Linux Assignement-6

Name- Abinaya.P

Class/Section- 3-C

Usn- ENG24CY0075

Roll.no- 44

1) Which command is used to list the contents of a directory?

The most common command to list what's inside a directory is ls. By default, if you type ls, it will show you all the files and subdirectories in the current working directory.

Examples:

- ls → shows names of files in the current folder.
- ls -l → shows details like file permissions, owner, size, and modification date in a long list format.
- ls -a → includes hidden files (those starting with a dot).
- ls -lh → human-readable sizes (like 5K, 2M, 1G).

So, ls is not just about listing names—it can also give a lot of details depending on the options used. It's the go-to tool to "see what's in here."

---

2) Write the command to create a new directory named 123test_dir.

The command is:

```
mkdir 123test_dir
```

This makes a new folder with that name in the current location. If you want to create multiple nested directories at once (like abc/xyz), you add -p:

```
mkdir -p abc/xyz
```

That way you don’t get errors if the parent directory doesn’t exist.

---

3) What is the purpose of the sed command?

sed stands for Stream Editor. It processes text line by line and can perform operations like find-and-replace, deletion, insertion, or printing. It’s super useful for quickly editing text in files without opening them manually.

Example: Replace all occurrences of “Linux” with “Unix” in file.txt:

```
sed 's/Linux/Unix/g' file.txt
```

Here s means substitute, and g means “replace globally in each line.”

Another example: Delete line 3 from a file:

```
sed '3d' file.txt
```

So, sed is basically like a robot editor—you tell it the rule, and it applies it automatically across the file.

---

4) Which distinct command is used to display one-line descriptions of any commands?

The whatis command.
For example:

```
whatis ls
```

It’s useful when you already know the command name but want to quickly check what it does without reading the full manual.

---

5) Write the command to create an empty file named notes.txt.

There are two common ways:

```
touch notes.txt
```

or

```
> notes.txt
```

The first (touch) updates the file timestamp or creates the file if it doesn’t exist. The second way (>) creates an empty file by redirecting nothing into it. Both result in a blank file.

---

6) Differentiate between grep and awk with an example.

- grep: mainly for searching patterns inside text.
  Example:

```
grep "error" logfile.txt
```

This prints all lines in logfile.txt that contain the word “error.”

- awk: more powerful, can search but also process fields in each line.
  Example:

```
awk '{print $1, $3}' logfile.txt
```

This prints only the 1st and 3rd column from each line (useful for structured data like CSVs).

So: grep = search tool, awk = mini programming language for text processing.

---

7) Write the command to give read, write, and execute permission to the owner of a file script.sh.

```
chmod u+rwx script.sh
```

Here u means user (owner), and +rwx means add read, write, and execute. After this, the owner can do everything with the file.

---

8) How is chown different from chgrp? Give one example each.

- chown changes the owner of a file.
  Example:

```
chown alice report.txt
```

Now alice is the new owner.

- chgrp changes only the group of a file.
  Example:

```
chgrp staff report.txt
```

Now the group that has access is staff.

So, chown = who owns the file. chgrp = which group the file belongs to.

---

9) A user complains they cannot execute a file even though it exists. How to troubleshoot?

Steps:

1. Run `ls -l filename` → check permissions. If execute (x) is missing, that explains it.
2. Use `whoami` → see which user is running the command. If the file is owned by someone else and "others" have no execute permission, that user can't run it.
3. Fix with `chmod +x filename` → this adds execute permission.

So the issue usually comes down to permission bits and ownership mismatch.

---

10) Command pipeline to find all .log files modified in the last 2 days in /var/log, show them, and save results into recent_logs.txt

```
find /var/log -name "*.log" -mtime -2 | tee recent_logs.txt
```

Explanation:

- find /var/log -name "*.log" → locate all .log files.
- -mtime -2 → only files modified in the last 2 days.
- tee recent_logs.txt → prints the results on screen *and* saves them to a file.

This way you don't miss anything—you see it live, and you also have a saved copy.