ASSIGNMENT – 2

Lakshit Dogra - KH

PART - A

Q. What will the following commands do?

Ans: The following commands will do/perform as mentioned: -

1. **echo "Hello, World!"**
   This command prints the text "Hello, World!" to the terminal.
2. **name="Productive"**
   This sets a variable name with the value "Productive" in the current shell session.
3. **touch file.txt**
   This command creates an empty file named file.txt. If the file already exists, it updates the timestamp of the file.
4. **ls -a**
   This lists all files in the current directory, including hidden files (those starting with a dot).
5. **rm file.txt**
   This removes (deletes) the file file.txt.
6. **cp file1.txt file2.txt**
   This copies the contents of file1.txt to file2.txt. If file2.txt exists, it will be overwritten.
7. **mv file.txt /path/to/directory/**
   This moves the file.txt to the specified directory (/path/to/directory/). If the file exists in the destination, it will overwrite it.
8. **chmod 755 script.sh**
   This sets the permissions of the file script.sh to rwxr-xr-x, meaning the owner can read/write/execute, and others can read/execute.
9. **grep "pattern" file.txt**
   This searches for the specified pattern (pattern) in the file file.txt and prints the matching lines.

10. **kill PID**

    This sends a signal to the process with the given PID (Process ID) to terminate it.

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

    Creates a directory called mydir. Changes the current directory to mydir. Creates an empty file file.txt. Writes "Hello, World!" to file.txt. Displays the content of file.txt, which will be "Hello, World!".

12. **ls -l | grep ".txt"**

    This lists all files in long format (ls -l) and then filters the output to only show files that contain .txt in their name.

13. **cat file1.txt file2.txt | sort | uniq**

    This concatenates file1.txt and file2.txt, sorts the combined content, and then filters out duplicate lines using uniq.

14. **ls -l | grep "^d"**

    This lists files in long format and filters out only directories (lines starting with d).

15. **grep -r "pattern" /path/to/directory/**

    This recursively searches for the specified pattern in all files under the given directory /path/to/directory/.

16. **cat file1.txt file2.txt | sort | uniq -d**

    This concatenates file1.txt and file2.txt, sorts them, and then shows only the duplicate lines using uniq -d.

17. **chmod 644 file.txt**

    This sets the permissions of file.txt to rw-r--r--, meaning the owner can read/write, while others can only read.

18. **cp -r source_directory destination_directory**

    This copies the entire contents of source_directory (including subdirectories) to destination_directory.

19. **find /path/to/search -name "*.txt"**

    This searches the specified path (/path/to/search) and lists all files with the .txt extension.

20. **chmod u+x file**

    This adds execute permission for the user (owner) on the file file.

PART - B

Q. Identify True or False:

Ans.

1. ls is used to list files and directories in a directory. **True**

2. mv is used to move files and directories. **True**

3. cd is used to copy files and directories. **False**

4. pwd stands for "print working directory" and displays the current directory. **True**

5. grep is used to search for patterns in files. **True**

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. **True**

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist. **True**

8. rm -rf file.txt deletes a file forcefully without confirmation. **True**

Q. Identify the Incorrect Commands:

1. chmodx is used to change file permissions. **Incorrect**

2. cpy is used to copy files and directories. **Incorrect**

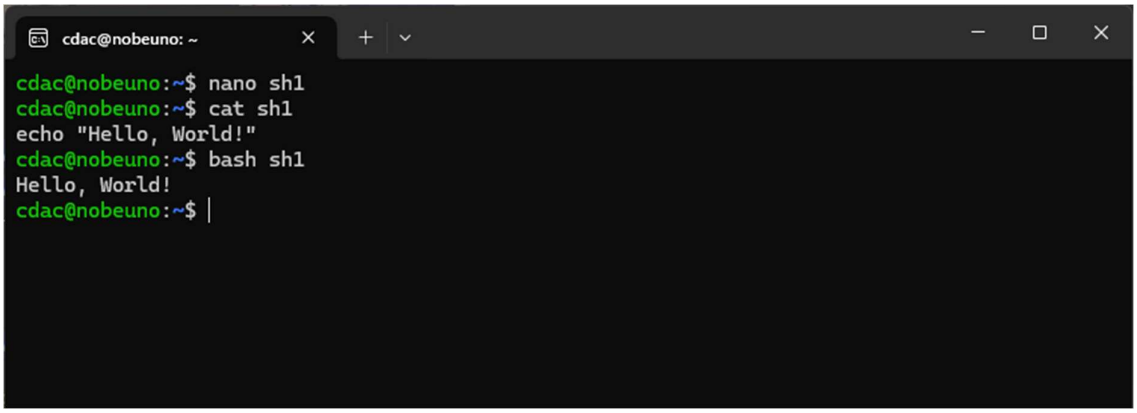
3. mkfile is used to create a new file. **Incorrect**

4. catx is used to concatenate files. **Incorrect**

5. rn is used to rename files. **Incorrect**

PART - C

Question 1: Write a shell script that prints "Hello, World!" to the terminal.



Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh2
cdac@nobeuno:~$ cat sh2
name="CDAC Mumbai"
echo $name
cdac@nobeuno:~$ bash sh2
CDAC Mumbai
cdac@nobeuno:~$
```

Question 3: Write a shell script that takes a number as input from the user and prints it.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh3
cdac@nobeuno:~$ cat sh3
read -p "enter a number: " num
echo "you entered $num"
cdac@nobeuno:~$ bash sh3
enter a number: 7
you entered 7
cdac@nobeuno:~$
```

Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh4
cdac@nobeuno:~$ cat sh4
a=5
b=3
sum=$((a+b))
echo "sume is $sum"
cdac@nobeuno:~$ bash sh4
sume is 8
cdac@nobeuno:~$
```

Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh5
cdac@nobeuno:~$ cat nano sh5
cat: nano: No such file or directory
read -p "enter a number: " num
if ((num%2==0)); then
        echo "even"
else
        echo "odd"
fi
cdac@nobeuno:~$ bash sh5
enter a number: 7
odd
cdac@nobeuno:~$ bash sh5
enter a number: 6
even
cdac@nobeuno:~$
```

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh6
cdac@nobeuno:~$ cat sh6
for i in {1..5}; do
        echo $i
done
cdac@nobeuno:~$ bash sh6
1
2
3
4
5
cdac@nobeuno:~$
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh7
cdac@nobeuno:~$ cat sh7
i=1
while [ $i -le 5 ]; do
        echo $i
        ((i++))
done
cdac@nobeuno:~$ bash sh7
1
2
3
4
5
cdac@nobeuno:~$
```

Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh8
cdac@nobeuno:~$ cat sh8
if [ -f "file.txt" ]; then
        echo "file exists"
else
        echo "file does not exist"
fi
cdac@nobeuno:~$ bash sh8
file does not exist
cdac@nobeuno:~$ ls
LinuxAssignments  sh1  sh2  sh3  sh4  sh5  sh6  sh7  sh8
cdac@nobeuno:~$
```

Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh9
cdac@nobeuno:~$ cat sh8
if [ -f "file.txt" ]; then
        echo "file exists"
else
        echo "file does not exist"
fi
cdac@nobeuno:~$ bash sh9
enter a number: 7
number is 10 or less
cdac@nobeuno:~$ bash sh9
enter a number: 88
number is greater than 10
cdac@nobeuno:~$
```

Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

```
cdac@nobeuno: ~
cdac@nobeuno:~$ nano sh10
cdac@nobeuno:~$ nano sh10
cdac@nobeuno:~$ cat sh10
for num in {1..5}; do
        echo "multiplication table for $num is"
        for i in {1..10}; do
                echo "$num x $i = $((num * i))"
        done
        echo ""
done

cdac@nobeuno:~$ bash sh10
multiplication table for 1 is
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

```
cdac@nobeuno: ~
multiplication table for 2 is
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

multiplication table for 3 is
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```

```
cdac@nobeuno: ~

multiplication table for 4 is
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

multiplication table for 5 is
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.

```
cdac@nobeuno: ~

cdac@nobeuno:~$ nano sh11
cdac@nobeuno:~$ cat sh11
while true; do
        read -p "enter a number: " num
        if [ $num -lt 0 ]; then
                break
        fi
        echo "square: $((num * num))"
done
cdac@nobeuno:~$ bash sh11
enter a number: 5
square: 25
enter a number: 7
square: 49
enter a number: 4
square: 16
enter a number: 0
square: 0
enter a number: 6
square: 36
enter a number: -1
cdac@nobeuno:~$
```

PART – E

Q. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

Q1 Calculate average using FCFS Scheduling:-

Process	Arrival	Burst	CT	WT	TAT
P_1	0	5	5	0	5
P_2	1	3	8	4	7
P_3	2	6	14	6	12

Gantt Chart

P_1	P_2	P_3	
0	5	8	14

$\therefore$ Average waiting time $= \dfrac{(P_1 + P_2 + P_3)\ \text{Waiting Time}}{\text{Total Processes}}$.

$= \dfrac{0+4+6}{3} = 0.33$.

Q. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Q Calculate the average turn-around time using SJF

Process	Arrival	Burst	CT	RT	WT	TAT
P_1	0	3	3	0	0	3
P_2	1	5	13	7	7	12
P_3	2	1	4	1	1	2
P_4	3	4	8	1	1	5
						5.5 Avg.

Gant Chart

P_1	P_3	P_4	P_2

0 3 4 8 13

$$\text{Average TAT} = \frac{3+12+2+5}{4}$$

$$= \frac{22}{4} = 5.5$$

Q. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

Q Calculate the average waiting time using Priority Scheduling.

Process	Arrival	Burst	Priority	CT	TAT	WT	RT
P_1	0	6	3	13	13	7	0
P_2	1	4	1	5	4	0	0
P_3	2	7	4	20	18	11	11
P_4	3	2	2	7	4	2	2
				Avg.	9.75	5	3.25

Gannt Chart	P_1	P_2	P_2	P_2	P_2	P_4	P_4	P_1	P_1	P_1	P_1	P_1	P_1	..
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

...	P_3	P_3	P_3	P_3	P_3	P_3	P_3
	14	15	16	17	18	19	20

Average Waiting Time $= \frac{7+0+11+2}{4}$

$= \frac{20}{4} = 5$