ~]# id
uid=1007(user03) gid=10000(group01) groups=1009(user03),10000(group01)
```

Manage User Passwords

Encrypted passwords were stored in the world-readable **/etc/passwd** file. This was considered adequate until dictionary attacks on encrypted passwords became common. The encrypted passwords were moved to the **/etc/shadow** file, which only the root user can read. Like the /etc/passwd file, each user has an entry with in the /etc/shadow file. An example entry from the /etc/shadow file has nine colon-separated fields:



Each field of this code block is separated by a colon:

user03 : Name of the user account.

6CSsXsd3rwghsdfarf : The encrypted password of the user.

17933 : The days from the epoch when the password was last changed, where the epoch is 1970-01-01 in the UTC time zone.

0: The minimum days since the last password change before the user can change it again.

99999: The maximum days without a password change before the password expires. An empty field means that the password never expires.

7: The number of days ahead to warn the user that their password will expire.

2: The number of days without activity, starting with the day the password expired, before the account is automatically locked.

18113: The day when the account expires in days since the epoch. An empty field means that the account never expires.

The last field is typically empty and reserved for future use.

Format of an Encrypted Password

The encrypted password field stores three pieces of information: the hashing algorithm in use, the salt, and the encrypted hash. Each piece of information is delimited by the dollar ($) character.

6CSsXcYG1L/4ZfHr/$2W6evvJahUfzfHpc9X.45Jc6H30E

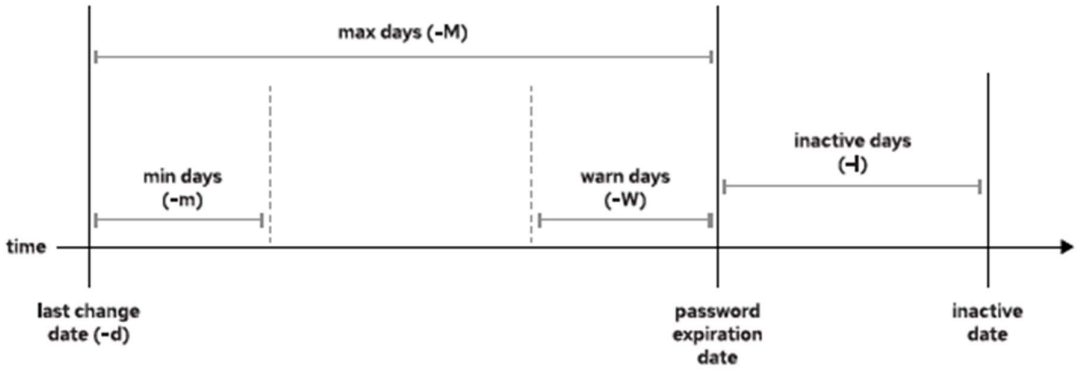
6: The hashing algorithm in use for this password. A 6 indicates a SHA-512 hash, the RHEL 9 default, a 1 indicates MD5, and a 5 indicates SHA-256.

CSsXcYG1L/4ZfHr/: The salt in use to encrypt the password; originally chosen at random.

2W6evvJahUfzfHpc9X.45Jc6H30E: The encrypted hash of the user's password; combining the salt and the unencrypted password and then encrypting to generate the password hash.

Configure Password Aging

The following diagram shows the relevant **password aging parameters**, which can be adjusted by using **the chage command** to implement a **password aging policy**. Notice that the command name is chage which stands for "change age", and it should not be confused with the word "change".



DESCRIPTION	COMMANDS / OPTIONS
To Configure Password Aging	**Syntax:** chage [options] UserName **Example:** [root@host ~]# chage Jon ↵ **-m/M** to specify the maximum and minimum number of days between password change **Example:** [root@host ~]# chage -M 5 root **-W** Warning Period **Example:** [root@host ~]# chage -W 2 UserName **-I** inactive period **Example:** [root@host ~]# chage -I 5 UserName **-E** To specify the date when the account expire **Example :** [root@host ~]# chage -E 2022-09-06 operator1 **-l** To view the account aging information **Example:** [root@host ~]# chage -l UserName **-d** To set the last password change date **Example :** [root@host ~]# chage -d 2018-12-01 root

DESCRIPTION	COMMANDS / OPTIONS
To administer the /etc/group and /etc/gshadow	**Syntax:** gpasswd [options] GroupName **-a** To add a user to the named group. **Example :** [root@host ~]# gpasswd -a [user] [group] ↲ **-d** To remove a user from the named group **Example:** [root@host ~]# gpasswd -d [user] [group] ↲ **-r** To remove the password from the named group **Example:** [root@host ~]# gpasswd -r developers ↲ **-R** To restrict the access to the named group. **-A** Set the list of administrative users. **-M** It set the list of group members.
To Change User Password	**Syntax:** Passwd[options][UserName] **-d** To delete password **-e** Immediately expires the account password **-i** inactive **-l** Locks the password of the user **-u** Unlocks the password of an account **-n** Rename group name
To change the file Owner or group	**Syntax:** chown [options] new_owner[:new_group] file(s) **-c** To report when a file change is made **Example :** chown -c UserName file1.txt ↲ -v -f **To Change the owner of a file** chown owner_name file_name ↲ **To change the group ownership of a file** chown :group1 file1.txt ↲ **To Change Owner and Group of the File** chown UserName:group1 file1.txt ↲