Concepts of Operating System

Assignment 2

Mrunal Rajkumar Shedge_JH

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

- **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

- **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 without confirmation.**
   True

Identify the Incorrect Commands:

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

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

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

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

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

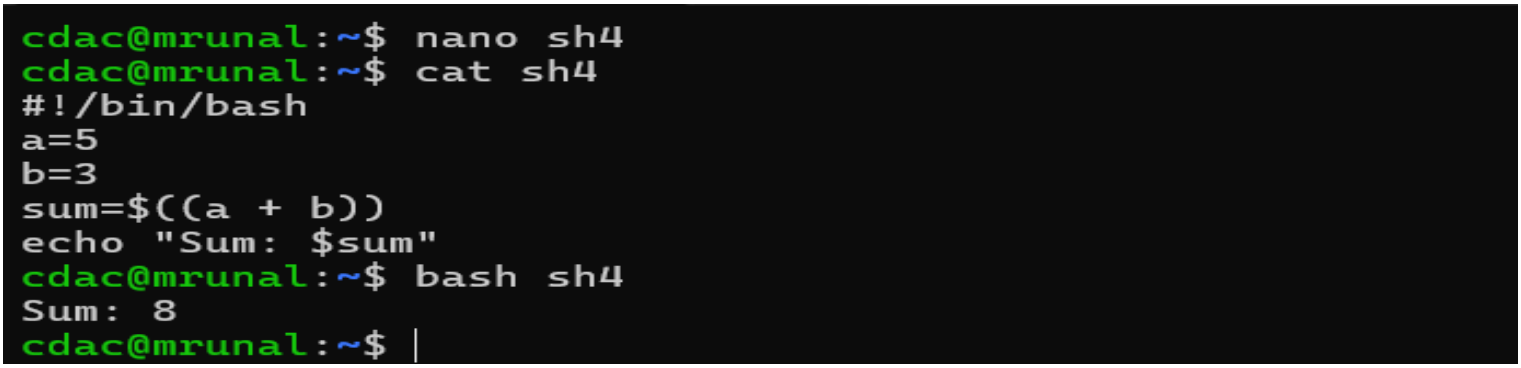
Part C

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

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

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

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



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

```
cdac@mrunal:~$ nano sh5
cdac@mrunal:~$ cat sh5
read -p "Enter a number: " num
if ((num % 2 == 0)); then
echo "Even"
else
echo "Odd"
fi
cdac@mrunal:~$ bash sh5
Enter a number: 45
Odd
cdac@mrunal:~$
```

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

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

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

```
cdac@mrunal:~$ nano 7
cdac@mrunal:~$ cat 7
i=1
while [ $i -le 5 ]; do
    echo "$i"
    ((i++))
done
cdac@mrunal:~$ bash 7
1
2
3
4
5
cdac@mrunal:~$
```

Question 8: Write a shell script that checks if a file named "file. Txt" exists in the current directory

```
cdac@mrunal:~$ nano sh8
cdac@mrunal:~$ cat sh8
if [ -f "file.txt" ]; then
echo "File exists"
else
echo "File does not exist"
fi
cdac@mrunal:~$ bash sh8
File does not exist
cdac@mrunal:~$ touch file.txt
cdac@mrunal:~$ bash sh8
File exists
cdac@mrunal:~$
```

Question 9: Write a shell script that checks if a number is greater than 10 and prints a message accordingly

```
cdac@mrunal:~$ nano sh9
cdac@mrunal:~$ cat sh9
read -p "Enter a number: " num
if [ $num -gt 10 ]; then
echo "Number is greater than 10"
else
echo "Number is 10 or less"
fi
cdac@mrunal:~$ bash sh9
Enter a number: 6
Number is 10 or less
cdac@mrunal:~$ bash sh9
Enter a number: 63
Number is greater than 10
cdac@mrunal:~$
```

Question 10: Write a shell script that prints a multiplication table for numbers from 1 to 5

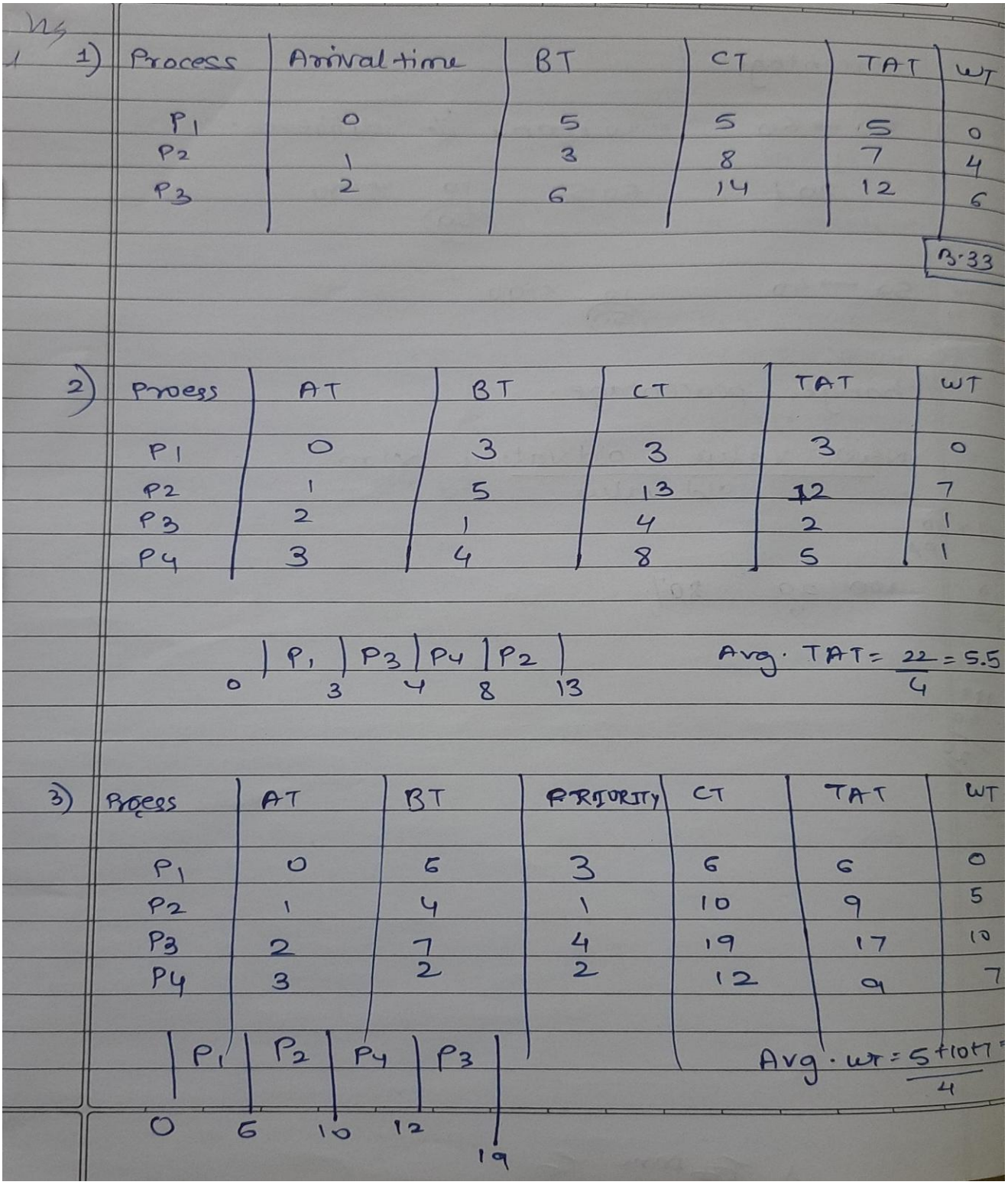
```
cdac@mrunal:~$ nano 10
cdac@mrunal:~$ cat 10
for i in {1..5}; do
for j in {1..5}; do
printf "%4d" $((i * j))
done
echo
done

cdac@mrunal:~$ bash 10
   1   2   3   4   5
   2   4   6   8  10
   3   6   9  12  15
   4   8  12  16  20
   5  10  15  20  25
cdac@mrunal:~$
```

Part E

Scheduling and Process Management Questions.

1. First-Come, First-Served (FCFS) Scheduling



1)

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

3.33

2)

Process	AT	BT	CT	TAT	WT
P1	0	3	3	3	0
P2	1	5	13	12	7
P3	2	1	4	2	1
P4	3	4	8	5	1

0	P_1	3	P_3	4	P_4	8	P_2	13

Avg. TAT = $\frac{22}{4} = 5.5$

3)

Process	AT	BT	PRIORITY	CT	TAT	WT
P_1	0	6	3	6	6	0
P_2	1	4	1	10	9	5
P_3	2	7	4	19	17	10
P_4	3	2	2	12	9	7

0	P_1	6	P_2	10	P_4	12	P_3	19

Avg. wt = $\frac{5+10+7}{4}$

4)

Process	AT	BT	CT	TAT	WT
P_1	0	4	8	8	4
P_2	1	5	14	13	8
P_3	2	2	6	4	2
P_4	3	3	13	10	7

Ready queue $P_1 \mid P_2 \mid P_3$

Gantt chart

P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_2

0 2 4 6 8 10 12 13 14

Avg. TAT $= \frac{34}{4}$

$= 8.75$

5) step 1 → Before fork() is called

int $x = 5$

step 2 → Calling fork

→ fork() system call create a new child.

→ Both parent & child have seperate memory space & contain $x = 5$

step 3 :- After fork () execut^{n}:

→ Both process will execute same step i.e. $x = x + 1$

step 4 :- final value of x

Parent : $x = 6$

Child : $x = 6$

Even tho both process increment x.

It is done in their own independent memory space.