Assignment 2

Part A

What will the following commands do?

1. echo "Hello, World!"

- The echo command **prints** text to the terminal.
- The text inside "" (double quotes) is displayed as output

Option	Description	Example	Output
echo “Enter Text”	Print the given Text.	echo "Hello, World!"	Hello, World!
echo -n "text"	Prints text **without a newline**.	echo -n "No New Line"	No New Line (cursor stays on the same line)
echo -e "text"	Enables **escape sequences** like \n (new line) and \t (tab).	echo -e "Line1\nLine2"	Line1 Line2
echo -E "text"	Disables escape sequences (default behavior).	echo -E "Hello\nWorld"	Hello\nWorld (does not interpret \n)
echo -e "Col1\tCol2"	Adds **tabs** between words.	echo -e "Name\tAge"	Name Age

2. name="Productive"

- This **assigns** the string "Productive" to the variable name in **Bash**.
- **No spaces are allowed** around = (i.e., **name = "Productive"** - would cause an error).

Option	Description	Example	Output
name="RaviRaj"	Assigns a value to a variable (**no spaces around** =).	echo "$name"	RaviRaj
$variable	Accesses the **value** of a variable.	echo "Hello, $name"	Hello, RaviRaj
${variable}	Alternative way to use a variable.	echo "${name}123"	RaviRaj123

unset variable	Deletes the variable from memory.	unset name	(Variable is removed)
readonly variable	Makes a variable **read-only** (cannot be changed).	readonly age=30	(Now age cannot be modified)
export variable	Makes the variable **available to child processes**.	export PATH=$PATH:/new/path	Adds /new/path to PATH

3. touch file.txt
    - The touch command is used to **create an empty file** or **update the timestamp** of an existing file.
    - If file.txt **does not exist**, touch creates it.
    - If file.txt **already exists**, touch updates its **last modified timestamp** without changing its contents.

Option	**Description**	**Example**
touch file.txt	Creates a **new empty file** or updates the timestamp.	touch newfile.txt
touch file1 file2	Creates multiple files.	touch a.txt b.txt
touch -c file.txt	**Does not create** the file if it doesn't exist.	touch -c missing.txt
touch -m file.txt	Updates **only the modification time**, not access time.	touch -m example.txt
touch -a file.txt	Updates **only the access time**, not modification time.	touch -a readme.txt
touch -t YYYYMMDDhhmm file.txt	Sets a **specific timestamp**.	touch -t 202403011200 file.txt
touch -r old.txt new.txt	Copies timestamp from another file.	touch -r ref.txt copy.txt

4. ls -a
    - The ls command lists **files and directories** in the current directory.
    - The -a (**all**) option **shows hidden files** (files starting with .).
    - In Linux, hidden files are **configuration files** (e.g., .bashrc, .gitignore).

File/Directory	**Meaning**
.	The **current directory**
..	The **parent directory**
.bashrc	A hidden configuration file
.profile	Another hidden file
document.txt	A **regular file**
folder/	A **directory**

5. rm file.txt

   Explanation

   - The rm (**remove**) command **deletes a file** in Linux.
   - Once deleted, **it cannot be recovered** unless you have a backup.
   - If file.txt is **write-protected**, it will ask for **confirmation** before deleting.

   Command Options

Option	**Description**	**Example**
rm file.txt	Deletes a file.	rm myfile.txt
rm -i file.txt	Asks before deleting.	rm -i important.txt
rm -f file.txt	Force delete **without confirmation**.	rm -f log.txt
rm -v file.txt	Shows **verbose output** (what's deleted).	rm -v oldfile.txt
rm *.txt	Deletes **all .txt files** in the directory.	rm *.log

6. cp file1.txt file2.txt.
   - The cp (**copy**) command is used to **copy files and directories**.
   - file1.txt is copied to file2.txt, creating a **duplicate**.
   - If file2.txt already exists, it will be overwritten without confirmation (unless -I is used).

Option	**Description**	**Example**
cp file1 file2	Copies a file.	cp a.txt b.txt
cp -i file1 file2	**Asks before overwriting**.	cp -i report.doc backup.doc
cp -v file1 file2	**Shows what's being copied**.	cp -v notes.txt copy.txt
cp -r dir1 dir2	Copies a **directory** recursively.	cp -r src/ backup/
cp -u file1 file2	**Copies only if the destination is older**.	cp -u data.txt backup.txt

7. mv file.txt /path/to/directory/
   - the mv (**move**) command is used to **move or rename files and directories**.
   - This command **moves** file.txt from its current location to /path/to/directory/.
   - If a file **with the same name** exists in /path/to/directory/, it **will be overwritten without confirmation** (unless -i is used).

Option	**Description**	**Example**	
mv file.txt /path/	Moves a file to a directory.	mv report.txt /home/user/	
mv oldname.txt newname.txt	Renames a file.	mv notes.txt summary.txt	
mv -i file.txt /path/	**Asks before overwriting** if the file exists.	mv -i data.txt /backup/	
mv -v file.txt /path/	**Shows what's being moved**.	mv -v log.txt /var/logs/	
mv -u file.txt /path/	**Moves only if the destination is older**.	mv -u script.sh /scripts/	

8. chmod 755 script.sh
   - The chmod (**change mode**) command is used to **modify file permissions** in Linux.
   - 755 sets **read, write, and execute** permissions for the **owner**, and **read and execute** permissions for **group and others**.
   - This is commonly used to make script.sh executable for everyone.

Understanding 755 Permissions

User	**Permissions (755)**	**Explanation**
Owner	rwx (4,2,1)	Read, Write, Execute
Group	r-x (4,0,1)	Read & Execute (No Write)
Others	r-x (4,0,1)	Read & Execute (No Write)

9. grep "pattern" file.txt

   - The grep (**Global Regular Expression Print**) command **searches for a specific pattern** in file.txt.
   - It prints **all lines** in file.txt that contain "pattern".

Option	**Description**	**Example**
grep "text" file.txt	Searches "text" in file.txt.	grep "error" log.txt

grep -i "text" file.txt	**Case-insensitive** search.	grep -i "warning" syslog
grep -w "word" file.txt	Matches **whole words only**.	grep -w "root" users.txt
grep -n "text" file.txt	**Shows line numbers**.	grep -n "failed" auth.log
grep -l "text" *.txt	Lists **files containing text**.	grep -l "TODO" *.sh
grep -r "text" dir/	Searches recursively in **all files** inside a directory.	grep -r "ERROR" /var/logs/
grep -A 2 "text" file.txt	Shows **2 lines after** the match.	grep -A 2 "404" logs.txt
grep -B 2 "text" file.txt	Shows **2 lines before** the match.	grep -B 2 "error" logs.txt
grep -C 2 "text" file.txt	Shows **2 lines before & after**.	grep -C 2 "error" logs.txt

10. kill PID
    - The kill command **terminates a process** using its **Process ID (PID)**.
    - PID is the **unique ID assigned** to a running process in Linux.
    - By default, kill PID sends **signal 15 (SIGTERM)** to **gracefully terminate** the process.

Signal	Number	Description	Usage Example
SIGTERM	15	**Gracefully stops** a process.	kill 1234
SIGKILL	9	**Force kills** a process immediately.	kill -9 1234
SIGHUP	1	**Reloads** a process.	kill -1 1234
SIGSTOP	19	**Pauses** a process.	kill -19 1234
SIGCONT	18	**Resumes** a paused process.	kill -18 1234

11. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

    Explanation

    This is a **chained command** using &&, meaning:
    - Each command runs **only if the previous command succeeds**.
    - It **creates a directory**, moves into it, creates a file, writes text, and displays the file content.

Step-by-Step Breakdown

Command	Explanation
mkdir mydir	Creates a directory named mydir.
cd mydir	Moves into mydir.
touch file.txt	Creates an empty file named file.txt.
echo "Hello, World!" > file.txt	Writes "Hello, World!" into file.txt.
cat file.txt	Displays the content of file.txt.

12. ls -l | grep ".txt"
    - ls -l → Lists **all files and directories** in **long format** (including permissions, size, date).
    - grep ".txt" → **Filters** the output to **show only files containing .txt** in their names.
13. cat file1.txt file2.txt | sort | uniq
    - This command **combines the contents** of file1.txt and file2.txt, **sorts** them, and removes **duplicate lines**.

Command	Description
cat file1.txt file2.txt	Displays the content of both files.
sort	Sorts the combined output (required for uniq to work).
uniq	Removes **duplicate lines** (only works on adjacent lines).

14. ls -l | grep "^d"
    - ls -l → Lists **files and directories** in **long format**.
    - grep "^d" → Filters **only directories**.

Pattern	Explanation
^	Matches **the beginning of the line**.
d	Indicates a **directory** in ls -l output.

15. grep -r "pattern" /path/to/directory/
    - grep → Searches for a specific **pattern** in files.
    - -r (**recursive**) → Searches inside all **files and subdirectories**.
    - "pattern" → The **text or keyword** you want to find.
    - /path/to/directory/ → The **directory** where the search starts.

16. cat file1.txt file2.txt | sort | uniq –d
    - cat file1.txt file2.txt → **Concatenates** (combines) the contents of file1.txt and file2.txt.
    - sort → **Sorts** the lines **alphabetically** (required for uniq to work correctly).
    - uniq -d → **Displays only duplicate lines** (lines that appear in both files).
17. chmod 644 file.txt
    The chmod (**change mode**) command is used to **modify file permissions** in Linux.
    - 644 sets:
        - **Owner:** rw- (**Read & Write**)
        - **Group:** r-- (**Read-Only**)
        - **Others:** r-- (**Read-Only**)

    This means:
    - The **owner** of file.txt can **read and write** the file.
    - **Group members and others** can **only read** the file.
    - **No one except the owner can modify the file.**

    Understanding 644 Permissions

User	Permissions (644)	Explanation
Owner	rw- (4,2,0)	Read, Write (No Execute)
Group	r-- (4,0,0)	Only-Read
Others	r-- (4,0,0)	Only-Read

18. cp -r source_directory destination_directory
    - The cp (**copy**) command is used to **copy files and directories**.
    - The -r (**recursive**) option ensures that **all files and subdirectories** inside source_directory are copied to destination_directory.
    - If destination_directory **does not exist**, it will be **created automatically**.

Option	Description	Example
cp -r source dest	**Copies a directory recursively**.	cp -r mydir /backup/

cp -rv source dest	**Verbose mode** (shows copied files).	cp -rv mydir /backup/
cp -rn source dest	**Does not overwrite existing files**.	cp -rn mydir /backup/
cp -rp source dest	**Preserves file attributes**.	cp -rp mydir /backup/
cp -u source dest	**Copies only newer files**.	cp -ru mydir /backup/

19. ind /path/to/search -name "*.txt"
    - The find command is used to **search for files and directories** in Linux.
    - /path/to/search → Specifies the **starting directory** for the search.
    - -name "*.txt" → Finds **files ending with .txt** (case-sensitive).

Option	**Description**	**Example**
find /path -name "*.txt"	**Finds all .txt files** (case-sensitive).	find /data -name "*.txt"
find /path -iname "*.txt"	**Finds .txt files (case-insensitive)**.	find /data -iname "*.TXT"
find /path -type d -name "folder"	Finds a **directory** named "folder".	find / -type d -name "backup"
find /path -size +10M -name "*.log"	Finds .log files **larger than 10MB**.	find /var/log -size +10M -name "*.log"
find /path -mtime -7 -name "*.txt"	Finds .txt files **modified in the last 7 days**.	find /home -mtime -7 -name "*.txt"
find /path -empty -type f	Finds **empty files**.	find /tmp -empty -type f
find /path -exec command {} \;	Executes a command on found files.	find /logs -name "*.log" -exec rm {} \;

20. chmod u+x file.txt
    - The chmod (**change mode**) command is used to **modify file permissions**.
    - u+x means:
        - **u (user/owner)** → Applies to the **file owner**.
        - **+x (add execute)** → Adds **execute** permission to the owner.
    - This allows the owner to **run the file as a script or program**.

chmod u+x vs. Other Permission Modifications

Command	**Effect**
chmod u+x file.txt	Adds **execute** permission for the **owner**.
chmod g+x file.txt	Adds **execute** permission for the **group**.
chmod o+x file.txt	Adds **execute** permission for **others**.
chmod a+x file.txt	Adds **execute** permission for **everyone**.

chmod u-x file.txt	Removes **execute** permission from the **owner**.

21. echo $PATH
    - The $PATH variable stores a **list of directories** where the system looks for executable files.
    - Running echo $PATH **displays the current search paths** for executables.

Part	Description	
/usr/local/sbin	Stores system administrator commands.	
/usr/local/bin	Stores locally installed programs.	
/usr/bin	Stores common system binaries (e.g., ls, grep).	
/bin	Stores essential user binaries (e.g., cat, echo).	
/home/user/scripts	Custom scripts added by the user.	