CDAC MUMBAI

Concepts of Operating System

Assignment 2 Answers

Name: Amey Mahesh Sonawane

PG-DAC, CDAC Mumbai

Part A

- echo “Hello, World!”

Answer: This command will simply print the given string in the terminal.

- name = “Productive”

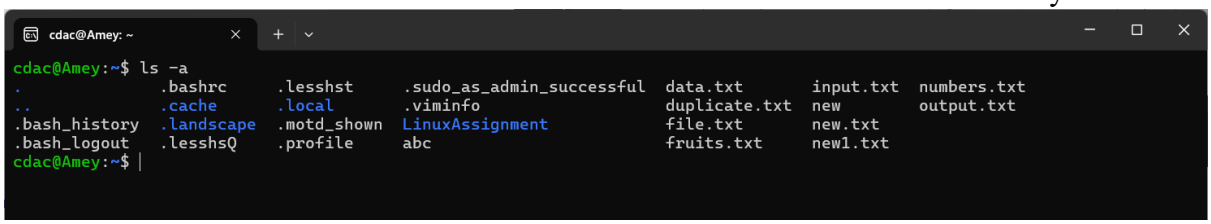
Answer: This command will assign the string “Productive” inside the variable “name”. No output will be displayed though.

- touch file.txt

Answer: This command will create a new file named “file.txt” or if it already exists, it will update the file’s last modified date and time. No output as such.

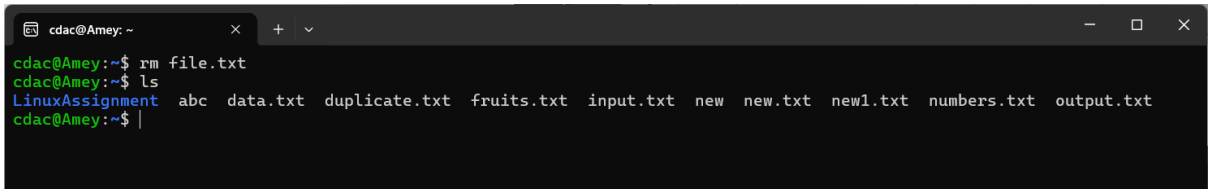
- ls -a

Answer: This command will list all the files and directories in the current directory.



- rm file.txt

Answer: This command will remove the file name file.txt from the current directory. No output as such.



- cp file1.txt file2.txt

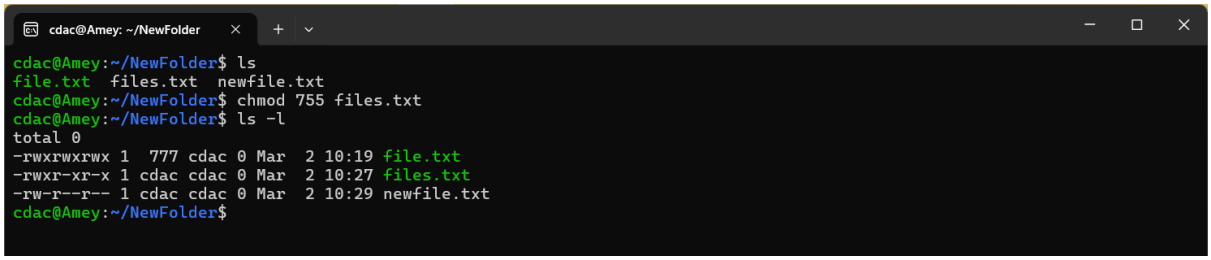
Answer: This command will copy the content of file1.txt to file2.txt. If file2.txt does not exist, it will create the same.

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

Answer: This command will move file.txt to specified directory.

- chmod 755 script.sh

Answer: This command changes the permission of the file script.sh. It sets the owner permission to Read, Write and Execute. The group permission to Read and Execute and the permission for others to Read and Execute as well.

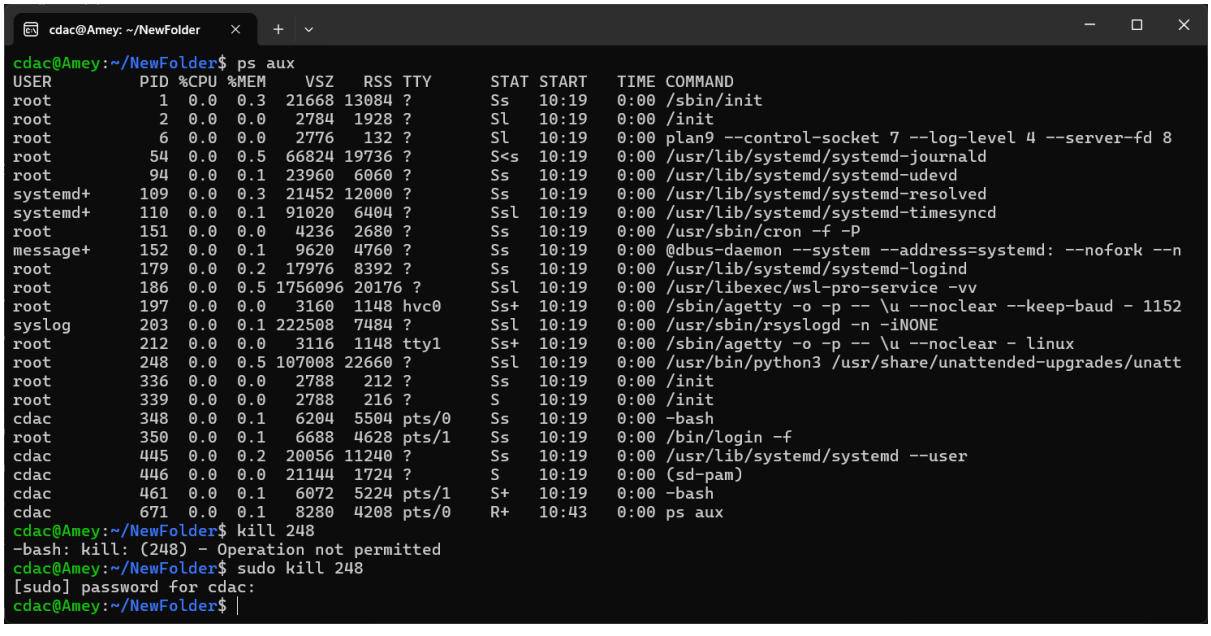


- grep "pattern" file.txt

Answer: This command searches for the word “pattern” inside the specified file.txt.

- kill PID

Answer: This command kills a specified running process, when we provide the process id for it in place of “PID”.



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

Answer: This command is a combination of various commands. Firstly

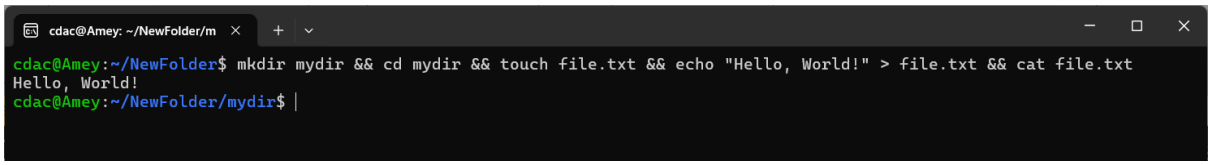
mkdir mydir: This will create a directory named “mydir” in current directory.

&& cd mydir: This will open the directory “mydir” in the terminal.

&& touch file.txt: This will create a file named file.txt inside the “mydir” directory.

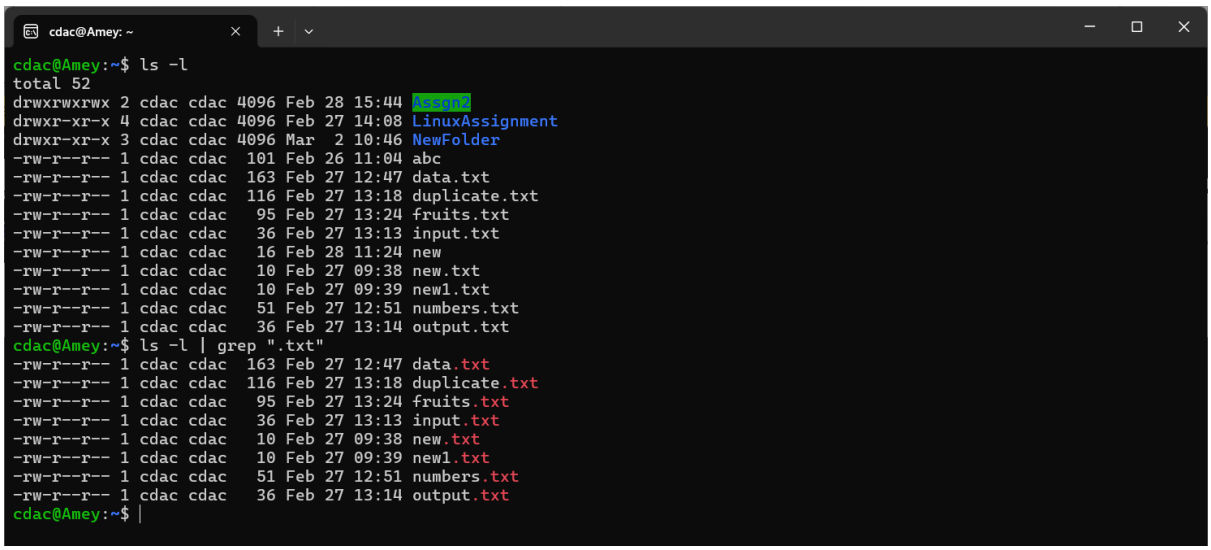
&& echo “Hello, World!” > file.txt: This will redirect the output “Hello, World!” into file.txt, storing the text inside the same.

&& cat file.txt: This will finally display the content of the file.txt on the terminal.



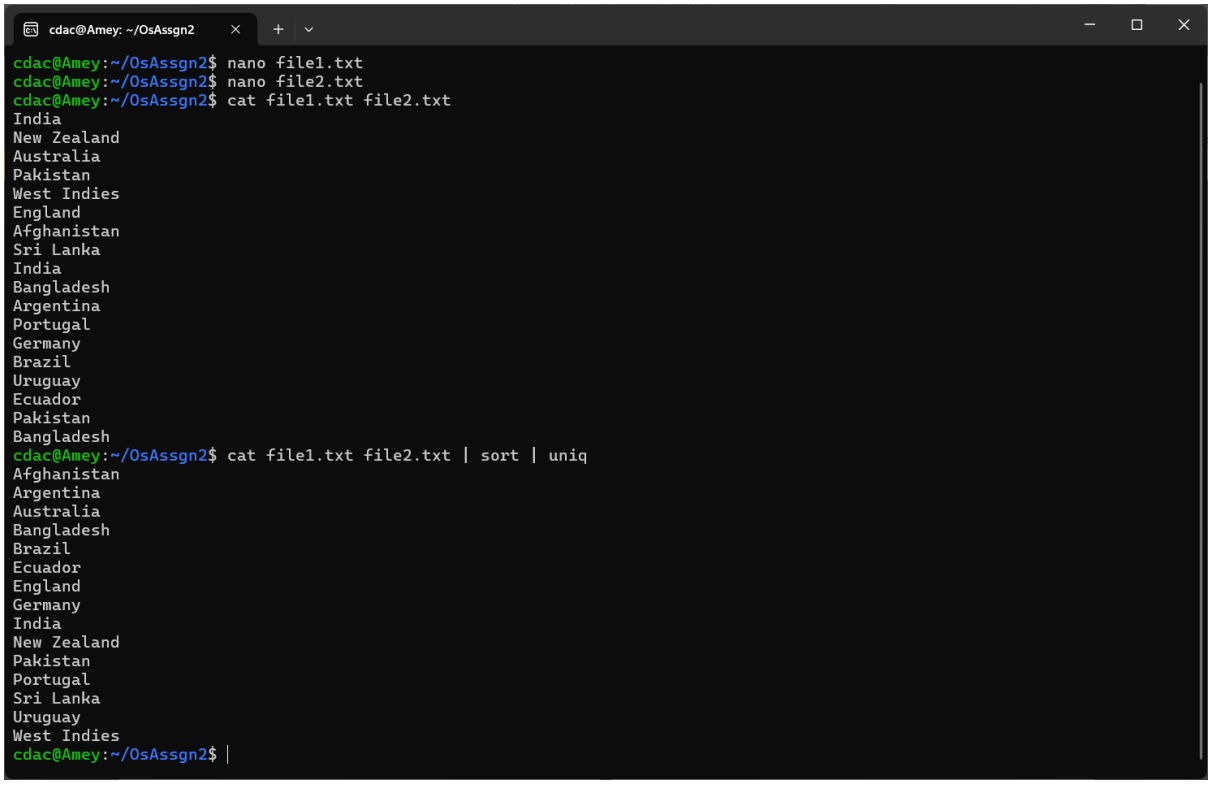
- ls -l | grep ".txt"

Answer: This command generates a detailed list of all the files in the current directory. And then this output is pipelined with grep “.text” command, which will sort the files in the list which have “.txt” in their name.



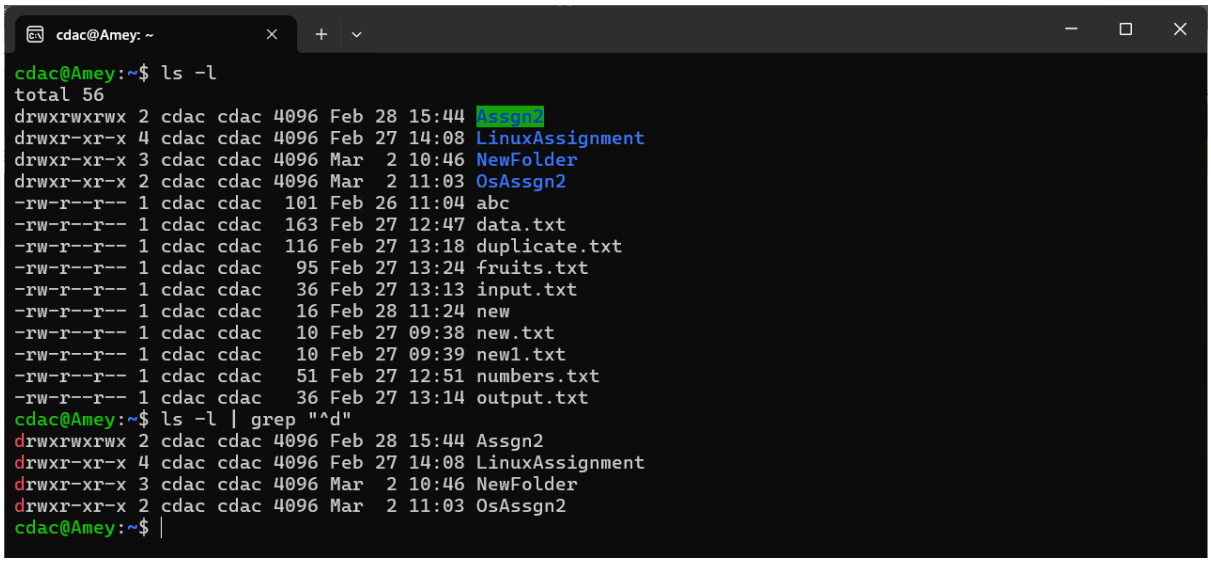
- cat file1.txt file2.txt | sort | uniq

Answer: This command will display the content of both the files file1.txt and file2.txt together, and this output is pipelined with the sort command which will alphabetically sort the content of the files. In the end uniq will display only those lines which are distinct, omitting all the repeated lines in the file.



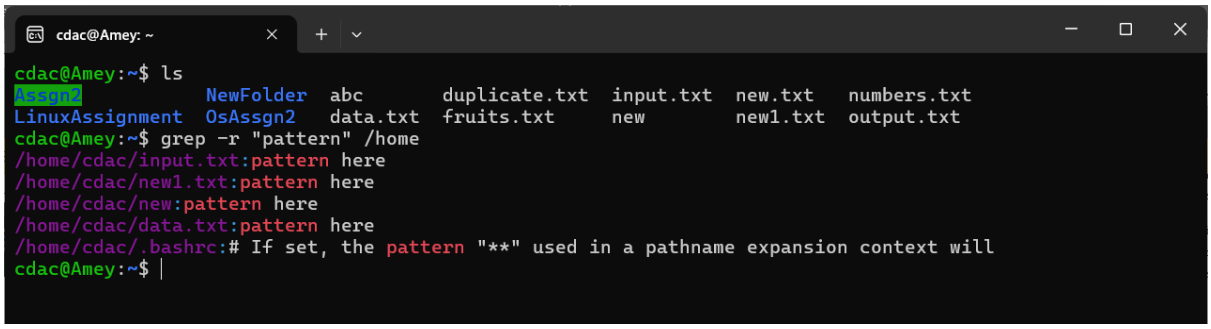
- ls -l | grep "^d"

Answer: This command will display a detailed list of all the directories in the current directory. Because the grep “^d” command filters out the files, as those lines won’t have d as the first character in them.



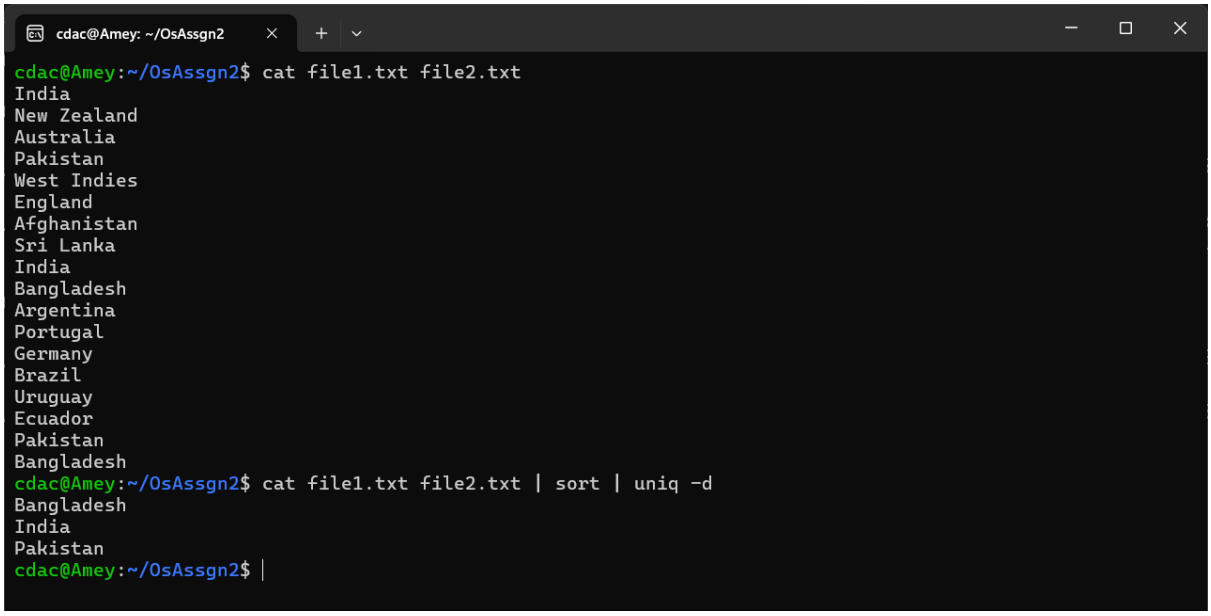
- grep -r "pattern" /path/to/directory/

Answer: This command will display the lines of files in a certain specified directory where the word “pattern” exists. Also, the search will be recursive, as in all the subdirectories and the files inside them will also be checked for the “pattern”.



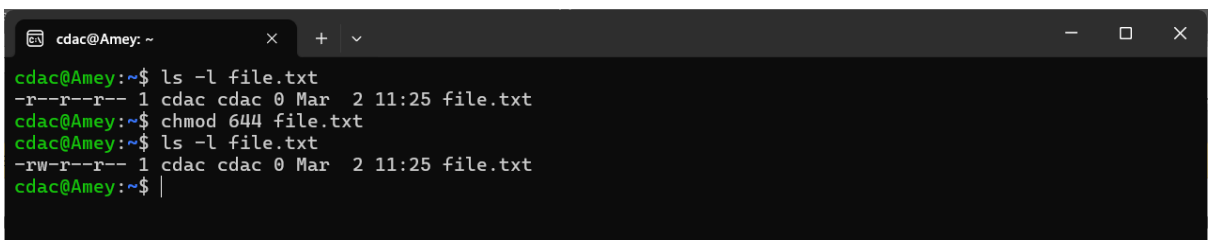
- cat file1.txt file2.txt | sort | uniq –d

Answer: This command will display all the duplicate lines inside the files file1.txt and file2.txt in the current folder.



- chmod 644 file.txt

Answer: This command will set the permissions for groups and others to read only, and read and write only for owner, for file.txt

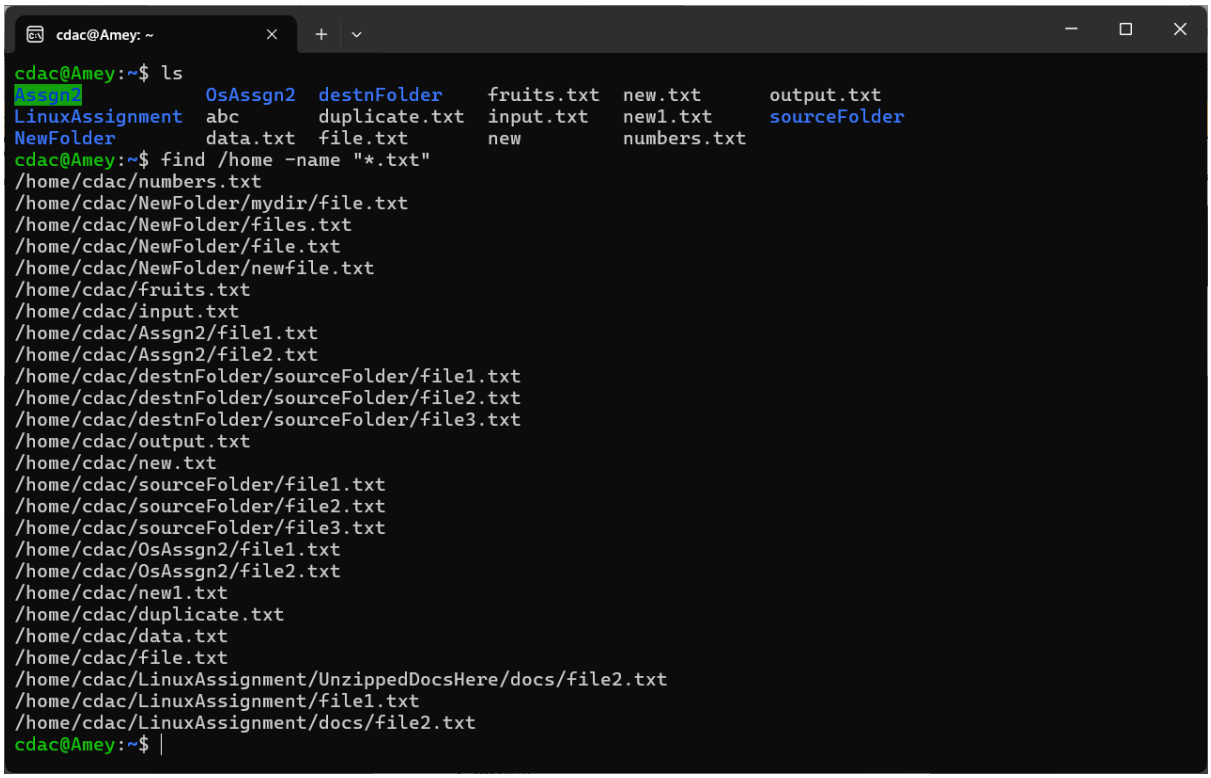


- cp -r source_directory destination_directory

Answer: This command will copy all he files and folders of the source directory to the destination directory.

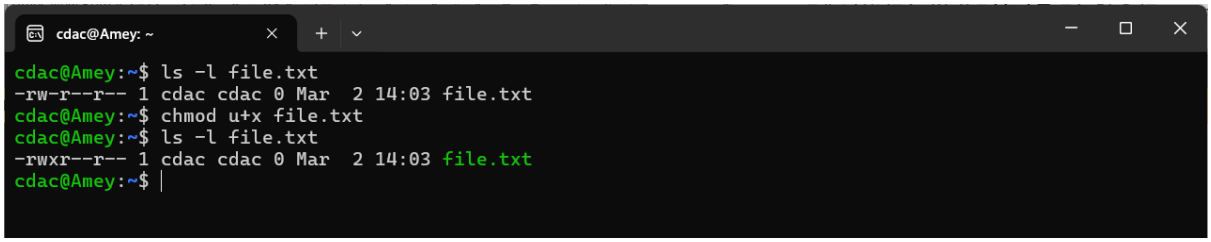
- find /path/to/search -name "*.txt"

Answer: This command will search in all the files and directories in the specified path for files with .txt in their name.



- chmod u+x file.txt

Answer: This command allows the user (owner) with the permission to execute the file named “file.txt”.



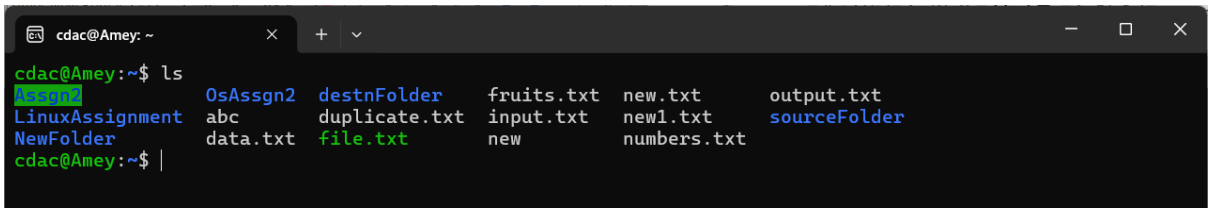
- echo $PATH

Answer: This command displays the system's environment variable PATH, which defines directories where the shell looks for executable commands.

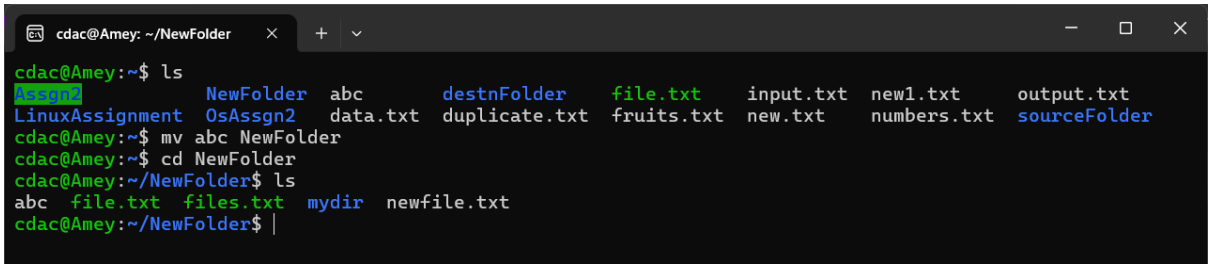
Part B

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



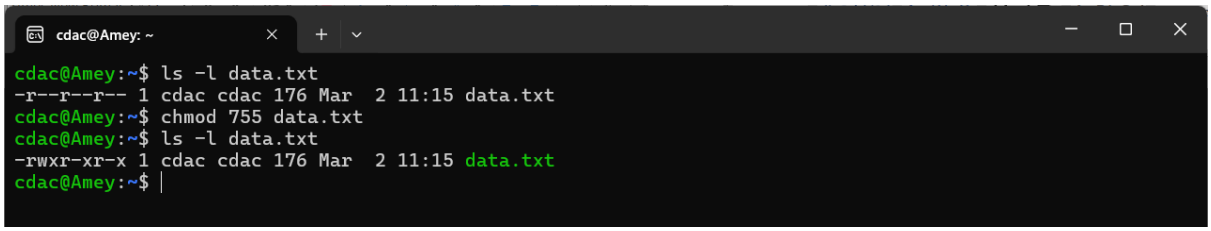
3. cd is used to copy files and directories.
   - False.

Reason: The command “cp” is used to copy files and directories instead of “cd”. “cd” is used to change the current directory in the terminal.

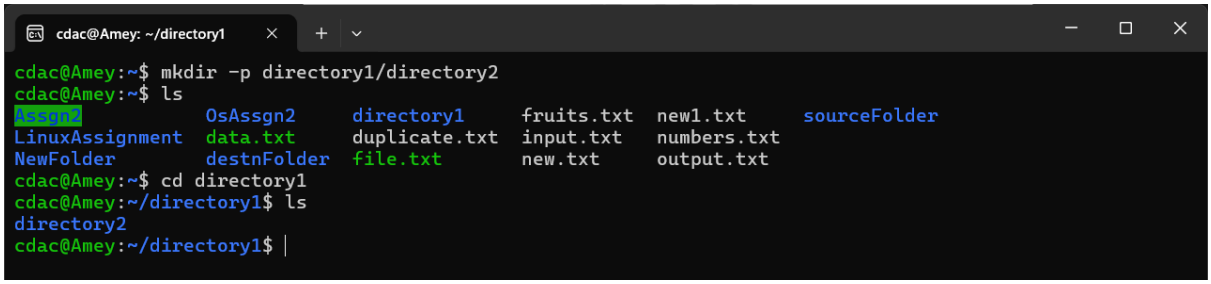
4. pwd stands for "print working directory" and displays the current directory.
   - True.

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

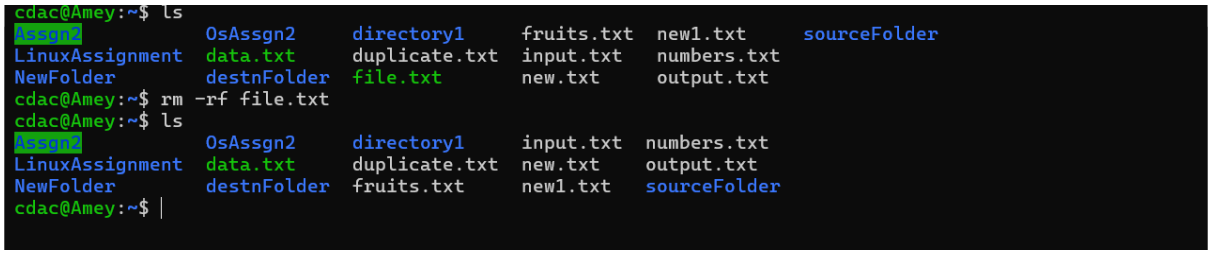
6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
   - True.



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. chmodx is used to change file permissions.
   - Correct

     If the “x” mentioned at the end of the command chmod is considered as a command line option, then yes, it is correct. chmod 755 gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
2. cpy is used to copy files and directories.
   - Incorrect

     cpy command does not copy files and directories. The task can be performed with the command cp followed by file name.
3. mkfile is used to create a new file.
   - Incorrect

     mkfile command does not exist in linux. To create a new file, one can use “touch file.name” instead.
4. catx is used to concatenate files.
   - Correct

     If the “x” mentioned at the end of the command is considered as a command line option then yes, it is correct. cat command is used to concatenate and display content of a file. It can be followed by -n to show numbered output lines, or by -b to show only non-empty lines.
5. rn is used to rename files.
   - Incorrect

     To rename a file, the mv command is used. There is no rn command in existence.

Part C

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

Answer: The command is as follows: -

echo “Hello, World!”

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

Answer: The command is as follows: -

name="CDAC Mumbai"

echo "The name is: $name"

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

Answer: The command is as follows: -

read -p "Enter a number: " num

echo "You entered: $num"

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

Answer: The command will be as follows: -

num1=5

num2=3

sum=$((num1 + num2))

echo "The sum is: $sum"

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

Answer: The command will be as follows: -

read -p "Enter a number: " num

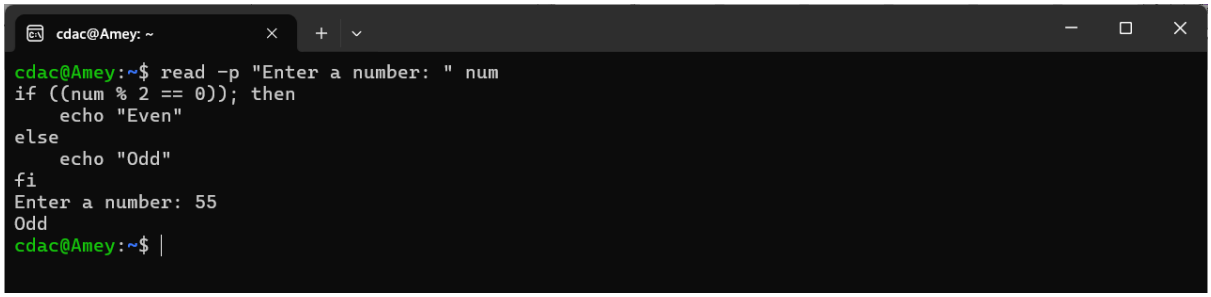
if ((num % 2 == 0)); then

    echo "Even"

else

    echo "Odd"

fi



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

Answer: The command is as follows: -

```
for i in {1..5}; do
    echo "$i"
done
```

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

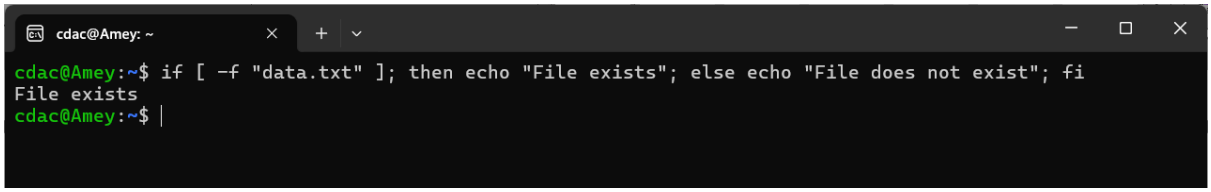
Answer: The command is as follows: -

```
i=1
while [ $i -le 5 ]; do
    echo "$i"
    ((i++))
done
```

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".

Answer: The command is as follows: -

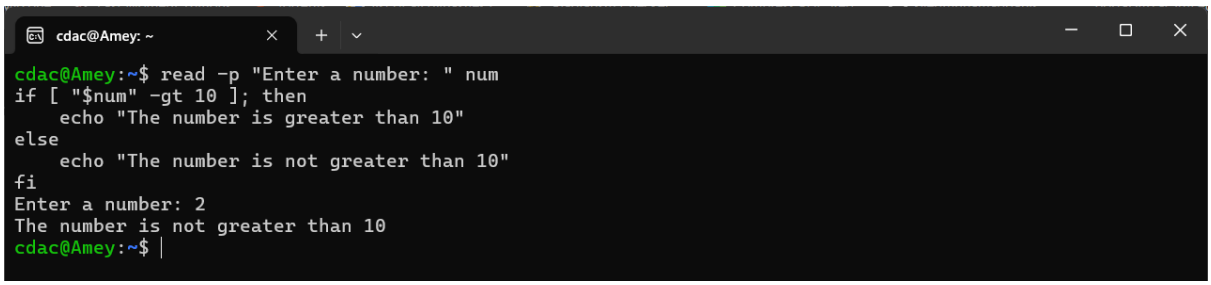
```
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
```



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.

Answer: The command is as follows: -

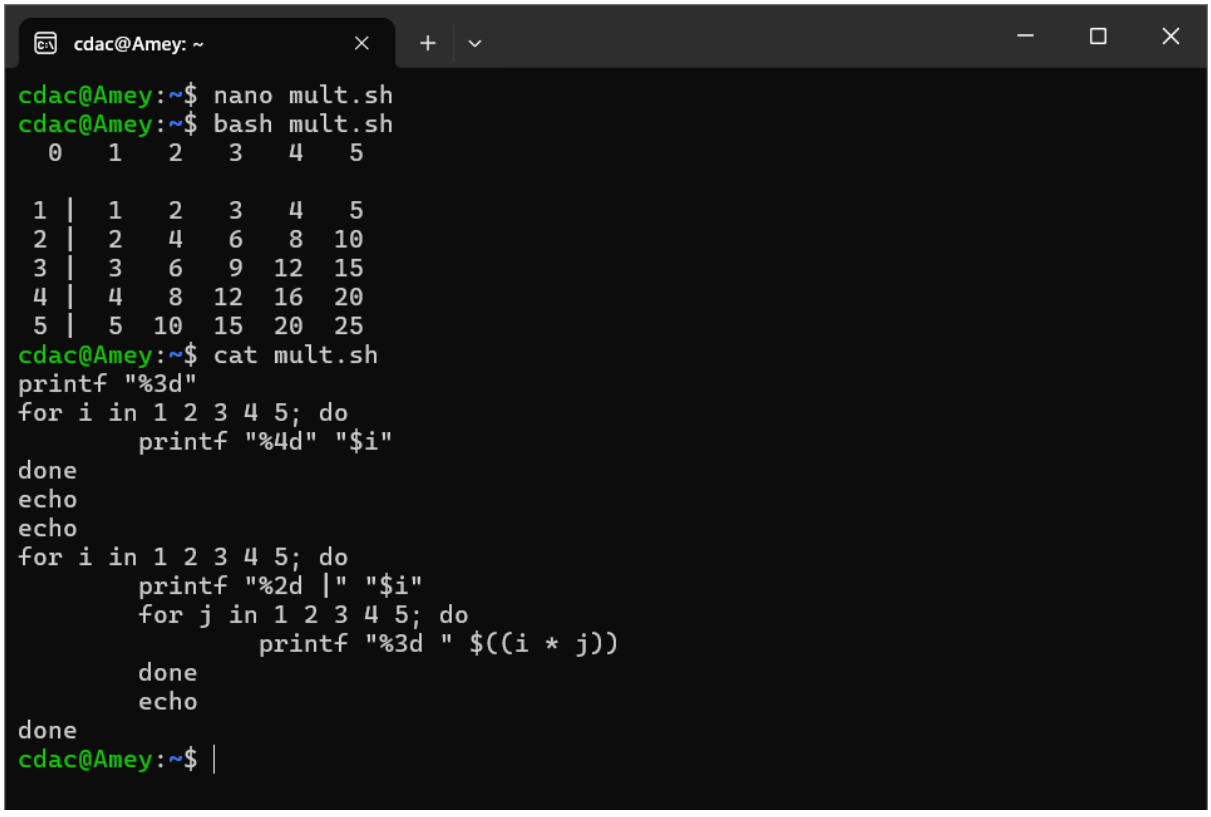
```
read -p "Enter a number: " num
if [ "$num" -gt 10 ]; then
    echo "The number is greater than 10"
else
    echo "The number is not greater than 10"
fi
```



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.

Answer: The shell script will be as follows: -

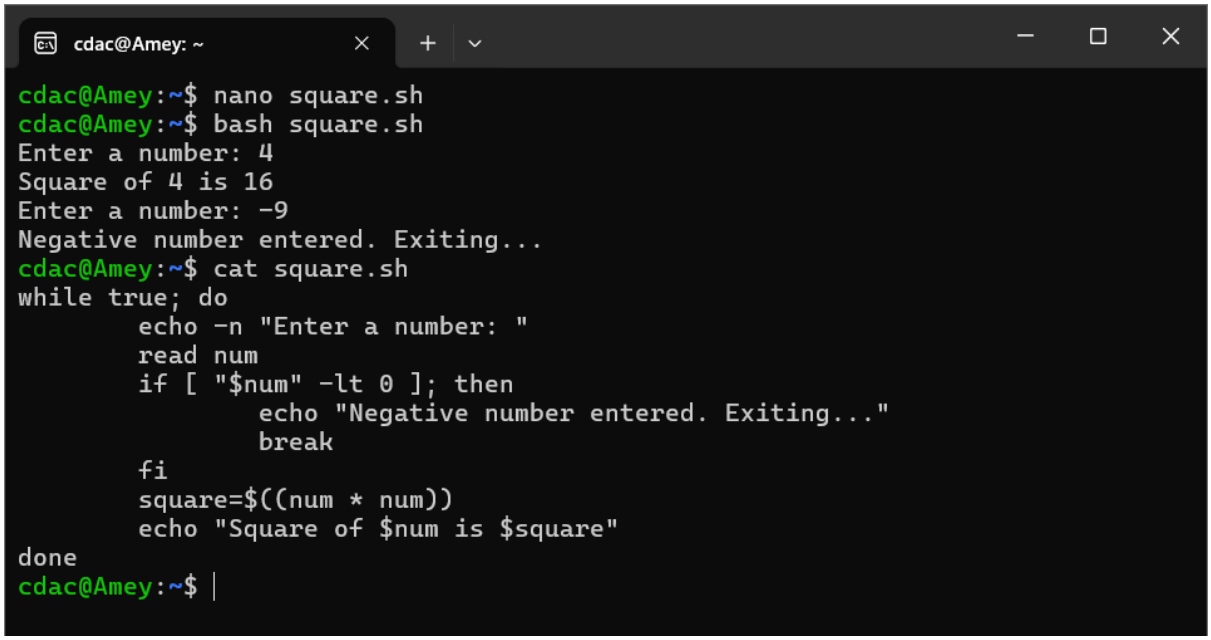
```
printf "%3d"
for i in 1 2 3 4 5; do
	printf "%4d" "$i"
done
echo
echo
for i in 1 2 3 4 5; do
	printf "%2d |" "$i"
	for j in 1 2 3 4 5; do
		printf "%3d " $((i * j))
	done
	echo
done
```



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.

Answer: The shell script will be as follows: -

```
while true; do
    echo -n "Enter a number: "
    read num
    if [ "$num" -lt 0 ]; then
        echo "Negative number entered. Exiting..."
        break
    fi
    square=$((num * num))
    echo "Square of $num is $square"
done
```



Part E

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

Process ID	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.

Solution:

Gantt Chart:

P1	P2	P3	
0	5	8	14

Response time is time taken by a process to respond to the CPU call.

So, we can infer the response times for the processes, from the gantt chart as these: -

Process ID	Arrival Time	Burst Time	Response Time
P1	0	5	0
P2	1	3	5
P3	2	6	8

Now waiting time can be calculated with the formula,

Waiting time = Response time – Arrival time (for non-pre-emptive

According the following table can be formed

Process ID	Arrival Time	Burst Time	Response Time	Waiting Time
P1	0	5	0	0 – 0 = 0
P2	1	3	5	5 – 1 = 4
P3	2	6	8	8 – 2 = 6

So, the average waiting time for these processes using FCFS algorithms is equal to

= (0 + 4 + 6) / 3

= 3.33 units.

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

Process ID	**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.

Solution:

Gantt Chart:

P1	P1	P3	P1	P4	P2	
0	1	2	3	4	8	13

Waiting time is the time spent by the process in the waiting queue,

For process P1, waiting time = 1

For process P2 = 8 – 1 = 7

For process P3 = 0

For process P4 = 4 – 3 = 1

therefore, the table can be formed like this

Process ID	**Arrival Time**	**Burst Time**	**Waiting Time**
P1	0	3	1
P2	1	5	7
P3	2	1	0
P4	3	4	1

And with waiting time available for each process, turn-around time can be calculated with the formula

Turn-Around Time = Waiting Time + Burst Time

Hence the table will look like this with added turn-around times of processes

Process ID	**Arrival Time**	**Burst Time**	**Waiting Time**	**Turn-Around Time**
P1	0	3	1	4
P2	1	5	7	12
P3	2	1	0	1
P4	3	4	1	5

So, the average turn-around time = (4 + 12 + 1 + 5) / 4

= 5.5 units

The average turn-around time for these processes is equal to 5.5 units.

3. 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.

Solution:

By using priority scheduling,

Gantt Chart:

P1	P2	P2	P2	P2	P4	P1	P3	
0	1	2	3	4	5	7	13	20

Waiting time is time spent by the process in the queue before it gets executed completely.

So, waiting times for each of the processes can be inferred as,

For process P1, waiting time = 0 + 7 = 7

For P2 = 1 – 1 = 0

For P3 = 13 – 2 = 11

And for P4 = 5 – 3 = 2

Accordingly, average waiting time can be calculated as

= (P1 + P2 + P3 + P4) / 4

= 20 / 4

= 5

Average waiting time of these processes is 5 units.

4. Consider the following processes with arrival times and burst times, and the time quantum for

Round Robin scheduling is 2 units:

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

Calculate the average turnaround time using Round Robin scheduling.

Solution:

By round robin scheduling algorithm,

Gantt Chart:

P1	P2	P3	P4	P1	P2	P4	P2	
0	2	4	6	8	10	12	13	14

Waiting time is equal to the total time a process spends waiting for completion of execution.

So, from the Gantt chart, waiting times can be inferred as: -
P1 = 0 + 6 = 6

P2 = (2 + 6 +1) – 1 = 8

P3 = 4 – 2 = 2

And P4 = (6 + 4) – 3 = 7

Using these values, we can calculate the turn-around time of each of the processes as well

The formula is,

Turn-Around Time = Waiting Time + Burst Time

Turn-Around Time for P1 = 6 + 4 = 10

For P2 = 5 + 8 = 13

For P3 = 2 + 2 = 4

And for P4 = 3 + 7 = 10

In conclusion, the average turn-around time would be equal to

= (P1 + P2 + P3 + P4) / 4

= 37 / 4

= 9.25 units

Thus, the average turn-around time for these processes is 9.25 units.

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?

Answer:

The fork() is a system call in operating systems to generate a child process from an original process or parent process. Such different instances of a process allow much more efficient process management environment. With child processes in existence, there can be parallel execution of tasks. This increases the speed of processing considerably.

What exactly happens when a duplicate copy of parent process is created, is that the child process gets a duplicate copy of the parent process's memory space, including any initialized variables. After the fork() is called, both child and parent process run independently, which means changing anything in parent process won't affect anything in child process.

Given situation is,

Variable x = 5 in parent process

fork() is called, so a child process is generated.

And finally, both parent and child increment x by 1

So, in parent process, x = 5 will become x = 5 + 1

That is x becomes x = 6

Child process remains unaffected and simply increments x by 1 too

So x = 5 + 1 = 6

So we can conclude that since parent and child process are independent, the final values of x in parent process will be x = 6 and in the child process will also be x = 6.