LP ASSIGNMENT-6

ENG24CY0159

SHADIK KHAN

1. Which command is used to list the contents of a directory? Justify with proper example.

The go-to command to list the contents of a directory is ls, which is short for **l**i**s**t.

Its basic use is simple, but it's very powerful. For a more useful and justified example, you'll almost always use it with flags, like ls -l, which provides a **l**ong listing format.

Example:

Bash

$ ls -l

total 12

drwxr-xr-x 2 user user 4096 Oct 11 22:10 Documents

-rw-r--r-- 1 user user 512 Oct 11 22:15 notes.txt

-rwxr-xr-x 1 user user 1024 Oct 11 22:20 script.sh

This is a better example because it justifies the command's usefulness by showing not just the names, but also the permissions, owner, size, and modification date of everything in the directory.

---

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

You use the mkdir command, which stands for **m**a**k**e **dir**ectory.

Command:

Bash

mkdir 123test_dir

---

3. What is the purpose of the sed command? Justify with proper example.

Think of sed (short for **S**tream **Ed**itor) as a powerful search-and-replace robot for text. Its main purpose is to read text (from a file or from a command pipeline), perform an action on it line-by-line, and then print the result.

It's most famous for substituting text.

Example: Imagine you have a file greeting.txt that contains the line "Hello world". You want to change "world" to "Linux".

Bash

sed 's/world/Linux/' greeting.txt

Output:

Hello Linux

Here, sed reads the file and follows the instruction 's/world/Linux/', which means "**s**ubstitute the first occurrence of 'world' with 'Linux' on every line." This shows its power to transform text without you having to open the file manually.

---

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

The command you're looking for is whatis.

It's the quickest way to find out the purpose of a command without having to read its entire manual page. It searches a database of command summaries and gives you a single, descriptive line.

Example:

Bash

$ whatis chmod

chmod (1) - change file mode bits

---

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

The simplest and most common way to create an empty file is with the touch command.

Command:

Bash

touch notes.txt

This command will create a new, zero-byte file named notes.txt. If a file with that name already exists, touch will simply update its last modified timestamp to the current time.

---

6. Differentiate between grep and awk commands with an example.

While both are used for processing text, they have very different jobs.

- **grep is like a highlighter.** Its job is to search for lines that contain a specific pattern and print *only those matching lines*. It's a filter.
- **awk is like a spreadsheet program.** It's a full-fledged programming language that processes text line by line. It can not only find lines that match a pattern but can also manipulate, rearrange, and perform calculations on the *columns* (or fields) within those lines.

Example: Imagine you have a file sales.txt with the contents:

shirts pending 15.00

pants shipped 35.00

shirts shipped 20.00

- **To find all lines about "shirts" (a grep job):**

Bash

grep "shirts" sales.txt

Output:

shirts pending 15.00

shirts shipped 20.00

- **To find all items that are "shipped" and print a new sentence with their name and price (an awk job):**

Bash

awk '/shipped/ {print "Item:", $1, "Price:", $3}' sales.txt

Output:

Item: pants Price: 35.00

Item: shirts Price: 20.00

grep just finds the lines. awk finds the lines *and* does something smart with the data inside them.

---

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

The command to change file permissions is chmod. The most precise way to give the owner these permissions without affecting the group or others is to use symbolic notation.

Command:

Bash

chmod u+rwx script.sh

This command is very readable: for the **u**ser (owner), **add** (+) the **r**ead, **w**rite, and e**x**ecute permissions.

---

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

Think of a file like a work document. There's always one person who is the main **owner** and a **team** (or group) that it belongs to.

- **chown** (**ch**ange **own**er) is used to change the individual user who owns the file. It's for when you need to completely hand over the document to a new person.
- **chgrp** (**ch**ange **gr**ou**p**) is used to change the team that the file belongs to, like moving it from the 'drafting' team to the 'review' team.

Examples: Let's say a file report.txt is owned by user bob and group writers.

- To make alice the new owner:

Bash

chown alice report.txt

- To make the editors group the new group owner:

Bash

chgrp editors report.txt

---

9. A user complains that they cannot execute a file even though it exists in their directory. How would you troubleshoot this using ls -l, chmod, and whoami?

This is a classic "permission denied" issue. Here's how you'd troubleshoot it step-by-step, like a detective.

1. **Confirm Identity with whoami**: First, you need to know who you are.

Bash

$ whoami

student

Okay, we are logged in as the user student.

2. **Examine the File with ls -l**: Next, let's look at the file's permissions. Let's say the file is myscript.

Bash

$ ls -l myscript

-rw-r--r-- 1 student staff 120 Oct 11 22:55 myscript

3. **Diagnose the Problem**: We look at the output:
    - The owner of the file is student, which matches our identity. So, we should be looking at the first set of permissions (rw-).
    - The permissions for the owner are **r**ead and **w**rite, but the e**x**ecute bit (x) is missing. **That's the problem.**
4. **Apply the Fix with chmod**: Now that we know what's wrong, we can add the execute permission for the owner.

Bash

chmod u+x myscript

After running this command, the user student will be able to execute the file successfully.

---

10. Design a command pipeline to: find all .log files modified in the last 2 days in /var/log, display them on screen, and save the results into a file recent_logs.txt using tee command.

This requires us to chain together find and tee using a pipe.

Command Pipeline:

Bash

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

Here’s how the data flows through the pipeline:

1. **find /var/log -type f -name "*.log" -mtime -2**: This is the detective. It searches the /var/log directory for files (-type f) ending in .log (-name "*.log") that have been modified in the last 2 days (-mtime -2). It then produces a list of these files.
2. **|**: The pipe takes the list generated by find.
3. **tee recent_logs.txt**: It sends that list to the tee command, which acts as a clerk. It displays the list on your screen so you can see it, and at the same time, it writes a copy of that list into the file recent_logs.txt for your records.
