LINUX USERS AND GROUPS MANAGEMENT

Contents

INTRODUCTION

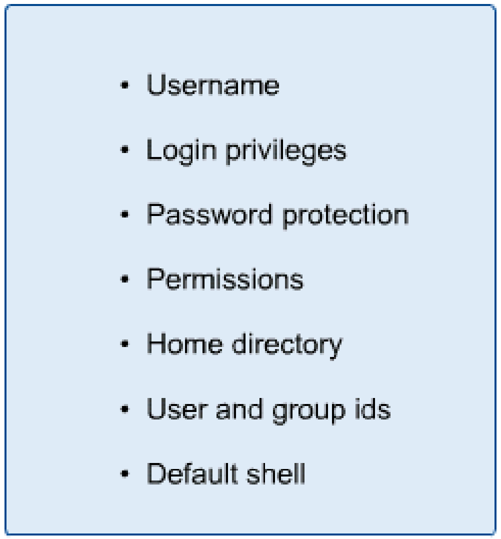
- Linux uses groups to help you manage users, set permissions on those users, and even monitor how much time they are spending in front of the PC.
- Normally Linux computers have two user accounts—your own user account, and the root account, which is the super user that can access everything on the PC, make system changes, and administer other users.
- Ubuntu works a little differently, though—you can't login directly as root by default, and you use the sudo command to switch to root-level access when you need to make a change.

INTRODUCTION

SuperUser or Root User: The administrator of the Linux system who has all the rights. The root account belongs to the superuser. The root user doesn’t need permission to run any command.

System User: The users created by the software or applications. For example, We installed Apache Kafka in the system, then it will create the user account named “Apache”. These are known as System Users created at the time of installing any application.

USER AND GROUP ACCOUNTS IN A LINUX ENVIRONMENT



- Username
- Login privileges
- Password protection
- Permissions
- Home directory
- User and group ids
- Default shell

- User accounts in a Linux system allow several people to be logged into the system at the same time or at different times without interfering with each other.
- The term user and account are sometimes used interchangeably.

USER AND GROUP ACCOUNTS IN A LINUX ENVIRONMENT

- The Linux operating system is both a multiuser and multitasking system.
- The most important user account is the Superuser account; also referred to as the root account.
- This account is used by the system administrator to perform any administrative tasks on a Linux system.
- The Superuser account can be used in several ways:
  - root login
  - Su
  - Sudo

THE ROOT USER

- Introduction

  Root user also called super user, who has access to all resources to which the other user will not have. Root user can modify the ownership and permissions of the files it does not own.

- Who is a Root User?
- Role of Root user in System administration

WHO IS A ROOT USER?

- Root user is a user who has access to all the commands and files in Linux operating system

- The root user is sometimes referred as super user or system administrator or administrative user.

- The root user has full access to all software and hardware on the system.

- The root user's account is stored in the directory '/root'.

ROLE OF THE ROOT USER IN SYSTEM ADMINISTRATION

- System administration designates a job position responsible for running a computer system.

- Linux is a multi-user operating system identified by a user ID and password can make use of the computer system.

- In Linux, the role of the root user in system administration includes:
  - User Management
  - System Maintenance
  - Managing Services
  - Performing Backups

Contd …

ROLE OF ROOT IN USER MANAGEMENT

- The root user has the rights to add or create a new user account into the system.
- The root user has the rights to edit or modify and delete user accounts.
- The root user sets the password aging policy to protect a user account.
- The root user maintains the user accounts in the /home directory
- The root users can modify, delete group accounts.

Contd …

ROLE OF ROOT IN USER MANAGEMENT

- Linux is a complex operating system when it comes to maintenance.
- The root user should develop a test plan for each change in the system like mounting a directory, creating a new group.
- The system administrator should communicate with the users about the changes after doing it or in advance.

Contd …

ROLE OF ROOT USER IN MANAGING SERVICES

- Services can be controlled by GUI (graphical user interface) and CLI (command line interface) in Linux.

- It is preferred to manage services in CLI in Linux.

- The system administrator manages the software installation, upgrade of the software using Redhat Package Manger (RPM).

- The root user configures Network File System, the standard UNIX file sharing protocol.

Contd …

ROLE OF ROOT IN PERFORMING BACKUPS

- Data stored in a system is important because of the time it takes to create it.

- So the system administrators will plan convenient methods to secure the data by taking backups.

- The root user has to identify the important data and allocate space accordingly for storing the data.

USER ADMINISTRATION FILES

- There are four main user administration files –
- **/etc/passwd** – Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- **/etc/shadow** – Holds the encrypted password of the corresponding account. Not all the systems support this file.
- **/etc/group** – This file contains the group information for each account.
- **/etc/gshadow** – This file contains secure group account information.

Check all the above files using the **cat** command.

/ETC/PASSWD FILE

- ***/etc/passwd*** Holds user account info

  Included fields are:
  - Login name
  - User Id (uid)
  - Group Id (gid)
  - General Comment about the user
  - Home Directory
  - Shell

/ETC/SHADOW FILE

- ***/etc/shadow*** Contains the encrypted password information for users' accounts and optionally the password aging information. Included fields are:
    - Login name
    - Encrypted password
    - Days since Jan 1, 1970 that password was last changed
    - Days before password may not be changed
    - Days after which password must be changed
    - Days before password is to expire that user is warned
    - Days after password expires that account is disabled
    - Days since Jan 1, 1970 that account is disabled

SUSPENDING A USER ACCOUNT

- Put a * as start of Password field in /etc/shadow
- Change login shell to /sbin/nologin
- Use GUI to suspend the user

CREATING AND MANAGING USERS AND GROUPS

- Introduction

  User account is a resource with which a person can have access to the system or in other words, accesses a system. A group of users who have the same action to perform are put into one group and a group account is maintained.

- Creating or Adding Users
  - User Private Groups
  - Modifying Users
  - Deleting Users
  - Group Administration
  - Switching Accounts
  - Password Aging Policies

CREATING AND MANAGING USER ACCOUNTS

- Using useradd
- Using passwd
- Using usermod
- Using userdel

CREATING AND ADDING USERS

- New Users can be created or added by using the root privileges. There are three basic methods of adding users in Linux:
  - The /etc/passwd file that contains information about every user in the system can be hand-edited to add a user.
  - Use certain commands like useradd.
  - Use the graphical front-end, the Red Hat User Manager

Contd …

USING COMMANDS

- A new user is created or added by using the command-line utility ***useradd.***
- The ***useradd*** command is located in the ***/usr/sbin*** directory.
- A new user is added into the system by typing:
  - Syntax: # useradd username
  - Example: # useradd jack

MODIFYING USER ACCOUNTS

- User accounts can be modified manually and by using a command.

- Manually, the user details can be edited in ***/etc/passwd*** file or the command, ***usermod*** is used to modify user account details.

  - The syntax is: **usermod [options] username**

DELETING USERS

- The syntax for deleting is userdel username

  For example: userdel sam deletes the user with the username, sam.

- If it is required to remove the user account along with its home directory, then the option '-r' is used.

  - The syntax: **userdel –r username**

    For example: **userdel –r sam** will delete the user account sam along with its home directory.

GROUP ADMINISTRATION

- The file ***/etc/group*** provides group account information.

- New groups may be generated by manually editing the file ***/etc/group*** or by using ***groupadd***.
  - The syntax is: # groupadd groupname

- To delete or remove groups, groupdel is used.
  - The syntax is: # groupdel groupname

- To change the name of the group, groupmod can be used.
  - The syntax is: # groupmod –n newname oldname

MANAGING GROUPS

- Using groupadd
- Using groupmod
- Using groupdel
- groups are defined in the **/etc/group** file. Each record is composed of the following four fields:

  Group:Password:GID:Users

- Group Specifies the name of the group. In the example above, the name of the group is video.
- Password Specifies the group password.

MANAGING GROUPS

- GID Specifies the group ID (GID) number of the group.
- Users Lists the members of the group.
- As with /etc/shadow, each line in /etc/gshadow represents a record for a single group. Each record is composed of the following fields: **Group_Name**:**Password**:Group_Admins:Group_ Members

USING GROUPDEL

Syntax: **groupdel** ***group_name***

example: **groupdel** ***student***

MANAGING OWNERSHIP

- Anytime a user creates a new file or directory, his or her user account is assigned as that file or directory's "owner."
- For example, suppose the ken user logs in to her Linux system and creates a file named linux_introduction.odt using OpenOffice.org in home directory. Because she created this file, ken is automatically assigned ownership of linux_introduction.odt.

HOW OWNERSHIP WORKS

- You can specify a different user and/or group as the owner of a given file or directory. To change the user who owns a file, you must be logged in as root. To change the group that owns a file, you must be logged in as root or as the user who currently owns the file.
  - Using **chown**
  - Using **chgrp**
  - You can also view file ownership from the command line using the **ls –l** command

USING CHOWN

- The chown utility can be used to change the **user** or **group** that owns a file or **directory**.

  Syntax **chown** user.group file or directory.

  Example: If I wanted to change the file's owner to the **ken1** user, I would enter

  chown ken1 /tmp/myfile.txt

- If I wanted to change this to the users group, of which **users** is a member, I would enter

  chown .users /tmp/myfile.txt

  Notice that I used a period **(.)** before the group name to tell chown that the entity specified is a group, not a user account.

  Ex: **chown** student.users /tmp/myfile.txt

  Note: You can use the –R option with chown to change ownership on many files at once recursively.

USING CHGRP

- In addition to chown, you can also use **chgrp** to change the group that owns a file or directory.
- Syntax: **chgrp group file (or directory)**
- Example: **chgrp student /tmp/newfile.txt.**

SWITCHING ACCOUNTS

- In Linux, to change user without logging out, the command ***su*** (derived from super user) is used.

- The syntax for switching accounts using su command is:

  su [options] username

- System administrators use this command frequently to become the root user without the user logging out.

- When the account is switched, the password of the account has to be given unless you are logged in as root.

PASSWORD AGING POLICIES

- Password aging means that after certain time, generally 90 days, the user will be asked to replace his password with a new one.

- As a method to secure passwords, password aging technique is used by system administrators .

- Password aging prevents an intruder using the cracked password only for a certain amount of time.

- There are two methods to set password aging.

- One way is to use the command ***chage*** and the other using the graphical application User Manager.

USING COMMAND CHAGE

- First method is the use of chage command
- The maximum period of time-of-validity of a password can be set before the system prompts the user to change his password.
- The time given for password changes is usually guided by the security policy of an organization.
- The minimum period of time also has to be set for a password to be in use.
- The syntax is # chage [options] username

CONFIGURING SUDO

- Introduction
  - **sudo** (superuser do) permits a system administrator to grant certain users (or groups of users) the permission to run some (or all) commands as root while logging all commands and arguments..
  - Sudo configuration
  - Using sudo

SUDO CONFIGURATION

- The command, ***sudo*** permits a user with proper permissions to execute a command as the superuser or other user.

- The file ***/etc/sudoers*** controls the ***sudo*** access.

- An editor and syntax checker, ***visudo*** edits the file ***/etc/sudoers***.

- ***sudo*** is easy to configure and is used in a straightforward syntax.

- ***sudo*** commands use a basic syntax.

SUDO

- The ***sudo*** utility helps in overcoming this difficulty faced while managing a server or system by several people
- The ***sudo*** utility permit users defined in the ***/etc/sudoers*** configuration file to have temporary access
- The commands can be run as ***user*** "***root***" or as any other ***user*** as in the ***/etc/sudoers*** configuration file.
- The privileged command to be run should first start with the word ***sudo*** followed by the command's regular syntax.

SU

- Short for *substitute* or *switch user*
- Syntax: `su [options] [username]`
  - If `username` is omitted, `root` is assumed
- After issuing command, prompted for that user's password
- A new shell opened with the privileges of that user
- Once done issuing commands, must type exit

SUDO

- Allows you to issue a single command as another user
- Syntax:
  ```
  sudo [options] [-u user] command
  ```
- Again, if no user specified, root assumed
- New shell opened with user's privileges
- Specified command executed
- Shell exited

SUDOERS

- Must configure a user to run commands as another user when using `sudo`
- Permissions stored in `/etc/sudoers`
- Use utility `visudo` to edit this file (run as root)
- Permissions granted to users or groups, to certain commands or all, and with or without password being required