linux ALL assignment

💠LINUX QUALITAIVE ASSIGNMENT - 1

1. What is a text editor in Linux?

A **text editor** is a program used to **create, view, and edit text files** in Linux.

Functions:

- Create and modify files
- Insert, delete, and search text
- Edit programs and configuration files

---

2. Explain Linux VI editor and its modes

VI editor is a command-based text editor in Linux.

Modes:

- **Command mode** – navigation and editing commands
- **Insert mode** – text insertion
- **Last line mode** – save, quit, search commands

---

3. Important VI commands

- `i` – insert mode
- `dd` – delete line
- `yy` – copy line
- `:w` – save
- `:q` – quit
- `:wq` – save and quit

---

4. Standard files in Linux

Standard files handle input and output.

- **stdin (0)** – input from keyboard
- **stdout (1)** – output to screen
- **stderr (2)** – error messages

Example:

```
ls invalidfile
```

---

5. Linux filter commands

Filter commands process input data.

- **grep** – search text

```
grep "root" /etc/passwd
```

- **cut** – extract columns

```
cut -d: -f1 file
```

- **wc** – count lines/words

```
wc -l file
```

- **tr** – translate characters

```
tr a-z A-Z
```

---

6. Input and output redirection

Redirection changes input/output location.

- `<` input
- `>` output overwrite
- `>>` output append
- `2>` error redirection

Example:

```
ls > out.txt
```

---

7. Linux pipes and tee

- **Pipe (`|`)** sends output of one command to another

```
ls | wc -l
```

- **tee** displays and saves output

```
ls | tee file.txt
```

---

8. Locate files using find

The `find` command searches files.

- By name:

```
find / -name file.txt
```

- By type:

```
find / -type f
```

---

9. Linux user management

Linux supports multiple users.

Types of users:

- **Root user** – full control
- **System users** – services
- **Normal users** – regular users

---

10. File ownership and permissions

Each file has **owner, group, and others**.

Permissions:

- `r` – read
- `w` – write
- `x` – execute

Change permissions:

```
chmod 755 file
```

💠LINUX QUALITAIVE ASSIGNMENT - 2

---

1. Explain the three primary modes of the Linux VI Editor (Command mode, Insert mode, and Last Line/Ex mode). Describe the steps a user must take to switch between these modes and the specific role of the Esc key.

Answer:
The Linux **VI editor** is a powerful text editor that operates using three primary modes. Understanding these modes is essential for efficient editing.

Command Mode:
This is the default mode when a file is opened in vi. In this mode, keys perform actions like moving the cursor, deleting lines, copying text, and executing commands. Users cannot directly type text here.

Insert Mode:
Insert mode allows the user to enter and edit text normally. It can be activated from command mode by pressing keys such as

`i` (insert before cursor), `a` (append after cursor), or `o` (open a new line).

Last Line/Ex Mode:
This mode is used for file operations like saving, quitting, searching, and replacing text. It is entered by pressing `:` while in command mode.

Switching Between Modes:

- Command → Insert: press `i`, `a`, or `o`.
- Insert → Command: press the **Esc** key.
- Command → Last Line Mode: press `:`.

Role of Esc Key:
The Esc key is very important because it always brings the editor back to command mode, ensuring that commands can be executed safely.

---

2. Compare the functionality of the grep and cut commands. Provide a practical example of a "pipelined" command where you would use both to search for a specific pattern and then extract a particular column from that result.

Answer:
The `grep` and `cut` commands are commonly used Linux utilities for processing text files.

grep Command:
The grep command searches for lines containing a specific pattern or keyword within a file. It is mainly used for filtering relevant data from large files.

cut Command:
The cut command extracts specific sections of each line, such as columns or characters. It is useful when working with structured text data separated by delimiters like `:` or `,`.

Pipelining:
Linux pipes (`|`) allow the output of one command to become the input of another command. This helps perform multiple operations in a single line.

Example:

```
grep "bharath" students.txt | cut -d":" -f2
```

In this example, grep searches for lines containing "bharath," and cut extracts the second column from those results.

---

3. Explain the difference between Standard Redirection (using > and >>) and Linux Pipes (|). Specifically, describe how the tee command functions and why it is useful when you need to save output while still viewing it on the terminal.

Answer:
Standard redirection and pipes are used to manage command output in Linux.

Standard Redirection:

- `>` redirects output to a file and overwrites existing content.
- `>>` redirects output and appends it to the end of a file.

Linux Pipes:
The pipe symbol (`|`) connects two commands so that the output of the first command becomes the input for the second command without storing it in a file.

tee Command:
The `tee` command is special because it allows output to be displayed on the terminal and saved to a file at the same time.

Example:

```
ls | tee output.txt
```

This command shows the directory listing on the screen and writes it into `output.txt`. It is useful for logging outputs while monitoring them live.

---

4. Consider a file with the permission string -rwxr-x—. Convert this symbolic representation into its numeric (octal) equivalent and define the specific access rights granted to the Owner, the Group, and Others.

Answer:
Linux file permissions define who can read, write, or execute a file.

Permission string:

```
-rwxr-x---
```

Breakdown:

- **Owner:** `rwx` → read, write, execute = 7
- **Group:** `r-x` → read and execute = 5
- **Others:** `---` → no permissions = 0

Numeric (Octal) Equivalent:
`750`

This means the owner has full control, the group has limited access, and others have no access to the file.

---

5. Discuss the find command's utility. Write a command to locate all files ending in .txt within the /home directory that have been modified within the last 7 days.

Answer:
The `find` command is used to search for files and directories based on conditions such as name, type, size, and modification date. It is very powerful for managing large file systems.

Example command:

```
find /home -name "*.txt" -mtime -7
```

This command searches the `/home` directory for all files ending with `.txt` that were modified in the last 7 days. It helps administrators quickly locate recent files.

---

6. Differentiate between Local and Global (Environment) variables in a shell environment. Explain why the export command is necessary when you want a variable defined in one script to be accessible by a sub-process or child script.

Answer:
Variables in shell scripting are classified as local or global.

Local Variables:
These exist only within the current shell or script. Child processes cannot access them.

Global (Environment) Variables:
These are available to all child processes and scripts started from the current shell.

The `export` command converts a local variable into a global environment variable.

Example:

```
name="bharath"
export name
```

Without using export, the variable will not be available to sub-processes or child scripts.

---

7. Define Command Substitution. Demonstrate the syntax used to perform substitution by writing a script snippet that stores the current system date into a variable and then displays it with a custom message.

Answer:
Command substitution allows the output of a command to be stored inside a variable. It makes scripts dynamic by capturing real-time system information.

Syntax:

```
variable=$(command)
```

Example:

```
today=$(date)
echo "Current system date is: $today"
```

Here, the `date` command runs first, and its output is saved in the variable `today`.

---

8. Compare and contrast the while loop and the until loop. Explain the logical condition required for each loop to terminate and provide a short code example of an until loop.

Answer:
Both while and until loops are used to repeat commands in shell scripting.

while Loop:
Executes repeatedly as long as the condition is TRUE. The loop stops when the condition becomes false.

until Loop:
Executes repeatedly until the condition becomes TRUE. It works opposite to the while loop.

Example:

```
i=1
until [ $i -gt 5 ]
do
 echo $i
 i=$((i+1))
done
```

This loop runs until the value of `i` becomes greater than 5.

---

9. Explain the syntax and execution flow of the case statement. Describe a scenario where using a case statement is more efficient than using multiple if-then-elif-else conditions.

Answer:
The `case` statement is used for multi-way decision making in shell scripts. It checks a variable against multiple patterns and executes the matching block.

Syntax:

```
case $var in
  1) echo "One";;
  2) echo "Two";;
  *) echo "Other";;
esac
```

Execution Flow:

- The variable is compared with each pattern.
- When a match is found, the corresponding commands run.
- `;;` ends each block.

It is more efficient than many if-elif statements when creating menus or handling multiple user choices.

---

10. Analyze the impact of the break and continue statements on loop execution. How does the shell's behavior differ when a continue statement is triggered versus when a break statement is encountered?

Answer:
The `break` and `continue` statements control how loops execute in shell scripts.

continue Statement:
Skips the remaining commands of the current iteration and moves to the next cycle of the loop.

break Statement:
Immediately stops the loop and transfers control outside the loop.

Example:

```
for i in 1 2 3 4 5
do
 if [ $i -eq 3 ]; then continue; fi
 if [ $i -eq 4 ]; then break; fi
 echo $i
done
```

Here, the number 3 is skipped because of continue, and the loop stops completely when it reaches 4 due to break.

💠LINUX CBA -1 - “this should be done in word file “

Question:

Topic: Read and summarize any two articles of your choice on **any one** of the following topics:

1. Linux Distributions
2. Linux Architecture
3. Importance of Linux
4. Windows vs Linux

Requirements:

1. The summary of each article should be **at least 200 words**.
2. Mention the **article name** and the **author** you selected.
3. The **URL of the article** must also be included.

💠linux cba - 2

1. Explain the Linux File System structure with neat diagrams. Describe the purpose of the directories /bin, /etc, /home, /var and /root.

Diagram

```
/
├── bin ├── etc ├── home
├── var ├── root ├── usr
├── dev ├── tmp └── proc
```

Directory Purpose

- /bin – Basic user commands (ls, cp, mv, rm).
- /etc – System configuration files.
- /home – User home folders.
- /var – Variable data (logs, cache).
- /root – Home directory of root user.

---

2. What are Linux file naming conventions? Write the rules and give at least five valid and five invalid file name examples.

Rules

- Case-sensitive.
- Can use letters, numbers, _ and .
- Avoid special chars (? | /).
- Max length 255 chars.
- Should not start with .

Valid

```
notes.txt , data1 , my_file , info.log , script.py
```

Invalid

```
my*file , a/b.txt , -hello , name? , test|code
```

---

3. Explain the seven types of files in Linux with suitable examples and commands to identify each file type.

- Regular file – text, images → file a.txt
- Directory – folder → file mydir/
- Character device – keyboard → /dev/tty
- Block device – HDD → /dev/sda
- FIFO (pipe) → mkfifo p1
- Socket → /var/run/docker.sock
- Symbolic link → ln -s a b

---

4. Differentiate between root user, normal user and system user in Linux. How can you switch users? Give commands and examples.

- Root user – full control (UID 0).
- Normal user – limited permissions.
- System user – used by services (no login).

Switch users

```
su username
su -
sudo -i
```

---

5. Explain any five Linux directory commands (pwd, ls, cd, mkdir, rmdir) with syntax, example and output.

- pwd – show current directory → pwd → /home/user
- ls – list files → ls → Documents Downloads
- cd – change directory → cd folder
- mkdir – create folder → mkdir test
- rmdir – remove empty folder → rmdir test

---

6. Write the use, syntax and example for the following file commands: file, touch, cp, mv and rm.

```
file - show file type → file a.txt
touch - create empty file → touch new.txt
cp - copy → cp a b
mv - move/rename → mv a b
rm - delete → rm a.txt
```

---

7. What is the purpose of the Linux man command? Explain its usage, navigation keys, and give examples of checking help pages for two commands.

Purpose

- Shows the manual/help for commands.

Usage

```
man command
```

Navigation

- Space (next)
- b (back)
- / (search)
- q (quit)

Examples

```
man ls , man mkdir
```

---

8. Explain the working of the following Linux file content commands with examples and outputs: cat, tac, head, tail.

```
cat - show file → cat a.txt
tac - reverse file → tac a.txt
```

```
head - first 10 lines → head a.txt
tail - last 10 lines → tail a.txt
```

---

9. What is the Linux Filesystem Hierarchy Standard (FHS)? Explain the purpose of the directories /usr, /dev, /tmp, /proc, /mnt and /media.

FHS

- Defines how Linux folders are organized.
- /usr – user apps, libraries
- /dev – device files
- /tmp – temporary files
- /proc – system info (virtual FS)
- /mnt – temporary mount point
- /media – USB, CD auto-mount

---

10. Differentiate between the following (any four):

a) cat vs tac

- cat → normal order
- tac → reverse order

b) cp vs mv

- cp → copy
- mv → move/rename

c) rm vs rmdir

- rm → removes files
- rmdir → removes empty folders

d) /home vs /root

- /home → normal user homes
- /root → root user home

e) /bin vs /sbin

- /bin → basic commands
- /sbin → system/admin commands

💠LINUX CBA - 3 -this should be submitted in “PDF FORMAT “

Category 1: Command Line Mastery (Filters & Pipes)

Choose one of the following tasks:

• Option A (The Data Miner): You have a large log file named access.log. Write a single-command pipeline that finds all lines containing the word “ERROR”, extracts only the 3rd column (the timestamp), and saves the count of these errors to a file named report.txt.

• Option B (The Cleanup Crew): You have a list of names in users.txt that are messy (mixed case, extra spaces). Use a combination of tr, cut, and wc to convert all names to lowercase and provide a total count of users in the list.

Category 2: The VI Power User

Choose one of the following tasks:

• Option A (Efficiency Challenge): List the specific VI command sequences required to:

1. Delete 5 lines of text starting from the cursor.
2. Undo that deletion.
3. Search for the word “Configuration” and replace it with “Setup” globally.

• Option B (The Editor’s Logic): Explain the functional difference between Command Mode, Insert Mode, and Last Line Mode. Why does Linux use a modal editor like VI instead of a standard “point-and-click” interface for system administration?

Category 3: Security & User Management

Choose one of the following tasks:

• Option A (The Permissions Audit): You encounter a file with the permissions -rwxr-xr–.

1. Translate this into octal (numeric) notation.
2. Identify what the “Others” group can and cannot do.
3. Provide the command to change this so the “Owner” and “Group” have full access, but “Others” have no access at all.

• Option B (The Admin’s Script): Outline the steps (and commands) to create a new user named dev_user, assign them to a group called developers, and ensure they own the directory /home/project_alpha.

Category 4: System Navigation & Redirection

Choose one of the following tasks:

• Option A (The Search Party): Use the find command to locate all files ending in .sh within the /etc directory that are larger than 10k. How would you redirect any “Permission Denied” errors to /dev/null so they don’t clutter your screen?

• Option B (The Tee Connection): Explain a scenario where using the tee command is superior to using a simple > redirection. Provide a code example demonstrating its use in a real-world system update or log monitoring task.

Submission Rules:

1. Sumit the assessment in PDF format.
2. For each category choose one option.
3. PDF should contain the screenshot/snapshots of the commands executed for each category followed by the answer to the question asked in each option.

💠LINUX CBA - 4 — this should be submitted in “pdf format “

Choice Based Assessment – 4

Category 1: Variables & Environment Logic

Focus: Variable Scope and Substitution

• **Option A (The Scope Experiment):** Create a short script that demonstrates the difference between a Local variable and an Exported (Global) variable. Explain what happens to the variable when you call a sub-shell or another script.

• **Option B (The Substitution Task):** Write a command using Command Substitution (backticks or $( )) that creates a directory named with the current system date and the current logged-in user (e.g., folder_2026-01-30_student).

Category 2: Conditional Logic & Branching

Focus: If/Else and Case Statements

• **Option A (The System Guard):** Write a Shell Script snippet using if-else that checks if a specific file (e.g., /etc/passwd) exists and is readable. If it is, print **"Security Check Passed"**; if not, print **"Alert"** and exit.

• **Option B (The Menu Builder):** Use a case statement to create a simple menu for a user. The user should be able to press **1** to see the current uptime, **2** to see disk usage, and **3** to exit. Why is case often preferred over multiple if-elif blocks for menus?

Category 3: Iteration & Loop Control

Focus: For, While, and Until Loops

• **Option A (The Bulk Processor):** Write a for loop that iterates through all `.txt` files in the current directory and renames them to `.bak` files.

• **Option B (The Monitoring Loop):** Write a script using a while or until loop that checks for the existence of a file named `stop.txt`. The script should print **"Monitoring..."** every 5 seconds and only stop once the file `stop.txt` is created by the user.

Category 4: Script Optimization (Loop Control)

Focus: Break and Continue

• **Option A (The Skip Logic):** Write a loop that counts from 1 to 10 but uses the **continue** statement to skip the number 5.

• **Option B (The Early Exit):** Write a loop that processes a list of items but uses the **break** statement to stop immediately if it encounters an item named **"DANGER"**.

Submission Rules

1. Sumit the assessment in PDF format.
2. For each category choose one option.
3. PDF should contain the screenshot/snapshots of the scripts executed for each category.
