ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

# LINUX AND OPEN SOURCE SOFTWARE

**Ubuntu Linux Unleashed 2021 Edition (Matthew Helmke) (z-lib.org)**

# CHAPTER 4

# MANAGING USERS

# Contents

- User Accounts
- Managing Groups
- Managing Users
- Managing Passwords
- Granting System Administrator Privileges to Regular Users
- Disk Quotas
- Related Ununtu Commands

# MANAGING USERS

User management and administration includes allocating and **managing /home directories**, putting in place good **password policies**, and applying effective security policies that include things such as disk quotas and file and directory **access permissions**.

- This chapter covers all these areas as well as some of the different types of users that you are likely to find on a typical Linux system.

# MANAGING USERS

## User Accounts

You normally find three types of users on Linux systems:

- *super user, the day-to-day user, and the system user.*

# MANAGING USERS

## User Accounts

All users who access your system must have accounts on the system. Ubuntu uses the /etc/passwd file to store information on the user accounts that are present on the system. All users, regardless of their type, have a one-line entry in this file that contains:

- Username (typically used for logging in to the system),
- Password an encrypted field (which contains an X to indicate that a password is present),
- User ID (commonly referred to as the UID), and a group ID (GID).

# MANAGING USERS

## User Accounts

- /home directory (usually /home/username)
- default shell for the user (/bin/bash is the default for new uers).
- GECOS field that uses a comma-delimited list to record information about the account or the user; most often when this field is used, it records the user's full name and contact information.

# MANAGING USERS

## User Accounts

**NOTE**

Although the Password field contains an *X*, this doesn't mean that what you read here is the actual password. All passwords are stored in `/etc/shadow` in an encrypted format for safekeeping. Ubuntu automatically refers to this file whenever a password is required. You can read more about this later in the chapter, in the "Shadow Passwords" section.

# MANAGING USERS

## User Accounts

- In keeping with long-standing tradition in UNIX-style operating systems, Ubuntu makes use of the well-established UNIX file ownership and permission system.
- To start with, everything in these systems is treated as a file, and all files (e.g., directories and devices) can be assigned one or more read, write, and execute permissions. These three “flags” can also be assigned as desired to each of three categories:
  - *The owner of the file, a member of a group, or anyone else on the system.*

# MANAGING USERS

## User Accounts

- The security for a file is drawn from these permissions and from file ownership. As the system administrator (also commonly referred to as the super user), it is your responsibility to manage these settings effectively and ensure that the users have proper UIDs and GIDs.
- The system administrator can use these file permissions to lock away sensitive files from users who should not have access to them.

# MANAGING USERS

## The Super User/Root User

- No matter how many system administrators there are for a system, there can be **only one super user account**. The super user account, more commonly referred to as the root user, has total and complete control over all aspects of the system.
- That account can access any part of the file system; read, change, or delete any file; grant and revoke access to files and directories; and carry out any operation on the system, including destroying it.
- The root user is unique in that it has a UID of 0 and GID of 0.

# MANAGING USERS

## The Super User/Root User

In Ubuntu, you execute a command with root, or super user, privileges by using the sudo command, like this:

matthew@seymour:~$ **sudo apt-get update**

You are then prompted for your password, typing in your password, press Enter. Ubuntu then carries out the command as if you were running it as root.

# MANAGING USERS

## The Super User/Root User

### THE ROOT USER

If you've used other Linux distros, you might be a little puzzled by the use of the `sudo` command because not all distros use it. In short, Ubuntu allows the first user on the system access to full root privileges using the `sudo` command. It also disables the root account so that no one can actually log in with the username *root*.

In other Linux distros, you change to the root user by issuing the command `su -` and then entering the root password when prompted. This lands you at the root prompt, which is shown as a pound sign (#). From here, you can execute any command you want. To get to a root prompt in Ubuntu, you need to execute the command `sudo -i`, and after you enter your password, you get the prompt so familiar to other Linux distros. When you've finished working as root, type `exit` and press Enter to get back to a normal user prompt (`$`).

# MANAGING USERS

## User IDs and Group IDs

A computer is, by its very nature, a number-oriented machine. It identifies users and groups by numbers known as the user ID (UID) and group ID (GID). The alphabetic names display on your screen just for ease of use.

- Root user is UID 0.
- System users: Numbers from 1 through 499 and number 65,534, sometimes called logical users, or pseudo-users.
- Regular users have UIDs beginning with 1,000; Ubuntu assigns them sequentially, beginning with this number.

# MANAGING USERS

## File Permissions

There are three types of permissions: read, write, and execute (r, w, x). For any file or directory, permissions are assigned to three categories: user, group, and other. Some commands used to change the group, user, or access permissions of a file or directory:

- **chgrp**—Changes the group ownership of a file or directory
- **chown**—Changes the owner of a file or directory
- **chmod**—Changes the access permissions of a file or directory

# MANAGING USERS

## Managing Groups

- Groups can make managing users a lot easier. Instead of having to assign individual permissions to every user, you can use groups to grant or revoke permissions to a large number of users quickly and easily.
- Setting group permissions enables you to set up workspaces for collaborative working and to control what devices can be used, such as external drives or DVD writers. This approach also represents a secure method of limiting access to system resources.

# MANAGING USERS

## Managing Groups

**Example:** The system administrator could put the users *matthew*, *ryan*, *sandra*, *holly*, *debra*, and *mark* in a new group named *unleashed*. Those users could each create files intended for their group work. Then ***chgrp*** those files to *unleashed*.

- Everyone in the *unleashed* group and only *root* can work with those files.
- The system administrator would create a directory owned by that group so that its members could have an easily accessible place to store those files. The system administrator could also add other users such as *chris* and *shannon* to the group and remove existing users when their part of the work is done. The system administrator could make the user *matthew* the group administrator so that *matthew* could decide how group membership should be changed.

# MANAGING USERS

## Group Listing

- Different UNIX operating systems implement the group concept in various ways. Ubuntu uses a scheme called UPG (user private group) in which the default is that each user is assigned to a group with his or her own name. (The user's username and group name are identical.)
- All the groups on a system are listed in **/etc/group** file.

```
matthew@seymour:~$ cat /etc/group
root:x:0:
daemon:x:1:
admin:x:115:matthew
saned:x:116:
gdm:x:119:
matthew:x:1000:
```

# MANAGING USERS

## Group Listing

**FINDING YOUR GROUPS**

You can find which groups your user account belongs to by using the groups command, like this:

```
matthew@seymour:~$ groups

matthew adm cdrom sudo audio dip plugdev lpadmin sambashare
```

Add a username after the command to list the groups for that user.

```
minh@minh-VM:~$ groups
minh adm cdrom sudo dip plugdev lpadmin lxd sambashare
minh@minh-VM:~$ groups minh
minh : minh adm cdrom sudo dip plugdev lpadmin lxd sambashare
minh@minh-VM:~$ groups root
root : root
```

# MANAGING USERS

## Group Management Tools

- Ubuntu provides several command-line tools for managing groups, and it also provides graphical tools for doing so.
- Most experienced system administrators prefer the command-line tools because they are **quick and easy to use**, they are always available (even when there is **no graphical user interface**),
- Commands can be **included in scripts** that system administrators may want to write to perform repetitive tasks.

# MANAGING USERS

## Group Management Tools

The most commonly used group management command-line tools:

- **groupadd**— Creates and adds a new group.
- **groupdel**— Removes an existing group.
- **groupmod**— Creates a group name or GIDs but doesn't add or delete members from a group.
- **gpasswd**— Creates a group password. Every group can have a group password and an administrator. Use the -A argument to assign a user as group administrator.
- **useradd -G**—The -G argument adds a user to a group during the initial user creation.
- **usermod -G**— Allows you to add a user to a group.
- **grpck**—This command checks the /etc/group file for typos.

# MANAGING USERS

## Group Management Tools

Let's say there is a DVD-RW device (/dev/scd0) on your computer that the system administrator wants a regular user named ryan to have permission to access.

*1. Add a new group with the groupadd command:*

matthew@seymour:**~$ sudo groupadd dvdrw**

*2. Change the group ownership of the device to the new group with the **chgrp** command:*

matthew@seymour:**~$ sudo chgrp dvdrw /dev/scd0**

*3. Add the approved user to the group with the **usermod** command:*

matthew@seymour:**~$ sudo usermod -G dvdrw ryan**

*4. Make user ryan the group administrator with the **gpasswd** command:*

matthew@seymour:**~$ sudo gpasswd -A ryan**

- Now ryan has permission to use the DVD-RW drive, as would anyone else added to the group by either the super user or ryan (a group administrator).

# MANAGING USERS

- A new user must be created, assigned a UID, provided a **/home** directory, provided an initial set of files for his or her **/home** directory, and assigned to groups in order to use the system resources securely and efficiently.
- The system administrator in some situations might want or need to restrict not only a user's access to specific files and folders but also the amount of disk space an account may use.

# MANAGING USERS

## User Management Tools

Most common commands to manage users:

▸▸ **useradd**— Adds a new user account. Its options permit the system administrator to specify the user's **/home** directory and initial group or to create the user with the default **/home** directory and group assignments.

▸▸ **useradd -D**—Sets the system defaults for creating the user's **/home** directory, account expiration date, default group, and command shell. Used without any arguments, the **useradd -D** command displays the defaults for the system. The default files for a user are in **/etc/skel**.

```
minh@minh-VM:~$ useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
```

# MANAGING USERS

## User Management Tools

**NOTE**

The set of files initially used to populate a new user's home directory is kept in `/etc/skel`. This is convenient for the system administrator because any special files, links, or directories that need to be universally applied can be placed in `/etc/skel` and will be duplicated automatically with appropriate permissions for each new user:

```
minh@minh-VM:~$ ls -al /etc/skel/
total 28
drwxr-xr-x   2 root root  4096 Thg 8  21 14:40 .
drwxr-xr-x 131 root root 12288 Thg 9   7 18:47 ..
-rw-r--r--   1 root root   220 Thg 2  25  2020 .bash_logout
-rw-r--r--   1 root root  3771 Thg 2  25  2020 .bashrc
-rw-r--r--   1 root root   807 Thg 2  25  2020 .profile
```

# MANAGING USERS

## Adding New Users

**NOTE**

A Linux username can be any alphanumeric combination that does not begin with a special character reserved for shell script use (mostly <space> and punctuation characters; see Chapter 14, "Automating Tasks and Shell Scripting," for disallowed characters). A username is often the user's first name plus the first initial of her last name or the first initial of the user's first name and his entire last name. These are common practices on larger systems with many users because it makes life simpler for the system administrator, but neither convention is a rule or a requirement.

# MANAGING USERS

## Adding New Users

matthew@seymour**:~$ sudo useradd sandra -p c00kieZ4ME -u 1042**

This example uses the *-p* option to set the password the user requested and the *-u* option to specify her UID. (If you create a user with the default settings, you do not need to use these options.)

The system administrator can also use the graphical interface that Ubuntu provides to add the same account (Page 209).

# MANAGING USERS

## User Management Tools

▸▸ **deluser**— Removes a user's. There is an older version of this command, *userdel*, that previous versions of this book discussed. *deluser* is preferred because it provides finer control over what is deleted. Whereas userdel automatically removes both the user account and also all the user's files, such as the associated **/home** directory, *deluser* deletes only the user account, unless you use a command-line option to tell it to do more. *deluser* includes options such as *--remove-home*, *--removeall- files*, *--backup*.

▸▸ **passwd**—This command updates the authentication tokens used by the password management system.

# MANAGING USERS

## User Management Tools

**TIP**

To lock a user out of his or her account, use the following command:

```
matthew@seymour:~$ sudo passwd -l username
```

This prepends an `!` (exclamation point, also called a bang) to the user's encrypted password; the command to reverse the process uses the `-u` option.

# MANAGING USERS

## User Management Tools

▸▸ **usermod**— Changes several user attributes. The most commonly used arguments are *-s* to change the shell and *-u* to change the UID. No changes can be made while the user is logged in or running a process.

▸▸ **chsh**— Changes the user's default shell. For Ubuntu, the default shell is */bin/bash*, known as the Bash, or Bourne Again Shell.

# MANAGING USERS

## Monitoring User Activity on the System

- Monitoring user activity is part of a system administrator's duties and an essential task in tracking how system resources are being used. The *w* command tells the system administrator who is logged in, where he is logged in, and what he is doing. No one can hide from the super user. The *w* command can be followed by a specific user's name to show only that user.
- The *ac* command provides information about the total connect time of a user, measured in hours. It accesses the */var/log/wtmp* file for the source of its information. The *ac* command is most useful in shell scripts to generate reports on operating system usage for management review. Note that to use the *ac* command, you must install the *acct* package from the Ubuntu repositories.

# MANAGING USERS

## Monitoring User Activity on the System

**NOTE**

The accounting system on your computer keeps track of user usage statistics and is kept in the current `/var/log/wtmp` file. That file is managed by the `systemd` processes. If you want to explore the depths of the accounting system, use the GNU info system: `info accounting`.

# MANAGING USERS

## Managing Passwords

- Passwords are an integral part of Linux security, and they are the most visible part to the user. In this section, you learn how to establish a minimal password policy for your system, where the passwords are stored, and how to manage passwords for your users.

**System Password Policy:**

- ▸▸ Allowed and forbidden passwords
- ▸▸ Frequency of mandated password changes
- ▸▸ Retrieval or replacement of lost or forgotten passwords
- ▸▸ Password handling by users

# MANAGING USERS

## Managing Passwords

**The Password File:**

The password file is */etc/passwd*, and it is the database file for all users on the system. The format of each line is as follows:

```
username:password:uid:gid:gecos:homedir:shell
```

Note that colons separate all fields in the /etc/passwd file. If no information is available for a field, that field is empty, but all the colons remain.

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

- Several Keeping passwords in */etc/passwd* is considered a security risk because anyone with read access could run a cracking program on the file and obtain the passwords with little trouble.
- To avoid this risk, *shadow passwords* are used so that only an X appears in the password field of */etc/passwd*; the real passwords are kept in */etc/shadow*, a file that can be read only by the system administrator (and PAM, the *Pluggable Authentication Modules* authentication manager; see the “PAM Explained” sidebar, later in this chapter, for an explanation of PAM).

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

Several services run as pseudo-users, usually with root permissions. These are the system, or logical, users mentioned previously. You would not want these accounts to be available for general login for security reasons, so they are assigned ***/sbin/nologin*** or ***/bin/false*** as their shell, which prohibits any logins from these accounts.

A list of ***/etc/passwd*** reveals the following (abridged for brevity):

```
matthew@seymour:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
messagebus:x:102:106::/var/run/dbus:/bin/false
avahi:x:105:111:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
couchdb:x:106:113:CouchDB Administrator,,,:/var/lib/couchdb:/bin/bash
haldaemon:x:107:114:Hardware abstraction layer,,,:/var/run/hald:/bin/false
kernoops:x:109:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
gdm:x:112:119:Gnome Display Manager:/var/lib/gdm:/bin/false
matthew:x:1000:1000:Matthew Helmke,,,,:/home/matthew:/bin/bash
sshd:x:114:65534::/var/run/sshd:/usr/sbin/nologin
ntp:x:115:122::/home/ntp:/bin/false
pulse:x:111:117:PulseAudio daemon,,,:/var/run/pulse:/bin/false
```

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

The fields are separated by colons and are, in order:

- The user's login name.
- The encrypted password for the user.
- The day on which the last password change occurred, measured in the number of days since January 1, 1970. This date is known in UNIX circles as the epoch. Just so you know, the billionth second since the epoch occurred was in September 2001; that was the UNIX version of Y2K—and as with the real Y2K, nothing much happened.
- The number of days before the password can be changed (which prevents changing a password and then changing it back to the old password right away—a dangerous security practice).

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

The fields are separated by colons and are, in order (cont.):

- The number of days after which the password must be changed. This can be set to force the change of a newly issued password known to the system administrator.
- The number of days before the password expiration that the user is warned it will expire.
- The number of days after the password expires that the account is disabled (for security).
- Similar to the password change date, although this is the number of days since January 1, 1970, that the account has been disabled.
- A “reserved” field that is not currently allocated for any use.

Note

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

$sudo cat /etc/shadow

Each line of the `/etc/shadow` file contains nine comma-separated fields:

```
mark:$6$.n.:17736:0:99999:7:::
[--] [----] [---] - [---] ----
|      |      |   |   |   |||+----------> 9. Unused
|      |      |   |   |   ||+-----------> 8. Expiration date
|      |      |   |   |   |+------------> 7. Inactivity period
|      |      |   |   |   +-------------> 6. Warning period
|      |      |   |   +-----------------> 5. Maximum password age
|      |      |   +---------------------> 4. Minimum password age
|      |      +-------------------------> 3. Last password change
|      +--------------------------------> 2. Encrypted Password
+---------------------------------------> 1. Username
```

**Encrypted Password:** The password is using the $type$salt$hashed format.
$type is the method cryptographic hash following values:

- $1$ – MD5
- $2a$ – Blowfish
- $2y$ – Eksblowfish
- $5$ – SHA-256
- $6$ – SHA-512

# MANAGING USERS

## Managing Passwords

### Shadow Passwords

- Note that password expiration dates and warnings are disabled by default in Ubuntu. These features are not often used on home systems and usually are not even used for small offices.
- It is the system administrator's responsibility to establish and enforce password expiration policies if they are to exist.
- The permissions on the */etc/shadow* file should be set so that it is not writable or readable by regular users: The permissions should be *600*.

# MANAGING USERS

## Managing Passwords

### PAM EXPLAINED

*Pluggable Authentication Modules (PAM)* is a system of libraries that handle the tasks of authentication on a computer. It uses four management groups: account management, authentication management, password management, and session management. This allows the system administrator to choose how individual applications will authenticate users. Ubuntu has preinstalled and preconfigured all the necessary PAM files for you.

The configuration files in Ubuntu are in `/etc/pam.d`. Each of these files is named for the service it controls, using the following format:

```
type control module-path module-arguments
```

The `type` field is the management group that the rule corresponds to. The `control` field tells PAM what to do if authentication fails. The final two items deal with the PAM module used and any arguments it needs. Programs that use PAM typically come packaged with appropriate entries for the `/etc/pam.d` directory. To achieve greater security, the system administrator can modify the default entries. Misconfiguration can have unpredictable results, so back up the configuration files before you modify them. The defaults provided by Ubuntu are adequate for home and small office users.

# MANAGING USERS

## Managing Password Security for Users

- Selecting appropriate user passwords is always an exercise in trade-offs. A password such as *password* (do not laugh, it has been used often in the real world and with devastating consequences) is just too easy to guess by an intruder. So are simple words or number combinations (the numbers from a street address or date of birth, for example). You would be surprised how many people use easily guessed passwords such as *123456*, *iloveyou*, *Qwerty*, and *abc123*.
- In contrast, a password such as *2a56u'"F($84u&#^Hiu44Ik%$([#EJD* is sure to present great difficulty to an intruder (or an auditor). However, that password is so difficult to remember.

# MANAGING USERS

## Managing Passwords

**Changing Passwords in a Batch**

On a large system, there might be times when a large number of users and their passwords need some attention. The super user can change passwords in a batch by using the *chpasswd* command, which accepts input as a name/password pair per line in the following form:

matthew@seymour:~$ **sudo chpasswd username:password**

# MANAGING USERS

## Managing Passwords

*Temporarily Changing User Identity with the **su** Command*

A popular misconception is that the su command is short for *super user*; it really just means *substitute user*. An important but often overlooked distinction is that between *su* and *su -*. In the former instance, you become that user but keep your own environmental variables (such as paths). In the latter, you inherit the environment of that user. This is most noticeable when you use su to become the super user, root. Without appending the -, you do not inherit the path variable that includes */bin* or */sbin*, so you must always enter the full path to those commands when you just *su* to root.

# MANAGING USERS

## Managing Passwords

The **su** command spawns a new shell, changing both the UID and GID of the existing user and automatically changing the environmental variables associated with that user, known as *inheriting the environment*. The syntax for the ***su*** command is as follows:

*matthew@seymour:~$* ***su option username arguments***

The man page for su gives more details. To return to the regular user's identity, just type the following:

*root~#* ***exit***

This takes you to the regular user's prompt:

*matthew@seymour:~$*

# MANAGING USERS

## Managing Passwords

**NOTE**

The `su` command is often seen as bad because what it is supposed to do is a bit ambiguous. On one hand, it is supposed to open a new session and change a number of execution parameters while also inheriting parameters from the session in which it was issued. It does give you a new execution shell, but that is not really the same thing as a full login. `systemd` has added a new command, `machinectl shell`, that is intended to do this "properly," according to its creators. Because `systemd` is covered in Chapter 15, "The Boot Process," this new command is also covered there.

# MANAGING USERS

## Managing Passwords

- Granting Root Privileges on Occasion: *The **sudo** Command*

- If you want to get to a root shell, thereby removing the need to type **sudo** for every command, just enter ***sudo -i*** to get the root prompt. To return to a normal user prompt, enter ***exit***, and press Enter.
  - An authorized user merely precedes a super user authority–needed command with **sudo**, like this:
    *matthew@seymour:~$* ***sudo command***
- When the command is entered, ***sudo*** checks the ***/etc/sudoers*** file to see whether the user is authorized to wield super user privileges;
- The ***sudo –l*** command presents users with a list of the commands they are entitled to use: *matthew@seymour:~$* ***sudo –l***

# MANAGING USERS

## Managing Passwords

Three man pages are associated with sudo: *sudo*, *sudoers*, and *visudo*. The first covers the command itself, the second the format of the */etc/sudoers* file, and the third the use of the special editor for */etc/sudoers*. You should use the special editing command because it checks the file for parse errors and locks the file to prevent others from editing it at the same time.

The *visudo* command uses the *vi* editor, so you might need a quick review of the *vi* editing commands. Begin the editing */etc/sudoers* by executing the *visudo* command:

*matthew@seymour:~$* ***sudo visudo***

# MANAGING USERS

## Managing Passwords

- The basic format of a sudoers line in the file is as follows:

  *user host_computer=command*

- The user can be an individual user or a group. (A % in front identifies a name as a group.)
- The *host_computer* is normally ALL for all hosts on the network and *localhost* for the local machine, but the host computer can be referenced as a subnet or any specific host.
- The command in the *sudoers* line can be ALL, a list of specific commands, or a restriction on specific commands (formed by prepending a ! to the command). A number of options are available for use with the *sudoers* line, and aliases can be used to simplify the assignment of privileges. Again, the *sudoers* man page gives the details, but let's look at a few examples.

# MANAGING USERS

## Managing Passwords

Suppose that you want to give user john permission across the network to be able to add users with the graphical interface. You would add the following line:

```
john   ALL=/users-admin
```

Or perhaps you would grant permission only on the user's local computer:

```
john   192.168.1.87=/usr/bin/users-admin
```

If you want to give the editor group system-wide permission with no password required to delete files, you use the following:

```
%editors ALL=NOPASSWD: /bin/rm
```

If you want to give every user permission with no password required to mount the CD drive on the localhost, you do so as follows:

```
ALL localhost=NOPASSWD:/sbin/mount /dev/scd0 /mnt/cdrom /sbin/umount /mnt/cdrom
```

# Useful Commands to manage user accounts

- **ac**—Provides user account statistics
- **change**—Sets or modifies user password expiration policies
- **chfn**—Creates or modifies user finger information in /etc/passwd
- **chgrp**—Modifies group memberships
- **chmod**—Changes file permissions
- **chown**—Changes file ownerships
- **chpasswd**—Modifies user passwords in batches
- **chsh**—Modifies a user's shell
- **groups**—Displays existing group memberships
- **logname**—Displays a user's login name
- **newusers**—Batches user management command
- **passwd**—Creates or modifies user passwords
- **su**—Executes a shell or command as another user
- **sudo**—Manages selected user execution permissions
- **useradd**—Creates, modifies, or manages users
- **usermod**—Edits a user's login profile

# Q&A