unit 5

Types of user-Root and normal user

In a system, users are typically categorized into two main types based on their permissions and access levels:

1 Root User:

The root user is a superuser with full administrative rights and control over the system.

They have unrestricted access to all system resources, files, and configurations.

Root users can create, delete, and modify other user accounts, as well as change system settings and access restricted data.

This user type is often used for performing critical and high-level administrative tasks.

2 Normal User:

Normal users have limited permissions and access only to the resources and files assigned to them.

They can perform basic tasks, such as reading and writing their files, and using applications.

They do not have access to restricted or sensitive system files or the ability to make system-wide changes.

This user type is typically used for day-to-day tasks without the risk of affecting the entire system.

Each type serves different roles in system management, ensuring both security and functionality.

Multiple logins at same time (Ctrl + Alt + F1, F2..F6)

In Linux systems, pressing Ctrl + Alt + F1, F2, ..., F6 allows users to open multiple virtual terminals (or login sessions) on the same machine.

Virtual Terminals:

Linux has multiple virtual terminals, and each one can run independently, allowing different users to log in simultaneously or the same user to open multiple sessions.

By default, most Linux systems have 6 virtual terminals, accessible using Ctrl + Alt + F1 through Ctrl + Alt + F6.

Switching Between Terminals:

Each terminal is like a new login screen. When you press Ctrl + Alt + F2, for example, you'll see a login prompt where you can enter your username and password to start a new session.

You can switch back and forth between these terminals using Ctrl + Alt + Fn (where n is the terminal number).

Example Use:

Suppose you're logged into a session on F1 but want to check something else without closing your current work. You can switch to F2, log in again, and start a new task.

This is useful in situations like server management, where multiple tasks or users may need to operate at the same time.

This setup lets you multitask efficiently on a single system.

1. who Command

Purpose: Shows the users currently logged into the system.

Usage: Just type who and press Enter.

Output: Displays a list of users with details like login time and terminal.

2. whatis Command

Purpose: Gives a short description of a command.

Usage: Type whatis <command_name> (e.g., whatis ls) to see a brief description of that command.

Output: Shows a one-line summary of the command's function.

3. --help Option

Purpose: Shows a quick summary of how to use a command.

Usage: Type <command_name> --help (e.g., ls --help) for a list of options and a brief explanation.

Output: Shows the main options for the command and what each one does.

4. man Command

Purpose: Opens the manual (full documentation) for a command.

Usage: Type man <command_name> (e.g., man ls) for detailed information.

Output: Shows detailed documentation, including usage, options, and examples. Use the arrow keys to scroll and q to quit.

These commands are helpful for learning about Linux commands and getting assistance on their usage.

basic Linux commands for handling files and directories, including tasks like displaying the current directory, creating, removing, renaming, copying, and moving files or directories:

1. Displaying Current Directory and Files

pwd: Displays the current directory path.

Example: pwd

Output: /home/user

ls: Lists the files and directories in the current directory.

Example: ls

Options:

ls -l (detailed list), ls -a (includes hidden files)

cd: Changes the current directory.

Example: cd /home/user/Documents (absolute path)

Example: cd Documents (relative path)

To go up one level: cd ..

2. Creating Files and Directories

touch: Creates an empty file.

Example: touch myfile.txt

mkdir: Creates a new directory.

Example: mkdir myfolder

3. Removing Files and Directories

rm: Deletes a file.

Example: rm myfile.txt

Warning: This permanently deletes the file.

rm -r: Deletes a directory and its contents recursively.

Example: rm -r myfolder

rmdir: Deletes an empty directory.

Example: rmdir myfolder

4. Renaming Files and Directories

mv: Moves or renames a file or directory.

Example: mv oldname.txt newname.txt (renaming a file)

Example: mv myfile.txt /home/user/Documents/ (moving a file)

5. Copying Files and Directories

cp: Copies a file to another location.

Example: cp myfile.txt /home/user/Documents/

cp -r: Copies a directory and its contents.

Example: cp -r myfolder /home/user/Documents/

6. Moving Files and Directories

mv: Moves a file or directory to a new location (also used for renaming).

Example: mv myfile.txt /home/user/Documents/

comparing, editing, and displaying file contents in Linux using various commands like tr, head, tail, last, grep, sort, and piping:

1. Comparing Files

diff: Compares two files line by line and shows the differences.

Example: diff file1.txt file2.txt

Output: Displays the lines that are different between the two files.

cmp: Compares two files byte by byte. It shows the first mismatch.

Example: cmp file1.txt file2.txt

2. Editing File Content

nano: A simple text editor for editing files.

Example: nano file.txt

After opening the file, you can edit and save it by pressing Ctrl + O and Ctrl + X to exit.

vim: A powerful text editor for editing files (for advanced users).

Example: vim file.txt

You can edit, save, and exit the file using :wq (write and quit).

3. Displaying File Content

cat: Displays the entire content of a file.

Example: cat file.txt

tac: Displays the content of a file in reverse order (line-by-line).

Example: tac file.txt

tr: Translates or replaces characters in a file.

Example: cat file.txt | tr 'a-z' 'A-Z' (converts all lowercase letters to uppercase)

head: Displays the first 10 lines of a file (by default).

Example: head file.txt

You can specify a number of lines: head -n 5 file.txt (to display the first 5 lines).

tail: Displays the last 10 lines of a file (by default).

Example: tail file.txt

You can specify a number of lines: tail -n 5 file.txt (to display the last 5 lines).

tail -f: Continuously displays new lines added to the file (useful for log files).

Example: tail -f /var/log/syslog

last: Displays the login history of users.

Example: last

Shows details of who logged into the system and when.

4. Searching and Filtering File Content

grep: Searches for a pattern within a file and displays matching lines.

Example: grep 'error' logfile.txt (searches for the word "error" in logfile.txt)

Useful options:

-i (case-insensitive search): grep -i 'error' logfile.txt

-r (recursive search): grep -r 'error' /path/to/directory

egrep: Extended version of grep that supports regular expressions.

Example: egrep 'error|warning' logfile.txt (searches for lines containing "error" or "warning")

5. Sorting and Manipulating File Content

sort: Sorts the content of a file (alphabetically by default).

Example: sort file.txt

You can specify sorting options, like -n for numeric sorting: sort -n file.txt uniq: Removes duplicate lines from a file (usually used with sort).

Example: sort file.txt | uniq

6. Piping and Redirecting Output

Piping (|): Passes the output of one command as the input to another.

Example: cat file.txt | grep 'error' (searches for "error" in the content of file.txt)

Redirecting Output (>, >>):

>: Redirects output to a file (overwrites if the file already exists).

Example: echo "New content" > file.txt (writes to file.txt)

>>: Appends output to a file.

Example: echo "Additional content" >> file.txt (appends to file.txt)

Combining Commands: You can combine commands using pipes and redirection.

Example: cat file.txt | sort | uniq > sorted_unique_file.txt (sorts and removes duplicates, then saves to a new file)

Example Usage Combining Commands:

```
cat file.txt | grep 'error' | sort | uniq
```

This command will:

Display the content of file.txt.

Search for the word "error".

Sort the results.

Remove duplicate lines.

General-Purpose Utilities

Calendar and Date:

cal: Displays the calendar of the current month.

Example: cal (shows current month)

date: Shows the current date and time.

Example: date (shows the date and time)

Calculator:

bc: A simple calculator for basic arithmetic.

Example: echo "5 + 3" | bc (outputs 8)

Compression and Extraction:

tar: To create or extract compressed files (tarballs).

Example: tar -cvf archive.tar folder/ (create a tar file)

Example: tar -xvf archive.tar (extract a tar file)

gzip: Compress files using gzip.

Example: gzip file.txt (compresses the file)

gunzip: Decompress a gzip file.

Example: gunzip file.txt.gz (decompresses the file)

zip: To compress files into a zip format.

Example: zip archive.zip file1.txt file2.txt (compresses files)

unzip: To extract files from a zip archive.

Example: unzip archive.zip (extracts files)

Text Editors in Linux

nano: A simple text editor for editing files.

Open file: nano filename.txt

To save: Ctrl + O

To exit: Ctrl + X

vim: A more advanced text editor. It has two modes:

Command Mode (for navigation and commands):

Open file: vim filename.txt

To save: :w

To quit: :q

To save and quit: :wq

To undo: u

Insert Mode (for typing text): Press i to start typing, and Esc to return to Command Mode.

vi: Similar to vim, often pre-installed in Linux systems.

Same commands as vim. Use i to insert text and Esc to return to command mode.

gedit: A graphical text editor (for GNOME desktop environments).

Open file: gedit filename.txt (opens in a GUI) joe: A simple text editor with keyboard shortcuts.

Open file: joe filename.txt

Use Ctrl + K for command operations.

Basic Operations in Text Editors

Command Mode & Insert Mode (in vim/vi):

Insert Mode: Press i to start editing text.

Command Mode: Press Esc to stop editing and use commands (e.g., save, quit).

Cut, Yank (Copy), and Paste:

Cut: In nano, use Ctrl + K to cut the current line.

Yank (Copy): In vim, use yy to copy a line.

Paste: In vim, use p to paste the copied or cut content.

Undo and Redo:

Undo: In vim, use u to undo changes.

Redo: In vim, use Ctrl + r to redo.

Managing Multiple Processes

Connecting Processes with Pipes (|):

```
ls | grep '.txt'
```

This lists files and filters out the ones with the .txt extension.

tee Command:

```
echo "Hello World" | tee output.txt
```

This will output "Hello World" to the screen and also save it in the output.txt file.

Redirecting Input and Output:

Redirecting Output (>): Redirects output to a file, overwriting it.

```
echo "Hello" > file.txt
```

Redirecting Output (>>): Appends output to a file.

```
echo "World" >> file.txt
```

Redirecting Input (<): Uses the contents of a file as input to a command.

```
sort < file.txt
```

Changing Process Priority with nice:

```
nice -n 10 command
```

This runs a process with a lower priority.

cron Command:

```
crontab -e
```

To run a script at 5 PM every day, add the following line to the crontab:

```
0 17 * * * /path/to/script.sh
```

kill Command:

```
kill <PID>
```

This will terminate the process with the specified Process ID (PID).

ps Command:

```
ps aux
```

Example output:

sql

Copy code

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
user       1001  0.1  1.5 102324  4564 ?        S    14:20   0:01
/bin/bash
```

Managing User Accounts

sudo Command:

```
sudo apt-get update
```

This updates the package lists with superuser permissions.

User Management Commands:

useradd: Adds a new user.

```
sudo useradd username
```

usermod: Modifies an existing user.

```
sudo usermod -aG sudo username
```

userdel: Deletes a user.

```
sudo userdel username
```

passwd: Changes a user's password.

```
sudo passwd username
```

Group Management

Primary and Secondary Groups:

Primary Group: Each user is assigned a primary group (default group).

Secondary Groups: Users can also belong to additional groups.

Group Management Commands:

groupadd: Creates a new group.

```
sudo groupadd groupname
```

groupdel: Deletes a group.

```
sudo groupdel groupname
```

chgrp: Changes the group of a file or directory.

```
sudo chgrp groupname file.txt
```

chown: Changes the ownership of a file or directory.

```
sudo chown username:groupname file.txt
```

Managing Permissions

Adding and Removing Permissions:

chmod: Modifies file permissions.

```
chmod 755 file.txt
```

This sets read, write, and execute permissions for the owner, and read and execute for others.

Permission Structure: Permissions are represented as numbers (e.g., 755).

7: rwx (read, write, execute)

6: rw- (read, write)

5: r-x (read, execute)

4: r-- (read only)

Installing Packages:

For Debian/Ubuntu-based systems: Use apt-get to install packages.

```
sudo apt-get install package-name
```

For Red Hat/CentOS-based systems: Use yum or dnf to install packages.

```
sudo dnf install package-name
```

Graphical Interface: You can also use GUI tools like Software Center (for Ubuntu) or Yum Extender (for Fedora) to install software.