

## Part A

What will the following commands do?

- **echo "Hello, World!"**

Prints Hello, World! to the terminal.

- **name="Productive"**

Creates a variable name and assigns it the value Productive

- **touch file.txt**

Creates an empty file named file.txt or updates its timestamp if it already exists

- **ls -a**

Lists all files and directories in the current directory, including hidden ones (those starting with . )

- **rm file.txt**

Removes the file file.txt permanently.

- **cp file1.txt file2.txt**

Copies file1.txt to file2.txt . If file2.txt exists, it will be overwritten.

- **mv file.txt /path/to/directory/**

Moves file.txt to the specified directory.

- **chmod 755 script.sh**

Grants the owner full permissions (read, write, execute) and gives others read and execute permissions on script.sh

- **grep "pattern" file.txt**

Searches for occurrences of "pattern" in file.txt and prints matching lines.

- **kill PID**

Terminates the process with the specified Process ID (PID)

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

Creates a directory mydir

Changes into mydir Creates an empty file file.txt

Writes "Hello, World!" into file.txt

Displays the contents of file.txt

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

Lists files in long format and filters only those containing ".txt" in their names

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

Concatenates file1.txt and file2.txt , sorts them, and removes duplicate lines

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

Lists directories (entries starting with d in long format output).

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

Searches for "pattern" recursively in all files under /path/to/directory/ .

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

Concatenates file1.txt and file2.txt , sorts them, and displays only duplicate lines

- **chmod 644 file.txt**

Grants the owner read and write permissions, while others get read-only access to file.txt .

- **cp -r source\_directory destination\_directory**

Recursively copies source\_directory to destination\_directory , preserving contents.

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

Finds all .txt files in /path/to/search and its subdirectories.

- **chmod u+x file.txt**

Gives the owner ( u ) execute permission on file.txt

- **echo \$PATH**

Displays the system's PATH environment variable, listing directories where executable files are searched for.

## Part B

### Identify True or False:

**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 with**

True

### Identify the Incorrect Commands:

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

Incorrect - chmodx is not a valid command. The correct command to change file permissions is chmod .

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

Incorrect - cpy is not a valid command. The correct command to copy files and directories is cp .

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

Incorrect - mkfile is not a standard Linux command. To create a new file, use filename .

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

Incorrect - touch catx is not a valid command. The correct command to concatenate files is cat .

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

Incorrect - **rn** is not a valid command. To rename files, use the **mv** command ( oldname newname )

## Part C

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

```
cdac@DESKTOP-P1QG0DM: ~  
cdac@DESKTOP-P1QG0DM:~$ mkdir Assignment2  
cdac@DESKTOP-P1QG0DM:~$ echo "Hello, World!"  
Hello, World!  
cdac@DESKTOP-P1QG0DM:~$
```

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

```
CDAC Mumbai  
cdac@DESKTOP-P1QG0DM:~$ name="CDAC Mumbai"  
cdac@DESKTOP-P1QG0DM:~$ echo $name  
CDAC Mumbai  
cdac@DESKTOP-P1QG0DM:~$ |
```

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

```
numbers are:  
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano number.sh  
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash number.sh  
Enter the numbers  
12 34 45  
numbers are: 12 34 45  
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ cat number.sh  
echo "Enter the numbers"  
read number  
  
echo "numbers are: $number"  
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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

```
cdac@DESKTOP-P1QG0DM: ~ X + v
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano addition.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash addition.sh
Enter the first number:
5
Enter the second number:
3
The sum of 5 and 3 is: 8
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ cat addition.sh
echo "Enter the first number: "
read num1

echo "Enter the second number: "
read num2

sum=$(expr $num1 + $num2)

echo "The sum of $num1 and $num2 is: $sum"
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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

```
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano evenodd.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash evenodd.sh
Enter the number
23
Number is odd
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ 22
22: command not found
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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

```
cdac@DESKTOP-P1QG0DM: ~ X + v
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ forloop.sh
forloop.sh: command not found
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ touch forloop.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano forloop.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash forloop.sh
1
2
3
4
5
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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

```
5
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ touch whileloop.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano whileloop.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash whileloop.sh
0
1
2
3
4
5
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano whileloop.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash whileloop.sh
0
1
2
3
4
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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@DESKTOP-P1QG0DM: ~ X + v
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ touch file2.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano file2.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash file2.sh
File does not exist
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ cat file2.sh
if [ -f "file.txt" ]
then
    echo "File exists"
else
    echo "File does not exist"
fi
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano file2.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash file2.sh
File exists
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ cat file2.sh
if [ -f "file1.txt" ]
then
    echo "File exists"
else
    echo "File does not exist"
fi
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

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@DESKTOP-P1QG0DM: ~ X + v
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ touch greater.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ nano greater.sh
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ bash greater.sh
Enter a number:
45
The number is greater than 10.
cdac@DESKTOP-P1QG0DM:~/feb25/assignment2$ |
```

**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.

```
5 | $((i * j))$((i * j))$((i * j))$((i * j))$((i * j))
cdac@DESKTOP-PIQG0DM:~/feb25/assignment2$ nano multiplication.sh
cdac@DESKTOP-PIQG0DM:~/feb25/assignment2$ bash multiplication.sh
Multiplication Table (1 to 5)
1 | 12345
2 | 246810
3 | 3691215
4 | 48121620
5 | 510152025
cdac@DESKTOP-PIQG0DM:~/feb25/assignment2$ cat multiplication.sh
echo "Multiplication Table (1 to 5)"

for i in {1..5}
do
    echo -n "$i | "
    for j in {1..5}
    do
        echo -n $((i * j))
    done
    echo
done
cdac@DESKTOP-PIQG0DM:~/feb25/assignment2$ |
```

**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@DESKTOP-PIQG0DM: ~  
cdac@DESKTOP-PIQG0DM:~$ nano readnumbers.sh  
cdac@DESKTOP-PIQG0DM:~$ bash readnumbers.sh  
Enter a number: 10  
Square: 100  
Enter a number: 45  
Square: 2025  
Enter a number: 67  
Square: 4489  
Enter a number:
```

## Part E

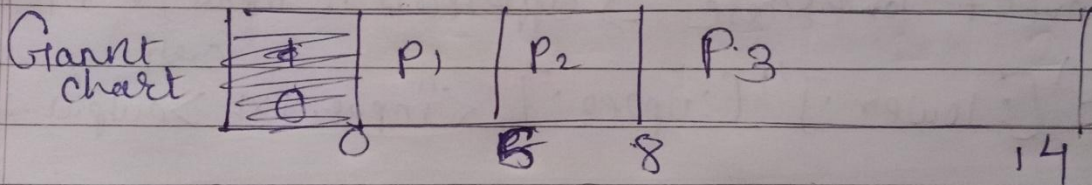
Q. Consider the following processes with arrival times & burst times:

Process	Arrival time	Burst time
P <sub>1</sub>	0	5
P <sub>2</sub>	1	3
P <sub>3</sub>	2	6

Calculate the average waiting time using FCFS.



Process	AT	BT	CT	RT	WT	TAT
P <sub>1</sub>	0	5	5	0	0	5
P <sub>2</sub>	1	3	8	4	4	7
P <sub>3</sub>	2	6	14	6	6	12
				3.33	3.33	8



$$\text{Average waiting time} = \frac{0 + 4 + 6}{3} = 3.33$$

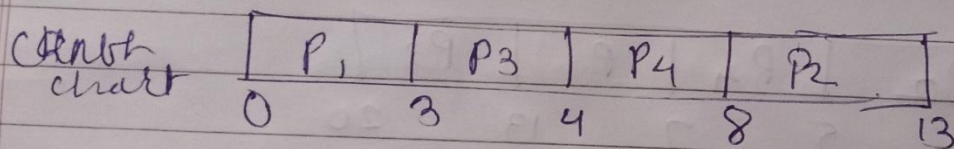


Q-2 Calculate the average turnaround time using shortest job first (SJF) scheduling.

Process	Arrival time	Burst time
P <sub>1</sub>	0	3
P <sub>2</sub>	1	5
P <sub>3</sub>	2	1
P <sub>4</sub>	3	4

Process	AT	BT	CT	RT	WT	TAT
P <sub>1</sub>	0	3	3	0	0	3
P <sub>2</sub>	1	5	13	7	7	12
P <sub>3</sub>	2	1	4	1	1	7
P <sub>4</sub>	3	4	8	1	1	5

P<sub>1</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>2</sub>



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

$$= 5.5$$



3) Consider the following process with arrival time, burst time & priorities (lower no. indicates higher priority).

Process	AT	BT	Priority
P <sub>1</sub>	0	6	3
P <sub>2</sub>	1	4	1
P <sub>3</sub>	2	7	4
P <sub>4</sub>	3	2	2



Process	AT	BT	Priority	CT	WT	RT	TAT.
P <sub>1</sub>	0	6	3	13	7	7	<del>13</del> 13
P <sub>2</sub>	1	4	1	5	0	0	<del>4</del> 4
P <sub>3</sub>	2	7	4	20	11	11	18
P <sub>4</sub>	3	2	2	7	2	2	4

chart	$P_2$	$P_4$	$P_1$	$P_3$	
	1	5	7	13	20

$$\text{Average waiting time} = \frac{0 + 2 + 7 + 11}{4}$$

$$= 5.$$



4) Consider the following process with arrival times & burst time & the time quantum for Round Robin scheduling is 2 units.

Process	AT	BT
P <sub>1</sub>	0	4
P <sub>2</sub>	1	5
P <sub>3</sub>	2	2
P <sub>4</sub>	3	3

→ Calculate the average turnaround time using Round Robin Scheduling

P	AT	BT	CT	RT	WT	TAT
P <sub>1</sub>	0	4	8	0	4	8
P <sub>2</sub>	1	5	14	2	8	13
P <sub>3</sub>	2	2	6	4	2	4
P <sub>4</sub>	3	3	12	6	6	9

Chart	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>1</sub>	P <sub>2</sub>	P <sub>4</sub>	P <sub>2</sub>	
	0	2	4	6	8	10	12	13	14

$$\text{Average Turnaround time} = \frac{8 + 13 + 4 + 9}{4} = 8.5$$

5. Consider a program that uses the `fork()` system call to create a child process. Initially, the parent process has a variable `x` with a value of 5. After forking, both the parent and child processes increment the value of `x` by 1. What will be the final values of `x` in the parent and child processes after the `fork()` call?

→ Step ① Before `fork()` is called,  
`int x = 5;`

② calling `fork()`.

- `fork()` call create a new child
- Both parent & child have separated
- memory space & contain  $x = 5$ .

③ After `fork()` execution:-

`x = x + 1;`

④ final value of `x`

parent	Value $x = 6$
child	Value $x = 6$

Even through both process increment `x`!

It is done in their own independent memory space.

So the final value remain `6` in both process

