**­**When a user is created in a Linux system, several actions occur to ensure the user has access and permissions to work within the system. Here’s a breakdown of what happens:

**1. Creation of an Entry in /etc/passwd:**

* An entry is added to the /etc/passwd file for the new user.
* This entry contains details like:
  + Username
  + User ID (UID)
  + Group ID (GID)
  + Home directory path
  + Default shell
  + Full name or description (optional)

Example of a line in /etc/passwd:

username:x:1001:1001::/home/username:/bin/

**2. Creation of an Entry in /etc/shadow:**

* A corresponding entry is made in the /etc/shadow file to store encrypted password information and password policies for the user.
* Fields include:
  + Username
  + Encrypted password
  + Last password change date
  + Password expiration details

Example of a line in /etc/shadow:

username:$6$hash...$encrypted\_password:19250:0:99999:7:::

**3. Default Group Membership:**

* A group is created for the user with the same name as the username (if configured to do so, which is common in many distributions).
* The group details are stored in /etc/group.  
  Example:

username:x:1001:

**4. Creation of a Home Directory:**

* If specified (or by default in many cases), a home directory is created at /home/username.
* The contents of /etc/skel/ (a skeleton directory) are copied into the home directory. This often includes default configuration files like .rc or .profile.

**5. Setting Ownership and Permissions:**

* The home directory and its contents are assigned ownership to the new user and their default group.
* Example:

chown username:username /home/username

chmod 700 /home/username

**6. Assignment of Default Shell:**

* The user is assigned a default shell, such as /bin/ or /bin/sh. This is specified during the creation or defaults to a system-wide setting.

**7. Logging and Audit:**

* Some systems log the creation of users in files like /var/log/auth.log or /var/log/secure.

**8. Optional Password Setup:**

* If no password is provided during creation, the account may be locked initially.
* To activate it, a password must be set using commands like passwd username.

**9. (Optional) Additional Configurations:**

* The user may be added to supplementary groups for additional permissions, like sudo for administrative rights:

usermod -aG sudo username

* Environment variables and custom shell configurations can be set up for the user.

**Key Command:**

To create a user, the most common command is:

sudo useradd -m -s /bin/ username

or, for interactive creation:

sudo adduser username

This process ensures that the user is properly configured to interact with the Linux system securely and efficiently.

In Linux and other Unix-like operating systems, **permissions** and **ownership** are mechanisms to control access to files and directories. They determine **who can access a file or directory** and **what actions they can perform** on it. Here's a detailed explanation:

### ****1. Ownership****

Each file and directory in Linux has an **owner** and an associated **group**.

#### **Types of Ownership**

1. **User (Owner):**
   * The user who owns the file or directory.
   * Typically, the creator of a file becomes its owner.
   * Ownership can be changed with chown command.
2. **Group:**
   * A group of users who have specific permissions to the file.
   * Multiple users in the same group can share access to files and directories based on group permissions.
   * Group ownership can also be changed using the chgrp command.
3. **Others:**
   * All other users on the system who are neither the owner nor part of the group.

### ****2. Permissions****

Permissions define **what actions are allowed** on a file or directory for each ownership category (User, Group, Others).

#### **Types of Permissions**

1. **Read (r):**
   * Files: Allows viewing the file's content.
   * Directories: Allows listing the contents of the directory.
2. **Write (w):**
   * Files: Allows modifying the file's content.
   * Directories: Allows creating, deleting, or renaming files within the directory.
3. **Execute (x):**
   * Files: Allows executing the file (if it’s a script or binary).
   * Directories: Allows entering the directory (cd command) and accessing its contents.

### ****Permission Representation****

Permissions are represented in two ways:

1. **Symbolic Representation:**

rwxr-xr--

* + First character: Indicates type (- for file, d for directory, etc.).
  + Next three characters (rwx): Permissions for the owner.
  + Next three characters (r-x): Permissions for the group.
  + Last three characters (r--): Permissions for others.

1. **Numeric Representation:**  
   Permissions can also be represented as an octal number using chmod:
   * 4: Read (r)
   * 2: Write (w)
   * 1: Execute (x)

Examples:

* + rwxr-xr-- → 755
  + rw-r--r-- → 644

### ****How Ownership and Permissions Work Together****

When a user attempts to access a file or directory, the system checks:

1. If the user is the **owner**, and applies the corresponding permissions.
2. If not, whether the user is in the **group** associated with the file, and applies the group permissions.
3. If neither, the **others** permissions are applied.

### ****Key Commands****

* **View Ownership and Permissions:**

ls -l

Example output:

-rw-r--r-- 1 username groupname 1024 Nov 16 15:00 example.txt

* + -rw-r--r--: Permissions
  + username: Owner
  + groupname: Group
* **Change Ownership:**

chown newowner filename

Change both owner and group:

chown newowner:newgroup filename

* **Change Group Ownership:**

chgrp newgroup filename

* **Change Permissions:**  
  Symbolic:

chmod u+r filename # Add read permission for the user

chmod g-w filename # Remove write permission for the group

chmod o+x filename # Add execute permission for others

Numeric:

chmod 755 filename

### ****Example****

1. A file with rwxr--r-- permissions:
   * Owner: Can read, write, and execute.
   * Group: Can only read.
   * Others: Can only read.
2. A directory with drwxr-xr-x permissions:
   * Owner: Full access (list, modify, and access contents).
   * Group: Can list and access contents but cannot modify.
   * Others: Same as group.

By combining **ownership** and **permissions**, Linux provides a robust security model to ensure that users can only access or modify files and directories as intended.

The /etc/sudoers file in Linux is a configuration file that determines **who can use the sudo command** and what specific commands they are allowed to execute with elevated (root) privileges. This file is a critical part of system security and must be edited carefully.

### ****Purpose of**** /etc/sudoers

* To control access to privileged commands and tasks.
* To define fine-grained permissions for users or groups.
* To maintain security by restricting access to administrative operations.

### ****Viewing the**** /etc/sudoers ****File****

The /etc/sudoers file should not be edited directly using a standard text editor because syntax errors can cause system issues. Instead, use the visudo command, which checks for syntax errors before saving changes.

To open the file safely:

sudo visudo

### ****Basic Structure of the**** /etc/sudoers ****File****

Here’s an example of typical entries in /etc/sudoers:

#### 1. **Defaults Settings**

These define global settings for sudo behavior. For example:

Defaults env\_reset

Defaults mail\_badpass

Defaults secure\_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

* env\_reset: Resets the environment for security.
* secure\_path: Ensures sudo uses a secure PATH variable.

#### 2. **User Privilege Specification**

Specifies permissions for individual users. Example:

root ALL=(ALL:ALL) ALL

* root: The username.
* ALL: Can use sudo from any host.
* (ALL:ALL): Can run commands as any user and any group.
* ALL: Can run all commands.

#### 3. **Group Privilege Specification**

Specifies permissions for user groups. Example:

%admin ALL=(ALL) ALL

* %admin: The group name (preceded by % to indicate a group).
* ALL=(ALL) ALL: Members of this group can execute all commands.

#### 4. **Command Restrictions**

You can restrict which commands a user can execute. Example:

johndoe ALL=(ALL) /usr/bin/systemctl

* The user johndoe can only run the systemctl command with sudo.

#### 5. **Alias Definitions**

Allows grouping of users, hosts, or commands for simplified management. Example:

Cmnd\_Alias SHUTDOWN\_CMDS = /sbin/shutdown, /sbin/reboot

johndoe ALL=(ALL) NOPASSWD: SHUTDOWN\_CMDS

* Cmnd\_Alias: Defines a command alias.
* NOPASSWD: Allows the user to run the commands without entering their password.

### ****Important Directives****

1. **ALL**: Represents all users, hosts, or commands.
2. **NOPASSWD**: Allows running commands without a password prompt.
3. **NOEXEC**: Prevents the execution of commands within a script.
4. **Defaults**: Customizes sudo behavior (e.g., logging, environment).

### ****Best Practices for**** /etc/sudoers

1. **Always use visudo**: It checks syntax before saving.
2. **Avoid giving blanket permissions**: Be specific about commands and users.
3. **Use groups**: Assign permissions to groups instead of individual users.
4. **Minimize NOPASSWD usage**: Require passwords for sensitive operations.
5. **Back up the file**: Before making changes, back up the file for recovery.

### ****Example Use Cases****

#### Grant Full Sudo Access to a User:

johndoe ALL=(ALL) ALL

#### Allow a User to Restart Services Without a Password:

johndoe ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart

#### Restrict a User to Specific Commands:

johndoe ALL=(ALL) /usr/bin/apt-get, /usr/bin/systemctl

By properly configuring /etc/sudoers, you can maintain system security while granting necessary administrative privileges to users and groups.

In Linux, **creating a user without a name** (often referred to as a "no-name user") isn't straightforward because standard user creation commands like useradd require a username. However, you can create **system users** or utilize a UID (User ID) directly without associating it with a standard username. Here's a detailed explanation:

### ****1. Can We Create a User Without a Name?****

* **Directly No:** Regular tools like useradd and adduser enforce the use of a valid username.
* **Indirectly Yes:**
  + You can create system entries for UIDs in /etc/passwd without assigning a conventional username.
  + Such entries typically have a blank username field or rely solely on their UID.

#### Example:

Modify /etc/passwd to create an entry like:

ruby

:x:1002:1002::/nonexistent:/usr/sbin/nologin

Here:

* The username field is empty (between the first : and the second :).
* The UID 1002 can still be used for processes or file ownership.

### ****2. What Are "No-Name Users" Used For?****

1. **System Processes and Services:**
   * Some services use user accounts purely for **privilege separation** or **process isolation**, with no need for a conventional username.
   * Such accounts typically do not allow login and exist solely to assign ownership of processes or files.

Example:

ruby

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

* + The nobody user is a well-known system user with minimal privileges, often used as a placeholder.

1. **Enhanced Security:**
   * "No-name" or pseudo-users can reduce attack vectors by restricting access.
   * They are configured with nologin or /dev/null as their shell, preventing interactive login.
2. **Legacy Systems:**
   * Older systems or custom implementations may have had users represented only by UIDs.
3. **Testing and Debugging:**
   * In specialized setups, admins may experiment with such users to mimic unique scenarios for testing.

### ****3. Characteristics of "No-Name" Users****

* **No Login Access:**
  + Usually configured with /sbin/nologin or /bin/false as their shell.
  + They don’t have home directories.
* **UID-Based Ownership:**
  + Files or processes owned by these accounts appear with their UID instead of a name when viewed using ls -l or ps.
* **Security Benefits:**
  + Since there’s no associated username, they’re harder to identify and exploit in brute-force login attempts.

### ****4. Example Use Cases****

#### Assign File Ownership to a UID

You can use a UID without a username to own files:

chown 1002:1002 /some/file

When listing files, you’ll see:

yaml

-rw-r--r-- 1 1002 1002 4096 Nov 16 16:00 file.txt

#### Process Isolation

A service may run under a system UID with no corresponding username:

ps -u 1002

### ****5. Risks and Challenges****

1. **Lack of Identification:**
   * Without a username, it's harder to audit or track ownership and permissions.
2. **Misconfiguration:**
   * Improper setup can cause system issues or conflicts with other UIDs.
3. **Human Readability:**
   * System administrators may struggle to identify the purpose of a UID-based user.

### ****6. Creating Pseudo or Restricted Users****

If the intent is to create a non-login, restricted user for system purposes:

sudo useradd --system --no-create-home --shell /usr/sbin/nologin pseudo\_user

This creates a restricted user that cannot log in or have a home directory.

In summary, while traditional no-name users are rare and not directly supported by tools like useradd, they can exist for specialized purposes, typically involving system processes, security, or testing.

The usermod command in Linux is used to **modify user account details**. It allows system administrators to change various attributes of a user account, such as their username, home directory, group memberships, shell, and more.

Here’s a breakdown of how to use the usermod command:

### ****Syntax****

usermod [options] USERNAME

* **USERNAME**: The name of the user account you want to modify.
* **[options]**: Various flags to specify what changes to make.

### ****Common Options for**** usermod

| **Option** | **Description** |
| --- | --- |
| -l NEW\_USERNAME | Change the user's username to NEW\_USERNAME. |
| -d NEW\_HOME\_DIR | Change the user's home directory to NEW\_HOME\_DIR. Does not move existing files. |
| -m | Moves the user's existing home directory to the new location specified with -d. |
| -s NEW\_SHELL | Change the user's login shell to NEW\_SHELL. |
| -c COMMENT | Change the user's description or comment (usually in /etc/passwd). |
| -g GROUP | Change the user's primary group to GROUP. |
| -G GROUP1,GROUP2 | Add the user to supplementary groups. |
| -a | Add the user to supplementary groups **without removing existing memberships** (used with -G). |
| -L | Lock the user's account (disables login). |
| -U | Unlock the user's account. |
| -e DATE | Set the account expiration date in the format YYYY-MM-DD. |
| -f DAYS | Set the number of days after a password expires before the account is disabled. |

### ****Examples of**** usermod ****Usage****

#### 1. **Change a User's Username**

sudo usermod -l new\_username old\_username

* Changes the username from old\_username to new\_username.

#### 2. **Change a User's Home Directory**

sudo usermod -d /new/home/directory username

* Updates the home directory path but doesn’t move files.

To move the files as well:

sudo usermod -d /new/home/directory -m username

#### 3. **Add a User to Supplementary Groups**

sudo usermod -G group1,group2 username

* Assigns the user to group1 and group2. Existing group memberships will be removed unless -a is used.

To append groups without removing existing ones:

sudo usermod -a -G group1,group2 username

#### 4. **Change a User's Login Shell**

sudo usermod -s /bin/zsh username

* Changes the user's default shell to Zsh.

#### 5. **Set or Remove an Account Expiry Date**

To set an expiration date:

sudo usermod -e 2024-12-31 username

To remove the expiration date:

sudo usermod -e "" username

#### 6. **Lock or Unlock a User Account**

Lock the account (prevent login):

sudo usermod -L username

Unlock the account:

sudo usermod -U username

### ****Where**** usermod ****Makes Changes****

* **/etc/passwd**: Updates the user’s details like username, home directory, and shell.
* **/etc/shadow**: Updates password-related settings, like expiry and locking.
* **/etc/group**: Updates group memberships.

### ****Best Practices****

1. **Backup Critical Files**: Before making significant changes, back up files like /etc/passwd and /etc/group.

­­­­­­

sudo cp /etc/passwd /etc/passwd.bak

sudo cp /etc/group /etc/group.bak

1. **Test After Changes**: Verify the user’s settings after modifications, especially when changing critical attributes like username or home directory.
2. **Be Cautious with Locking Accounts**: Ensure you don’t inadvertently lock out critical system accounts.

The usermod command is powerful and provides a flexible way to manage user accounts in Linux.

The chmod command in Linux is used to **modify file permissions**. The permissions for files and directories are represented as a combination of **owner**, **group**, and **others**. When using chmod, you can specify permissions in symbolic or numeric form.

If you **add something before or after x** in the symbolic mode of chmod, it refers to specific permission changes related to execution or other attributes. Here’s an explanation:

### ****1. Adding Something Before**** x

When you add a character like u, g, or o before x, it specifies **which category of users** the execution permission is being applied to.

#### Categories:

* **u (user):** The file's owner.
* **g (group):** Members of the file's group.
* **o (others):** All other users.
* **a (all):** All three categories (u, g, and o).

#### Example:

chmod u+x file.txt

* Adds execution permission for the file owner.

chmod g+x file.txt

* Adds execution permission for the group.

chmod a+x file.txt

* Adds execution permission for all categories (owner, group, others).

### ****2. Adding Something After**** x

The x in chmod represents the **execution permission**. If you add other symbols or letters after x, it specifies combinations or additional attributes to apply.

#### Examples:

1. **Adding + or - with x:**
   * +x: Add execution permission.
   * -x: Remove execution permission.

chmod +x script.sh

* + Adds execution permission for everyone.

chmod g-x file.txt

* + Removes execution permission for the group.

1. **Combining with r or w:**
   * r: Read permission.
   * w: Write permission.
   * x: Execution permission.

Example:

chmod u+rx script.sh

* + Grants the file's owner read (r) and execute (x) permissions.

chmod go-wx file.txt

* + Removes write (w) and execute (x) permissions for the group and others.

### ****3. Special Permissions with**** x

Sometimes, x can be combined with **special attributes** like s (setuid/setgid) or t (sticky bit).

#### Setuid and Setgid (s with x):

* **Setuid (u+s):** Allows a file to run with the privileges of its owner.
* **Setgid (g+s):** Allows a file to run with the privileges of its group.
* Common for executables and shared directories.

chmod u+s program

chmod g+s shared\_folder

#### Sticky Bit (o+t):

* Prevents others from deleting files in a directory unless they own them.
* Commonly used in directories like /tmp.

chmod o+t /shared\_dir

### ****4. Symbolic and Numeric Modes****

If you prefer numeric notation, chmod uses octal values:

* 4 = Read (r)
* 2 = Write (w)
* 1 = Execute (x)

Example:

chmod 755 file.txt

* 7 = rwx (owner has all permissions).
* 5 = r-x (group has read and execute permissions).
* 5 = r-x (others have read and execute permissions).

### ****Key Notes****

* **Before x:** Specifies the user category (u, g, o, a).
* **After x:** Often used to combine or clarify other permissions like +x, -x, +rwx.
* Use ls -l to view and verify permission changes:

ls -l file.txt

This makes chmod a versatile tool for managing permissions and securing files and directories.

To **remove a permission** in Linux using the chmod command, you use the - operator in symbolic mode. Specifically for execution permission (x), here’s how you can remove it:

### ****Syntax for Removing Permissions****

chmod [WHO]-[PERMISSION] FILE

* **WHO**: Specifies the target category (u, g, o, or a):
  + u: User (owner).
  + g: Group.
  + o: Others.
  + a: All (default if no WHO is specified).
* **PERMISSION**: The type of permission to remove:
  + r: Read.
  + w: Write.
  + x: Execute.
* **FILE**: The file or directory whose permissions are being changed.

### ****Examples****

#### 1. **Remove Execute Permission for a Specific User Category**

* Remove execute permission for the **user** (owner):

chmod u-x file.txt

* Remove execute permission for the **group**:

chmod g-x file.txt

* Remove execute permission for **others**:

chmod o-x file.txt

#### 2. **Remove Execute Permission for All Categories**

To remove execute permission for **everyone**:

chmod a-x file.txt

### ****Check Permissions****

After modifying the permissions, use ls -l to verify:

ls -l file.txt

The output will look like this:

-rw-r--r-- 1 user group 1234 Nov 16 12:34 file.txt

* If x is removed, it will no longer appear in the permission string (e.g., rwx becomes rw-).

### ****Numeric Mode****

You can also remove execute permissions using numeric notation by setting the appropriate value for each permission category:

* **Read (4)**, **Write (2)**, **Execute (1)**.

#### Example:

To set permissions to **read and write only** (no execute):

chmod 644 file.txt

* 6 = rw- (user).
* 4 = r-- (group).
* 4 = r-- (others).

#### Another Example:

To set permissions for **read/write for owner, no permissions for group and others**:

chmod 600 file.txt

### ****Removing Special Permissions****

If special permissions like s (setuid, setgid) or t (sticky bit) are set, you can remove them as follows:

* Remove **setuid**:

chmod u-s file.txt

* Remove **setgid**:

chmod g-s file.txt

* Remove **sticky bit**:

chmod o-t directory

### ****Key Points****

* Use - with the symbolic mode (u-x, g-x, a-x) to remove permissions.
* Use numeric mode to redefine permissions entirely (e.g., chmod 644).
* Always verify changes using ls -l.

This flexibility allows fine-grained control over file and directory access.

A **cron job** is a scheduled task in Unix-like operating systems, executed automatically at specified intervals using the **cron** daemon. It is commonly used for automation, such as running scripts, performing backups, monitoring systems, or managing repetitive tasks.

### ****Key Concepts of Cron Job****

1. **Cron Daemon (cron)**:
   * A background service that checks a configuration file (crontab) for scheduled tasks and executes them at the specified times.
   * It must be running for cron jobs to work.
2. **Crontab File**:
   * The configuration file where cron jobs are defined.
   * Each user can have their own crontab file.
   * The system-wide crontab is located at /etc/crontab.
3. **Cron Syntax**: Cron jobs follow a specific syntax to define the schedule:

MIN HOUR DOM MON DOW COMMAND

* + **MIN**: Minute (0–59).
  + **HOUR**: Hour (0–23, in 24-hour format).
  + **DOM**: Day of the month (1–31).
  + **MON**: Month (1–12 or Jan, Feb, etc.).
  + **DOW**: Day of the week (0–7, where 0 and 7 are Sunday).
  + **COMMAND**: The command or script to execute.

### ****Examples****

1. **Run a command every minute**:

\* \* \* \* \* /path/to/command

1. **Run a script at 3:00 AM daily**:

0 3 \* \* \* /path/to/script.sh

1. **Run a job every Monday at 5:30 PM**:

30 17 \* \* 1 /path/to/command

1. **Run a command every 15 minutes**:

\*/15 \* \* \* \* /path/to/command

1. **Run a task on the 1st of every month at midnight**:

0 0 1 \* \* /path/to/command

### ****Managing Cron Jobs****

#### List Cron Jobs

To view all cron jobs for the current user:

crontab -l

#### Edit Cron Jobs

To edit the crontab for the current user:

crontab -e

* This opens a text editor where you can add or modify cron jobs.

#### Remove Cron Jobs

To remove all cron jobs for the current user:

crontab -r

#### View System-Wide Cron Jobs

Check the system-wide crontab:

cat /etc/crontab

### ****Special Scheduling Keywords****

Instead of specifying time fields, you can use special keywords:

* @reboot: Run the job once at startup.
* @yearly or @annually: Run the job once a year (0 0 1 1 \*).
* @monthly: Run the job once a month (0 0 1 \* \*).
* @weekly: Run the job once a week (0 0 \* \* 0).
* @daily: Run the job once a day (0 0 \* \* \*).
* @hourly: Run the job once an hour (0 \* \* \* \*).

Example:

@daily /path/to/daily-task.sh

### ****Common Uses****

1. **Automated Backups**: Schedule regular backups of databases or filesystems.

0 2 \* \* \* /usr/bin/backup.sh

1. **Log Rotation**: Rotate and compress logs to save disk space.

0 0 \* \* \* /usr/sbin/logrotate /etc/logrotate.conf

1. **Monitoring and Alerts**: Run a monitoring script every minute.

\* \* \* \* \* /path/to/monitor.sh

1. **Cleanup Tasks**: Delete temporary files older than 7 days.

0 3 \* \* \* find /tmp -type f -mtime +7 -exec rm {} \;

### ****Important Notes****

1. **Environment Variables**: Cron jobs may not inherit your shell's environment. Define necessary variables (e.g., PATH) in the script or the crontab.

Example:

PATH=/usr/bin:/bin:/usr/local/bin

1. **Output and Logging**: By default, cron sends output to the user's email (if configured). Redirect output to a file for logging:

\* \* \* \* \* /path/to/command >> /path/to/logfile 2>&1

1. **Permissions**: Ensure the script or command has executable permissions.

chmod +x /path/to/script.sh

Cron jobs are powerful tools for task automation, simplifying routine maintenance and operational workflows.

Here’s a simple **cron job script** example that performs a routine task: backing up a directory.

### ****Script:**** backup.sh

#### Purpose:

This script backs up the contents of a specified directory to a backup location, compressing it with a timestamp for uniqueness.

#!/bin/

# Directories

SOURCE\_DIR="/home/ubuntu/data" # The directory to back up

BACKUP\_DIR="/home/ubuntu/backups" # Where the backup will be stored

# Create the backup directory if it doesn't exist

mkdir -p "$BACKUP\_DIR"

# Generate a timestamp for the backup file

TIMESTAMP=$(date +"%Y-%m-%d\_%H-%M-%S")

# Backup file name

BACKUP\_FILE="$BACKUP\_DIR/backup\_$TIMESTAMP.tar.gz"

# Perform the backup using tar

tar -czf "$BACKUP\_FILE" "$SOURCE\_DIR"

# Print a message

echo "Backup completed successfully: $BACKUP\_FILE"

### ****Set up the Cron Job****

1. **Make the Script Executable**:

chmod +x /path/to/backup.sh

1. **Schedule the Script in Cron**: Open the crontab editor:

crontab -e

Add the following line to schedule the script daily at 2:00 AM:

0 2 \* \* \* /path/to/backup.sh >> /path/to/backup.log 2>&1

* + This schedules the script to run daily at 2:00 AM.
  + The output and errors are logged to /path/to/backup.log.

### ****How It Works****

1. **Directories**: The script defines the directory to back up and the location to store the backup.
2. **Timestamp**: A unique timestamp is added to the backup file name to prevent overwrites.
3. **Compression**: The tar command compresses the backup into a .tar.gz file.
4. **Automation**: The cron job ensures this process runs automatically without manual intervention.

### ****Verify Cron Job Execution****

* Check the backup directory (/home/ubuntu/backups) to ensure backups are created.
* Check the log file (backup.log) for any errors or status messages.

When an operating system (OS) starts (also known as **booting**), it goes through a series of steps to initialize the system, load the necessary components, and make the computer ready for use. This process can be broken down into several stages:

**1. Power-On (POST):**

* **Power-on Self Test (POST)** is the first step. When you press the power button, the hardware of the computer is powered on.
* **BIOS/UEFI** (Basic Input/Output System / Unified Extensible Firmware Interface) runs a self-test to check essential components like the CPU, RAM, hard drives, keyboard, and other hardware components for errors. If everything is functioning properly, the process proceeds.

**2. BIOS/UEFI Initialization:**

* BIOS or UEFI firmware, located on the motherboard, takes control of the system and looks for the bootable device (like a hard disk, SSD, USB drive, etc.).
* BIOS/UEFI configures low-level hardware components, and it also checks the boot sequence configured in the system's settings (e.g., boot from the hard drive first, then USB, etc.).

**3. Bootloader (Stage 1):**

* After BIOS/UEFI has finished its task, it transfers control to a **bootloader**. The bootloader is a small program stored on the boot device (like a hard drive or SSD).
* **GRUB** (Grand Unified Bootloader) or **LILO** (Linux Loader) is commonly used in Linux systems. On Windows, it could be the Windows Boot Manager.
* The bootloader’s job is to load the OS kernel into memory. If there are multiple operating systems (as in dual-boot systems), the bootloader provides a menu to select which OS to boot.

**4. Loading the Kernel:**

* The bootloader loads the **kernel** (e.g., vmlinuz in Linux) into memory. The kernel is the core part of the operating system responsible for interacting with the hardware and providing services to software applications.
* The kernel is the interface between the hardware and the user applications. It manages the system's resources (CPU, memory, storage, etc.).
* After loading, the bootloader hands control over to the kernel.

**5. Initializing the Kernel:**

* The kernel initializes critical hardware and sets up system resources like:
  + **Memory management**: Setting up virtual memory, paging, and RAM.
  + **Device drivers**: Loading drivers for hardware devices such as network interfaces, hard drives, display, etc.
  + **Process management**: Setting up the process scheduler and starting system processes.
* The kernel then mounts the **root filesystem** (the main file system where the OS resides) and begins to initialize other subsystems needed to operate the OS.

**6. Init Process (Systemd, SysVinit, Upstart):**

* After the kernel is loaded and initialized, it launches the **init process** (the first process in Unix-like operating systems), which is assigned process ID 1 (PID 1).
  + **Systemd** (the most commonly used init system on modern Linux distributions) or **SysVinit** or **Upstart** can be the init system, depending on the Linux distribution or UNIX system being used.
* The init process is responsible for starting and managing background services (daemons) and processes necessary for the system to function.
  + **Systemd** is a modern and complex init system that not only starts services but also handles tasks like managing log files, user sessions, and system states.

**7. Starting System Services (Daemons):**

* The init system starts essential background services (or **daemons**) such as:
  + Networking services (e.g., NetworkManager, dhclient, etc.)
  + Display manager (e.g., lightdm, gdm for graphical login)
  + File system services (e.g., mounting filesystems)
  + Logging services (e.g., syslog, journalctl for logging)
* These services ensure that the system is ready for user interaction and all system resources are functional.

**8. User Login:**

* Once the system services are running, the OS will display a login prompt (for command-line interfaces like Linux or a graphical login screen if the system uses a GUI).
* The user can log in by providing their credentials (username and password).
  + In a graphical system, a display manager (e.g., GDM, LightDM) controls the graphical login screen.
  + In a non-graphical system, a terminal or console login prompt appears.

**9. User Environment Setup:**

* After the user logs in, the system loads user-specific settings, configurations, and environment variables.
  + This is usually done through startup files like ~/.rc, ~/.profile, etc., which configure user preferences such as terminal settings, environment variables, and startup applications.

**10. User Session:**

* The system now enters an interactive session where the user can start using applications, run commands, or interact with the graphical desktop (if applicable).
* The system is fully operational, with all services, applications, and user interfaces running.

**Summary of Boot Process:**

1. **Power-on**: Hardware starts, POST runs.
2. **BIOS/UEFI**: Initializes hardware, checks boot devices.
3. **Bootloader**: Loads OS kernel.
4. **Kernel**: Initializes system resources, mounts the root filesystem.
5. **Init Process**: Starts essential services and daemons.
6. **User Login**: Displays login prompt.
7. **User Session**: The user begins interacting with the system.

Each of these steps is vital for the system to become fully operational and ready for use, allowing users and applications to run efficiently.

**LINUX**

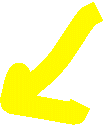
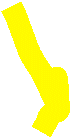
What is operating system

* Hardware (Motherboard, RAM, Processor, Hard Disk etc.)
* Memory Management
* Managing Multitasking
* Managing multiple users
* Process management
* Device management
* Error Handling and logging

Things to Remember in Linux

* Everything is a File
* The CLI is your friend

Server



Hardware + Operating System(windows , Linux [RHEL ,Ubuntu , Kali ,CENTOS, etc]

**Architecture of LINUX**

* Hardware
* Kernel (Program)
* Shell (Program)
* Commands
* File and directories (Folder in windows)

C: window file /user/program files (files which are installed)



Windows



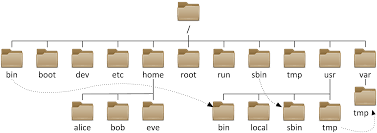
D: data

**Linux**

* / -> root -> starting of operating system
* root -> home dir of root user
* root user admin account in linux
* root user non-root user

**Notes:**

* server operating system used by the companies is red hat Linux
* fedora has more bugs than CentOS thus always testing is done on Fedora then on CentOS both are test operating system provided by red hat
* CLI - Command line interface
* In VMWARE we can’t run multiple virtual Machine at once we can run it one by one but in virtual box it is possibl we can run n-number of virtual machines parallel
* In Linux there is home directory (where we are going to work) for each user
* in Linux "directories"==in windows "folders"
* kernel (inside the O.S) understand the language of hardware and the kernel understand the end user command using shell program explain command to kernel and kernel interpretate to hardware and then we get the output of command
* root user can run any command that is admin
* non root user can run only limited commands
* non root user can only run those command which is in ==>/usr/bin (directory)
* shell program runs the files inside the bins and other directories and all to get desired output in seconds but in backend all the process is occuring
* ==> Bourne again shell( most commonly used default shell program responsible to run the Linux commands)
* sh ==> Bourne shell 1970 (old version)
* csh ==> for developer
* TCSH
* If rotating arrow symbol is with any Folder it means it is shortcut file
* to check which shell program used in Linux ==> echo $SHELL
* [student@localhost ~]$ ls 🡺 here student is which user login (root), localhost is system name , $ normal user is login not any super user



**/ – Root**

* Every single file and directory starts from the root directory.
* Only root user has write privilege under this directory.
* Please note that /root is root user’s home directory, which is not same as /.

**/bin – User Binaries**

* Contains binary executables.
* Common linux commands you need to use in single-user modes are located under this directory.
* Commands used by all the users of the system are located here.
* For example: ps, ls, ping, grep, cp.

**/sbin – System Binaries**

* Just like /bin, /sbin also contains binary executables.
* But, the linux commands located under this directory are used typically by system aministrator, for system maintenance purpose.
* For example: iptables, reboot, fdisk, ifconfig, swapon

**/etc – Configuration Files**

* Contains configuration files required by all programs.
* This also contains startup and shutdown shell scripts used to start/stop individual programs.
* For example: /etc/resolv.conf, /etc/logrotate.conf

**/dev – Device Files**

* Contains device files.
* These include terminal devices, usb, or any device attached to the system.
* For example: /dev/tty1, /dev/usbmon0

**/proc – Process Information**

* Contains information about system process.
* This is a pseudo filesystem contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
* This is a virtual filesystem with text information about system resources. For example: /proc/uptime

**/var – Variable Files**

* var stands for variable files.
* Content of the files that are expected to grow can be found under this directory.
* This includes — system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

**/tmp – Temporary Files**

* Directory that contains temporary files created by system and users.
* Files under this directory are deleted when system is rebooted.

**/usr – User Programs**

* Contains binaries, libraries, documentation, and source-code for second level programs.
* /usr/bin contains binary files for user programs. If you can’t find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
* /usr/sbin contains binary files for system administrators. If you can’t find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
* /usr/lib contains libraries for /usr/bin and /usr/sbin
* /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

**/home – Home Directories**

* Home directories for all users to store their personal files.
* For example: /home/john, /home/nikita

/**boot – Boot Loader Files**

* Contains boot loader related files.
* Kernel initrd, vmlinux, grub files are located under /boot
* For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

**/lib – System Libraries**

* Contains library files that supports the binaries located under /bin and /sbin
* Library filenames are either ld\* or lib\*.so.\*
* For example: ld-2.11.1.so, libncurses.so.5.7

**/opt – Optional add-on Applications**

* opt stands for optional.
* Contains add-on applications from individual vendors.
* add-on applications should be installed under either /opt/ or /opt/ sub-directory.

**/mnt – Mount Directory**

* Temporary mount directory where sysadmins can mount filesystems.

**/media – Removable Media Devices**

* Temporary mount directory for removable devices.
* For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

**/srv – Service Data**

* srv stands for service.
* Contains server specific services related data.
* For example, /srv/cvs contains CVS related data.

**NOTES:**

* cat ==>to see content of a file
* su ==> to switch user eg: su root ==> i will become root and it will ask for password
* etc => system configuration file
* var ==> store website file, database file, log file
* run ==> used by kernel (not even for root user)
* home ==> non root user can store their personal data
* root ==> home directory of root user (it will always require password if other than root user is trying to access)
* temp ==>any user can create here file and directory //if a file is not touched from many days then it automatically get deleted after 10 days if not accessed from 10 days
* boot ==> contain kernel program and booting related file
* dev ==> all files access the hardware
* ls ==>command shows the content of the current location , red colour directory are zipped file
* pwd ==>present working directory
* ls -ld/var ==> don’t go inside the var directory just give the information
* ls --help ==> will provide the options which could be used with this command ls

**MOST IMPORTANT COMMANDS OF LINUX**

1. **ls** - The most frequently used command in Linux to list directories [ files in red are zipped files]
2. **pwd** - Print working directory command in Linux
3. **cd** - Linux command to navigate through directories
4. **mkdir** - Command used to create directories in Linux
5. **mv** - Move or rename files in Linux
6. **cp** - Similar usage as mv but for copying files in Linux
7. **rm** - Delete files or directories
8. **touch** - Create blank/empty files
9. **ln** - Create symbolic links (shortcuts) to other files
10. **cat** - Display file contents on the terminal
11. **clear** - Clear the terminal display
12. **echo** - Print any text that follows the command
13. **less** - Linux command to display paged outputs in the terminal
14. **man** - Access manual pages for all Linux commands
15. **uname** - Linux command to get basic information about the OS
16. **whoami** - Get the active username
17. **tar** - Command to extract and compress files in Linux
18. **grep** - Search for a string within an output
19. **head** - Return the specified number of lines from the top
20. **tail** - Return the specified number of lines from the bottom
21. **diff** - Find the difference between two files
22. **cmp** - Allows you to check if two files are identical
23. **comm** - Combines the functionality of diff and cmp
24. **sort** - Linux command to sort the content of a file while outputting
25. **export** - Export environment variables in Linux
26. **zip** - Zip files in Linux
27. **unzip** - Unzip files in Linux
28. **ssh** - Secure Shell command in Linux
29. **service** - Linux command to start and stop services
30. **ps** - Display active processes
31. **kill and killall** - Kill active processes by process ID or name
32. **df** - Display disk filesystem information
33. **mount** - Mount file systems in Linux
34. **chmod** - Command to change file permissions
35. **chown** - Command for granting ownership of files or folders
36. **ifconfig** - Display network interfaces and IP addresses
37. **traceroute** - Trace all the network hops to reach the destination
38. **wget** - Direct download files from the internet
39. **ufw** - Firewall command
40. **iptables** - Base firewall for all other firewall utilities to interface with
41. **apt, pacman, yum, rpm** - Package managers depending on the distro
42. **sudo** - Command to escalate privileges in Linux
43. **cal** - View a command-line calendar
44. **alias -** Create custom shortcuts for your regularly used commands
45. **dd** - Majorly used for creating bootable USB sticks
46. **whereis** - Locate the binary, source, and manual pages for a command
47. **whatis** - Find what a command is used for
48. **top** - View active processes live with their system usage
49. **useradd and usermod** - Add new user or change existing users data
50. **passwd** - Create or update passwords for existing users

ec2-user ]# 🡺 here ec2 is location of file

ctrl +o 🡺enter🡺ctrl+x to save files (follow the commands)

vi, nano ,vim 🡺 editor (press “i” for inserting and to make changes)

:wq! 🡺save and exit

Cp -fv 🡺copy with force verbose ( it shows the command operated with its output)

rm 🡺 to delete file and directories

/\*t 🡺 all file starting with t

@ 🡺 at which server/name/IP

~ 🡺 home directory

$ 🡺 non root user

# 🡺 root user

W 🡺 who all are login right now and what are they doing

* Syntax: Command -options arg1 arg2 …….argn
* for example: ls -l 🡺present dir will act as a arg OR ls -ld /var

NOTES:

* **[ Student@localhost~]$** 🡺It means the user “Student” whose server name is localhost is at home directory and the user is non-root user.
* Ls 🡺 provide list of content
* Ls -l 🡺 provide list of data with additional data
* Ls -l/var/usr 🡺 it will take multiple arguments to show their list

**DIFFERENT COMMANDS WITH THEIR MEANING**

* [Student@serverA ~]$ pwd 🡺 Here it will show the present working directories , since ~ is present which means it is home directory

o/p🡺 /Home/Student

* [Student@serverA ~]$ cd Desktop/ 🡺 change directory to desktop

o/p 🡺[Student@serverA Desktop]$

* [Student@serverA Desktop]$ touch file1🡺 it will create a file in Desktop directory with different options of read and write
* [Student@serverA Desktop]$ touch file1 file2 ..==> it will create multiple files
* [Student@serverA Desktop]$ date 🡺 it will show the date and time
* [Student@serverA Desktop]$ rm file1 🡺 it will remove the file1
* [Student@serverA Desktop]$ mkdir dir1 🡺 it will create empty directory dir1
* [Student@serverA Desktop]$ rmdir dir1 🡺it will remove the directory dir1
* [Student@serverA Desktop]$ cat/etc/ssh/sshd\_config 🡺 permissions will be denied because it is accessed by the root user only
* [Student@serverA Desktop]$ cd/var/ 🡺 change directory to var

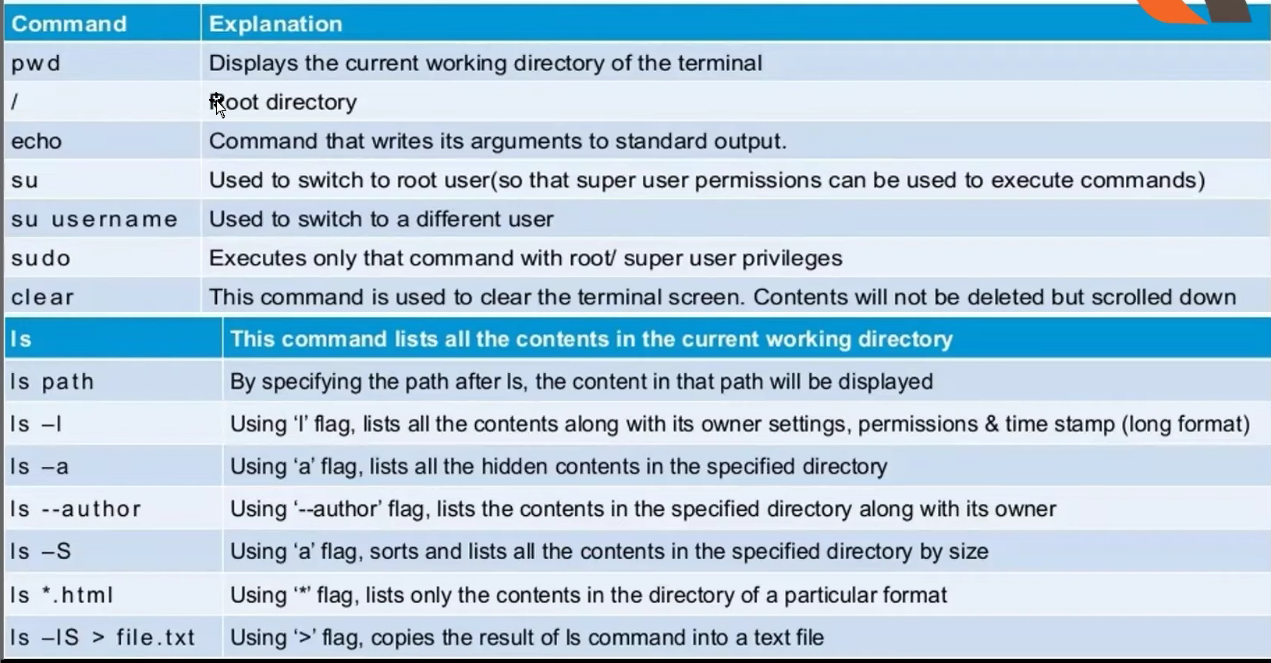
NOTES:

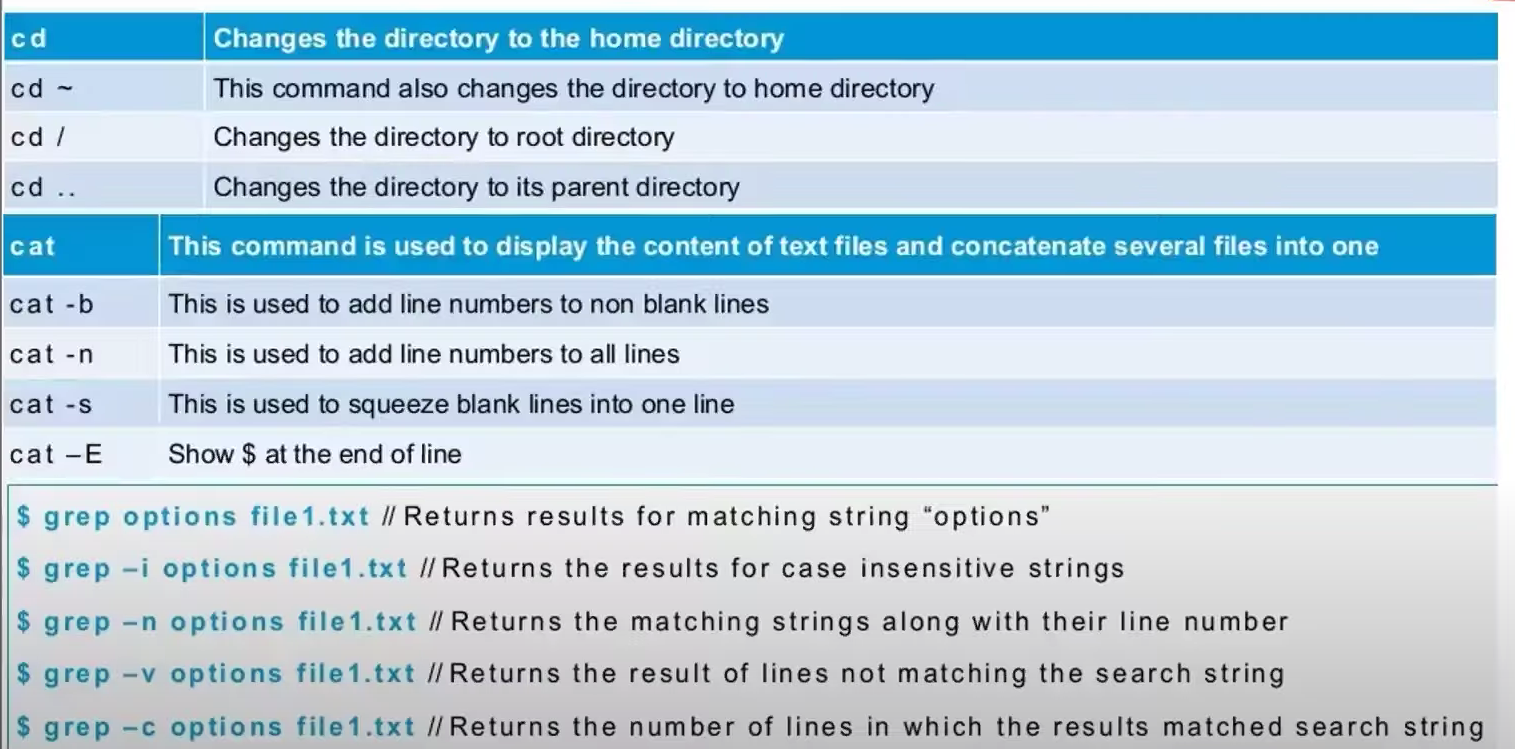
* If we want to change the server name or other name at place of IP ( the local host is default name of server if not changed during installation)

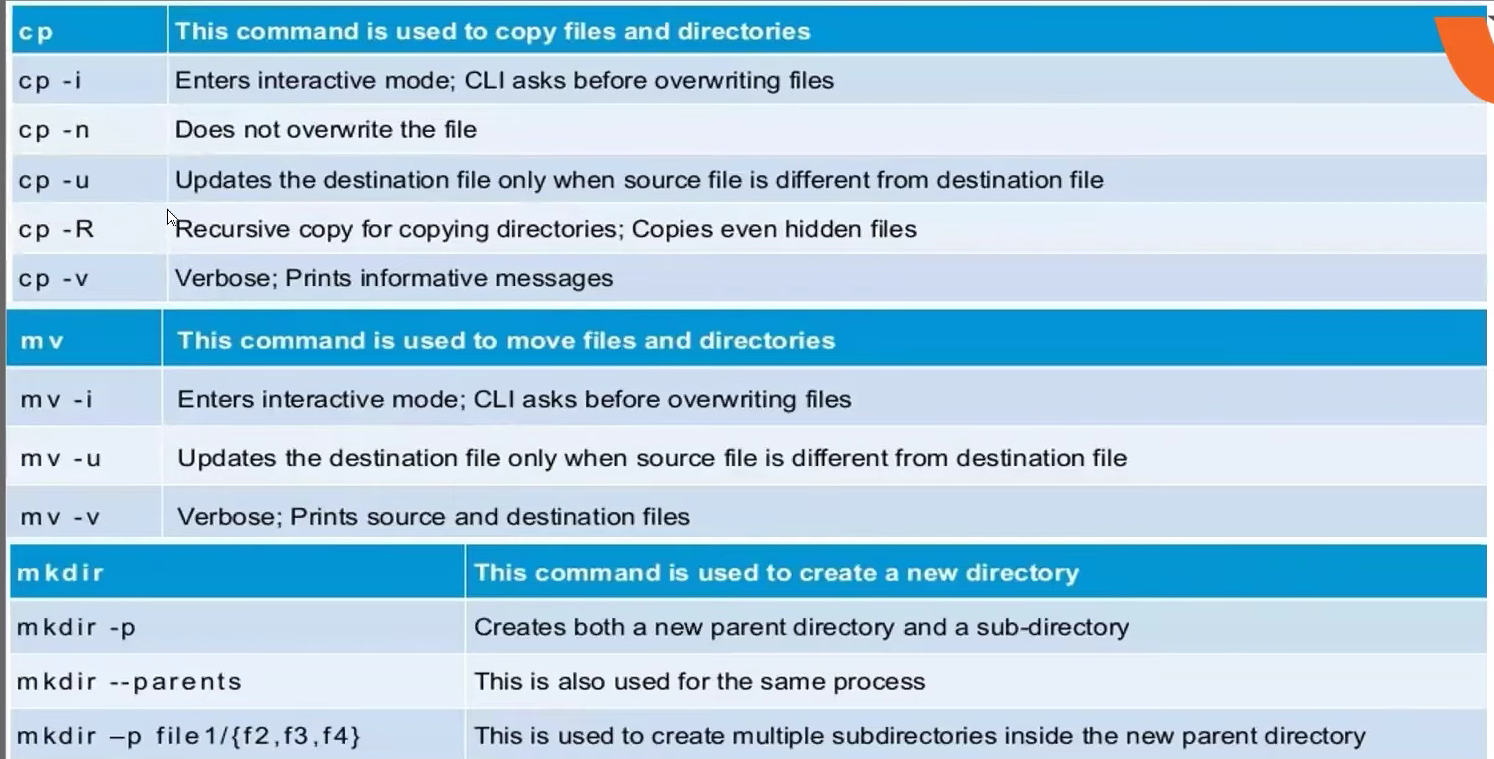
Then use command **nmtui( network manager terminal user interface)**

[Student@serverA Desktop]$ nmtui 🡺(only arrow key will work here and it will ask for root password)🡺 as press ok then run the command “” to see the changes.

* If we have created a File then again if we write command touch with sane file then all thing will be same only its creation date will change
* Never delete any file because we cant store it directly it could only be restored when hard-disk goes to lab it is very expensive process
* /home/student in linux is same as C:\users\vaibhaw
* A non-root user can become a root user by using sudo( all user not have right to use sudo , only few can use who have authorization like system administrator in problem dusring troubleshooting
* shell script 🡺 Linux and Power shell script 🡺 windows
* All the directories are of blue color by default ( auto by alias)
* Comments are done with # symbol
* “-“ 🡺load the environment variable of user
* In Vi editor screen is divided into rows and column
* To work with Vi mode we have to write insert mode then we have start the command for saving and other commands









* [Student@serverA Desktop]$ cd.. 🡺 it will take two steps back path
* [Student@serverA Desktop]$ cd 🡺 it will take to root path
* [Student@serverA Desktop]$ cat file1 🡺 it will display ( read mode)

This is some sample data

This is another line

* [Student@serverA Desktop]$ clear 🡺 it will scroll down so that user directly reach to the first line
* [Student@serverA Desktop]$ who 🡺 it will give the name of user with date and time
* [Student@serverA Desktop]$ whoami 🡺it will only give the name of user
* [Student@serverA Desktop]$ ip a 🡺 it will give the ip of system with different other data’s about the system like Broad cast id and many things
* [Student@serverA Desktop]$ hostname 🡺 it will give the host name
* [Student@serverA Desktop]$ uname 🡺 it will give name of O.S i.e linux
* [Student@serverA Desktop]$ uname -a 🡺 will provide all data about the system and its IP
* [Student@serverA Desktop]$history 🡺 it will give the history
* [Student@serverA Desktop]$ echo “hello world” 🡺 it will show hello world
* [Student@serverA Desktop]$ su root 🡺 it will switch user to root but it require password of root
* [Student@serverA Desktop]$ sudo su -root 🡺 it will switch a user to root without any root user password
* [Student@serverA Desktop]$ logout 🡺to logout user ( alternate ctrl +d)
* [Student@serverA Desktop]$ sudo useradd Rohit 🡺 to add user Rohit
* [Student@serverA Desktop]$ rm -r dir1 🡺 it will remove all the files of dir1 recursively and at last it will delete the directory
* [Student@serverA ~]$ which useradd 🡺 it will tell which file has useradd command i.e /usr/sbin/useradd
* [Student@serverA ~]$ sudo useradd abc 🡺 after adding provide the password
* [Student@serverA ~]$ id abc 🡺 it will provide the user id and group id of user and each user has default group id along userid
* [Student@serverA ~]$ sudo userdel abc 🡺 it will delete the user abc
* [Student@serverA Desktop]$ rm -rf file1 🡺 go to directory remove the file recursively then delete it forcefully
* [Student@serverA Desktop]$ rm -r file1 🡺 not forcefully remove
* [Student@serverA ~]$ exit or logout or ctrl+d 🡺to exit the terminal if it is at its initial path
* [Student@serverA Desktop]$ grep sample file1 🡺 it will help to find the string sample in the file1
* [Student@serverA Desktop]$ grep -i sample file1 🡺 it will help to find string independent to case notation( capital or small)
* [Student@serverA Desktop]$ grep -o sample file 🡺 it indicate the occurrence of word sample in file1
* [Student@serverA Desktop]$ grep -v”#” file1 🡺 it will remove the comment with # symbol if we execute the file 🡺cat /etc/ssh/sshd\_config
* [Student@serverA ~]$ grep -v “#” /etc/ssh/sshd\_config 🡺 give content without comment with # symbol
* [Student@serverA ~]$ grep -o Vaibhaw/home/student/Desktop/file1🡺 it will show all times vaibhaw occurred in this content
* Cd +enter 🡺 directly to home directory
* Cd - 🡺previous directory
* Cd ~ 🡺 home directory
* Cd / 🡺 will take to root

**Basic Commands**

1.file commands

* + Is, touch, mkdir, rm, rmdir, cat

2.directory navigation commands

* cd, cd.., cd/path, pwd

3. users

* + id, last, whoami, who,w,

4. system information

* + uname, hostname, date,

5. network information

* + ip a, nmtui, ping,

6. history

* history
* crtl+r
* Inumber

**Practise with outcome**

* **[student@serverA Desktop]$ sudo su – root** 🡺 **give password and switch to root**
* **[root@serverA~]$pwd** 🡺**it will show the current working directory of user Student**
* **[root@serverA~]$ cd Desktop/** 🡺 **change directory to desktop**
* **[root@serverA Desktop]$ ls -l** 🡺 **it will show the content of desktop directory if anything is present**
* **[root@serverA Desktop]$ touch file1** 🡺 **created a file**
* **[root@serverA Desktop]$date** 🡺**it will show when the file is created**
* **[root@serverA Desktop]$ touch file1** 🡺 **since file1 is already created in same directory thus it will show its only updated time and date as we do ls**
* **[root@serverA Desktop]$ ls -l** 🡺 **updated only date and time will be shown**
* **[root@serverA Desktop]$ touch file2** 🡺 **file will be created**
* **[root@serverA Desktop]$ pwd** 🡺 **to see the present working directory i.e Desktop**
* **[root@serverA Desktop]$ rm file1** 🡺 **it will remove the file1 from the Desktop**
* **[root@serverA Desktop]$ rmdir dir1** 🡺 **delete directory dir1 if present**
* **[root@serverA Desktop]$ cat/etc/ssh/sshd\_config** 🡺 **Desktop has no right to access this directory**
* **[root@serverA Desktop]$ cd /var/** 🡺 **change directory to var**
* **[root@serverA var]$ LL(in small letter)** 🡺 **to see wheather temp is in var or not**
* **[root@serverA var]$ cd /tmp/** 🡺 **change directory to tmp**
* **[root@serverA var]$ LL** 🡺 **to see which files and directories are presents**
* **[root@serverA var]$ cd..**
* **[root@serverA var]$ cd**
* **[root@serverA var]$ cd Desktop/**
* **[root@serverA Desktop]$ touch file1**
* **[root@serverA Desktop]$ ls -l**
* **[root@serverA Desktop]$ cat file1** 🡺 **it will show if any thing is present in file1( read mode enable)**
* **[root@serverA /]$ sudo userdel abc** 🡺**it will show delete abc if it is their**
* **[root@serverA /]$ id abc** 🡺 **if abc present it will show abc id**
* **[root@serverA ~]$ cd /home/student**
* **[root@serverA student]$ cd Desktop/**
* **[root@serverA Desktop]$ LL** 🡺 **see the directory present for futher commands**
* **[root@serverA Desktop]$ mv dir1 /tmp/** 🡺 **it will move dir1 to tmp**
* **[root@serverA Desktop]$ mv file1 chapter1** 🡺 **it will rename file1 to chapter1**
* **Example: if want to add one file into other like** 🡺**mkdir school/staff/techer1**
* **[root@serverA Desktop]$ mkdir -p school/staff/teacher1** 🡺**to create parent directory**
* **[root@serverA Desktop]$ ls -l** 🡺 **check weather created or not**
* **[root@serverA Desktop]$ cd school/**
* **[root@serverA School]$ ls**
* **[root@serverA School]$ cd staff/**
* **[root@serverA staff]$ ls** 🡺 **it will show teacher**

Alias and commands

Alias

An alias is a command that a user can define as needed. Some aliases are provided by default; type alias on the command line to get an overview.

To define an alias, use allas newcommand='oldcommand"

Aliases are executed before anything else.

Ls – lh 🡺 command to show the ls command output in human readable form

Ls – ltrh 🡺list, timestamp, reverse sorting, human readable

Alias lh = ‘ls -ltrh’ 🡺 alias command that means value is stored in a new variable so next time we have to just write lh to get lh -ltrh

SPATH

To look up commands, use the $PATH variable.

This variable defines a list of directories that is searched for a matching filename when a user enters a command. To find out which exact command the shell will be using, you can use the which command,

For instance, type which is to find out where the shell will get the is command from. If you need to start a command that is in the current directory but not in SPATH then add / in front of it

The dot stands for the current directory, and by running it as /, you tell to look for the command in the current directory

* [student@serverA ~]$ cd /tmp 🡺 change directory to temp
* [student@serverA tmp]$ ls 🡺 list content not in proper way
* [student@serverA tmp]$ ls -lh 🡺list of content in proper way
* [student@serverA tmp]$ ls -ltrh 🡺 content in more proper way
* [student@serverA tmp]$ alias lh = ‘ls -ltrh’ 🡺 created alias
* [student@serverA tmp]$ lh 🡺 content in more proper way without writing whole command
* [student@serverA tmp]$ alias 🡺 it will show default alias shortcut for different commands
* [student@serverA tmp]$ which ls 🡺 it will show alias ls =’ls –color=auto’ /usr/bin/ls
* [student@serverA tmp]$ echo $PATH 🡺 it will give value of a variable or path
* Ls => (shell will try to find the location of ls command using info saved in $PATH) will read the command the file translate to kernel ,kernel will talk to hardware to get the output
* [student@serverA tmp]$ ls -la 🡺 it will shows all hidden files of list
* Dot “ represent” the hidden files in list 🡺.file1
* Mv .fiile1 newfile 🡺 by renaming file name we can make it unhidden
* [student@serverA ~]$ env 🡺 it will show the current user environment

**User and groups**

* + The three critical files containing user and group information are /etc/passwd, /etc/group, and /etc/shadow.
  + The su and sudo commands can be used to run commands as the superuser.
  + The useradd, usermod, and userdel, passwd commands can be used to manage users.
  + The groupadd, and groupdel commands can be used to manage groups.
  + The chage command can be used to configure and view password expiration settings for users
* Every user is By default user of primary group
* su - 🡺 this will switch to the root
* passwd 🡺 to set password for both root and non root
* user cant keep password less then 8 character but root can keep
* user cant keep username as password but root can overite these rules
* for Example: College - ABC – SERVERA - balancesheet.txt only users part of finance group can acces or modify

finance

emp1, emp2, emp3

staff

techerl, techer2, techer3, techer4

User - abcd -

primary group abcd

secondary group finance, staff

**Commands for practice**

[student@serverA~]$ id student 🡺 it will give user id and both primary and secondarygroup id (if exist)

[student@serverA~]$ su - 🡺 switch user to root by providing password

[root@serverA~]# useradd abcd 🡺it will add user abcd

[root@serverA~]# id abcd

[root@serverA~]# passwd abcd 🡺 it will set abcd password

[root@serverA~]# usermod -G wheel abc 🡺 it will add user in secondary group wheel

[root@serverA~]# id abcd 🡺 check the user added or not

[root@serverA~]# cat /etc/passwd

Wheel abc

Student:x:1000:1000:student:/home/student:/bin/

Field-1 username

Field-2 denotes password(X 🡺 password encrypted by cryptography,md5,sha512,sha128)

Field-3 UID

Field-4 GID

Field-5 home dir info

Field-6 comments

Field-7 which shell assigned

Example : useradd -c “this is a test user”

Student:x:1000:1000:this is a test user:/home/student:/bin/

[root@serverA~]# groupadd staff 🡺 it will create group staff

[root@serverA~]# groupadd finance 🡺 it will create group finance

[root@serverA~]# cat/etc/group

[root@serverA~]# id abc

[root@serverA~]# usermod -G finance abc 🡺 it will add user to group finance but it will be removed from group wheel

[root@serverA~]# usermod -aG wheel abc 🡺 to retain all the previous groups

[root@serverA~]# usermod -aG staff abc 🡺 abc is now added to group finance ,wheel and staff

[root@serverA~]#grep finance /etc/group 🡺 it will show the finance group with the user id and other groups

[root@serverA~]# usermod -aG wheel, finance student 🡺 student is added to both the secondary groups

[root@serverA~]# groupdel staff

[root@serverA~]#cat /etc/shadowstudent

: $6$qMKW0:18926:1:90:7:::

Field-1 - username

Field-2 - Password Hash

Field-3 - Unix timestamp - when was the last time password was changed

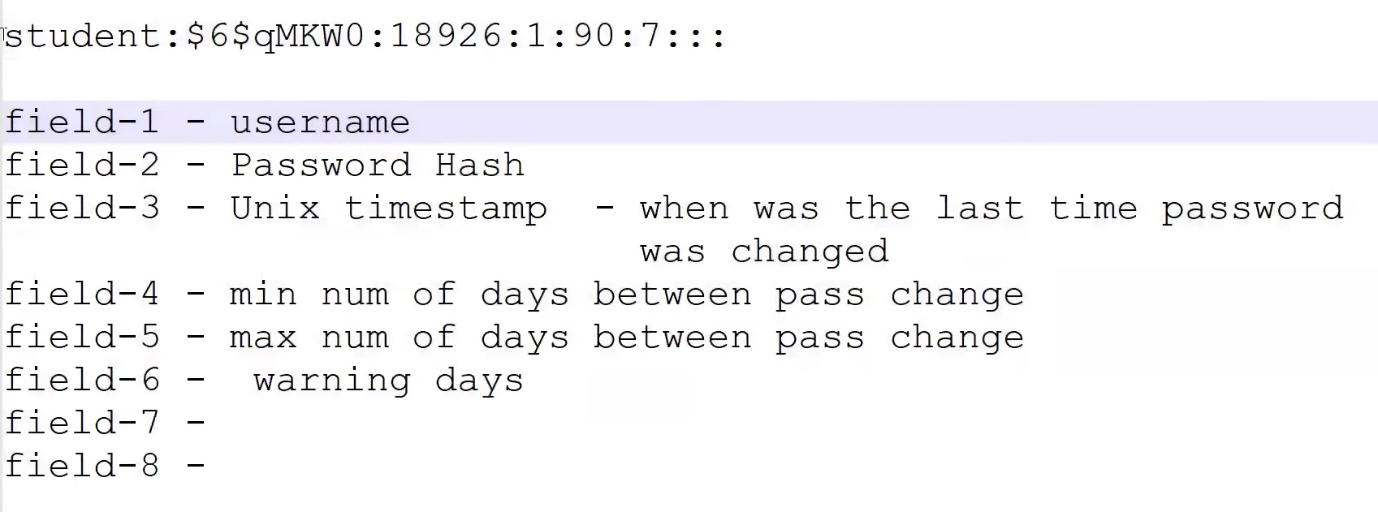
Field-4 - min num of days between pass change

Field-5 - max num of days between pass change

Field-6 – warning days

Field-7

Field-8



[root@serverA~]# chage -l student 🡺 it will give all details of of password of student

[root@serverA~]# chage student 🡺 to change the password information [-1] 🡺 don’t change the password and don’t lock the user id

[student@serverA ~] $ passwd 🡺 it will change the password of student

[root@serverA~]# which useradd 🡺 it will show where the command is /usr/sbin/useradd

[student@serverA~]$ cd /home/ 🡺[student@serverA home]$ ll

[root@serverA~]# userdel -r abcd 🡺 to delete user recursively

**VI Editing commands**

* Insert at cursor (goes into insert mode)
* a - Write after cursor (goes into insert mode)
* ESC-Terminate insert mode
* u - Undo last change
* dd - Delete line
* 3dd - Delete 3 lines.
* 1yy-copy 1 line
* 3yy-copy 3 lines
* p-paste
* :se nu 🡺to see line no.s
* Wc -l 🡺 to count the lines

**Saving and Closing the file**

* Shift+zz - Save the file and quit
* :w - Save the file but keep it open
* :q - Quit without saving
* :wq - Save the file and quit
* ! – forcefully

Commands for practice

[student@serverA ~]$ ls 🡺 list the content of root

[student@serverA ~]$cat file2🡺 readable mode active

[student@serverA ~]$ vimtuor 🡺 to see the commands of vim basically for help

[student@serverA ~]$ vi file2 🡺 editor open for file2

**Permissions (permissions are the foundation of security of linux)**

* -normal / regular file
* d directory
* c Character File (Printer device will be shown as char file in /dev)
* b Block File (storage devices will be shown as block file in /dev)
* 1 link or Shortcut or Softlink

rw-r—r-- 1 root root 1438 Oct 26 10:56 initial-setup-ks.cfg

r(read,4) , w(write,2) ,x (execute ,1🡺programmable files)

Onwer GroupOwner Others

rwx rwx rwx

rw- r-- r--

* (ACL+/ .)

Rwx Max permissions (7)

Users – u, Groups -g , others -o ,All -a

Change Permission - Change Ownership

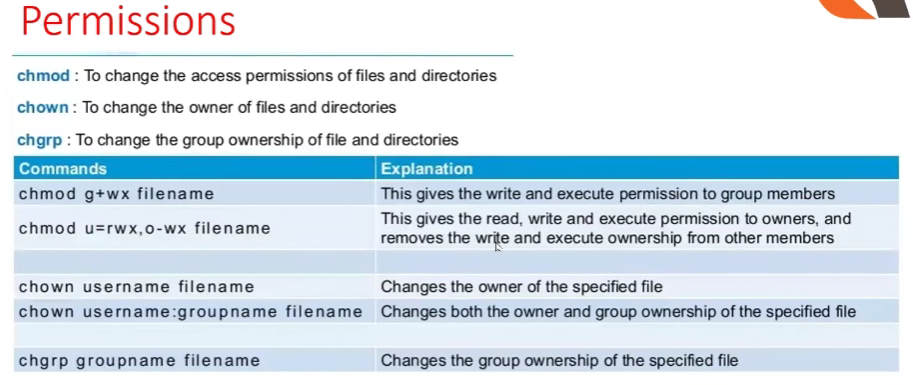
chmod u-rwx, g=---, o=--- testdirl

chmod u g o filename

chmod 7 0 0 testdirl

chmod 6 4 4 file1

chmod 7 5 0 calulator.c



[vaibhaw@serverA~]$ ll

[vaibhaw@serverA~]$ grep staff /etc/group

[vaibhaw@serverA~]$ grep finance /etc/group

[vaibhaw@serverAtemp]$ mkdir testdir1

[vaibhaw@serverAtmp]$ ls -l testdir1/

[vaibhaw@serverAtmp]$ ls – ld testdir1/

[vaibhaw@serverAtmp]$ su –

[root@serverAtmp]# cd/tmp/

[root@serverAtmp]# ls – ld testdir1/

[root@serverAtmp]# chown student:staff testdir1/

[root@serverAtmp]# chmod 750 testdir1/