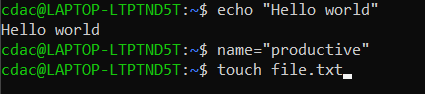
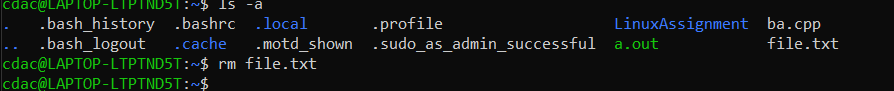
Assignment 2

Part A

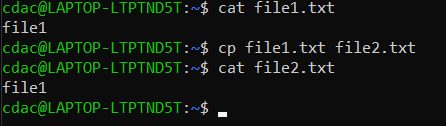
* Prints “Hello World” on the console
* name is the variable it is assigned the value Productive
* touch file.txt creates a new file if it doesn’t exist
* ls -a displays all the files n directories of the pwd



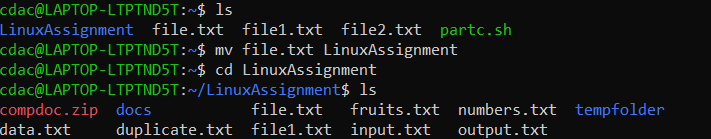
* rm file.txt will delete the file file.txt
* cp file1.txt file2.txt will return the msg

“no such file or directory” since it was deleted

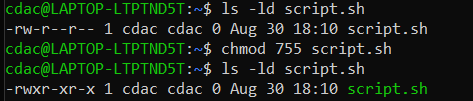
If file1.txt exists it will create a file a file2.txt and copy all contents

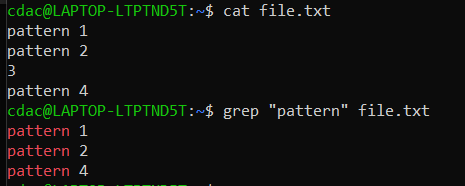
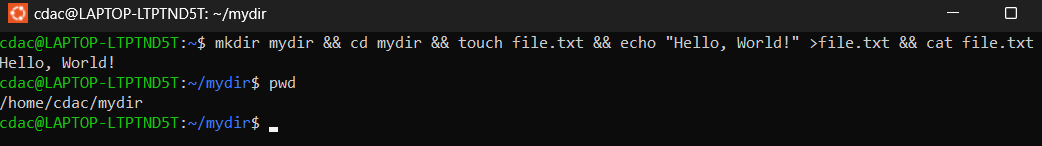
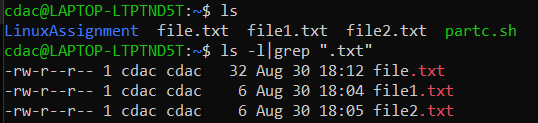
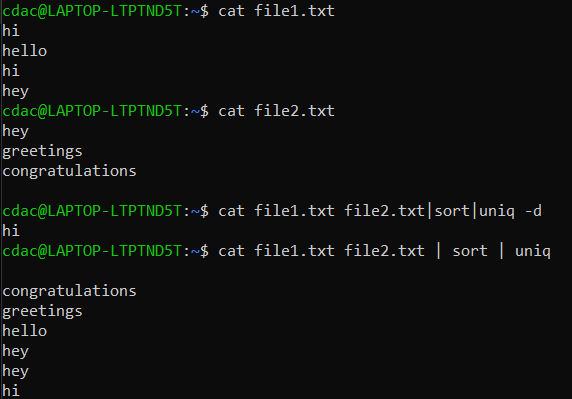
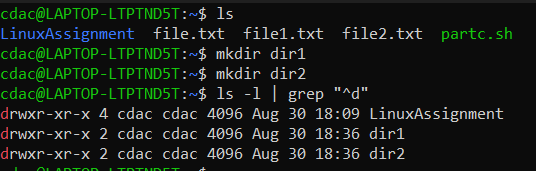
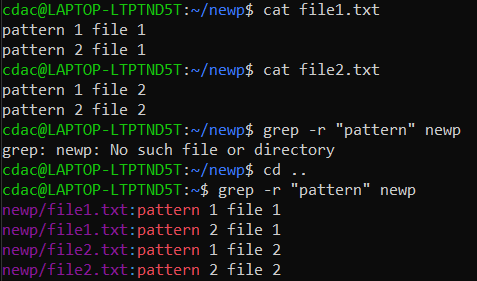
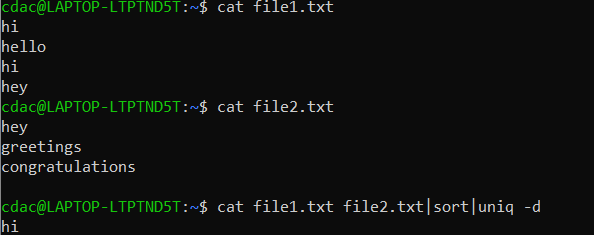
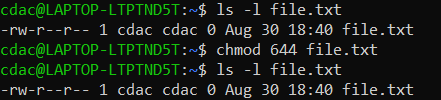
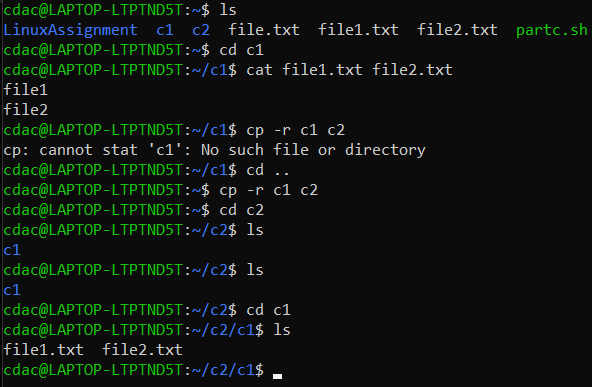
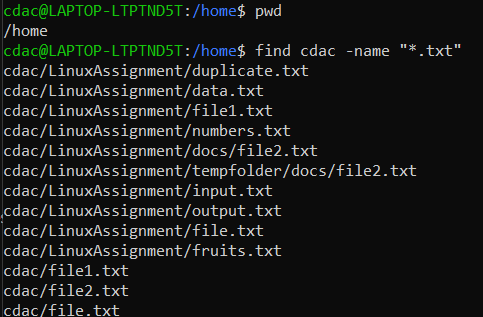
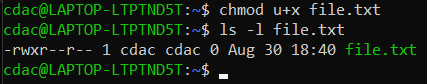
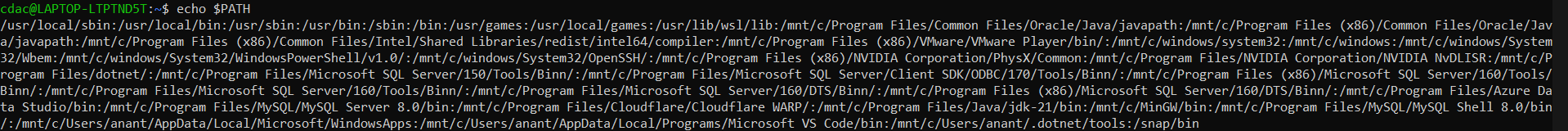


* mv file.txt /path/to/directory: moves the file.txt to the specified directory



* chmod 755 script.sh- all users get read/execute permissions, owner gets write permission also



* grep "pattern" file.txt- returns all the lines where the word “pattern” occurs
* kills a process, requires the port no. and pid to kill the process
* mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt 🡪 makes a directory mydir, makes mydir the pwd, creates file.txt, “hello world” is inputted into file.txt, file.txt contents are displayed on console
* ls -l | grep ".txt"🡪 will list all files that are .txt along with permissions
* cat file1.txt file2.txt | sort | uniq 🡪 combines the contents of file1.txt and file2.txt, sorts them in alphabetical order and then displays all the unique lines
* ls -l | grep "^d" will display all child directories in the parent directory
* grep -r "pattern" /path/to/directory/ will search all lines with the word pattern in all files in the directory then display the lines on console
* cat file1.txt file2.txt | sort | uniq –d🡪combines content from both files, sorts in alphabetical order than displays all duplicate lines
* chmod 644 file.txt🡪all users have read permission for file.txt, only owner has read permission, none have execute permission
* cp -r source\_directory destination\_directory copies entire source directory into destination directory
* find /path/to/search -name "\*.txt"🡪finds all .txt files in the given directory
* chmod u+x file.txt 🡪 owner gets execute permissions for file.txt
* echo $PATH-🡪specifies all files which can be executed with mentioning type of file

Part B

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. It is used to change the present working directory
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 incorrent commands

1. chmodx is used to change file permissions. 🡪chmod
2. cpy is used to copy files and directories. 🡪cp
3. mkfile is used to create a new file. 🡪touch or nano
4. catx is used to concatenate files. 🡪cat
5. rn is used to rename files. 🡪mv

Part C

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

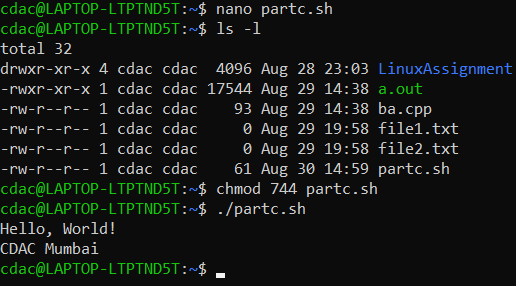
**echo Hello, World!**

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

value of the variable.

**name=”CDAC Mumbai”**

**echo $name**



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

**echo Enter a number**

**read n**

**echo $n**

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

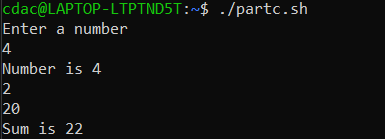
result.

read n1

read n2

sum=$(($n1+$n2))

echo Sum is $sum



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

prints "Odd".

echo “enter a number”

read n

res=$(($n%2))

if [ $res -eq 0 ]

then

echo “Number $n is even”

else

echo “number $n is odd”

fi

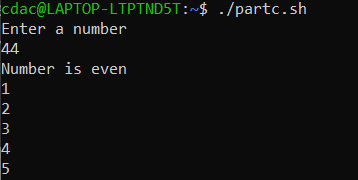
1. Write a shell script that uses a for loop to print numbers from 1 to 5.

for a in {1..5};

do

echo $a

done



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

n=1

while [ n -le 5 ];

do

echo $n

n++

done

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".

if [ -e file.txt ]

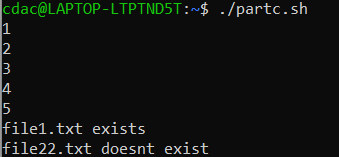
then

echo file exits

else

echo file doesn’t exist

fi



9 Write a shell script that uses the if statement to check if a number is greater than 10 and

prints a message accordingly.

echo “enter a number”

read n

if [ $n -gt 10 ]

then

echo $n “is greater than 10”

elif [ $n -eq 10 ]

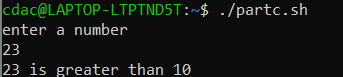
then

echo $n “is equal to 10”

else

echo $n “is less than 10”

fi



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.

for i in {1..5}

do

echo "Multiplication table for $i:"

for j in {1..5}

do

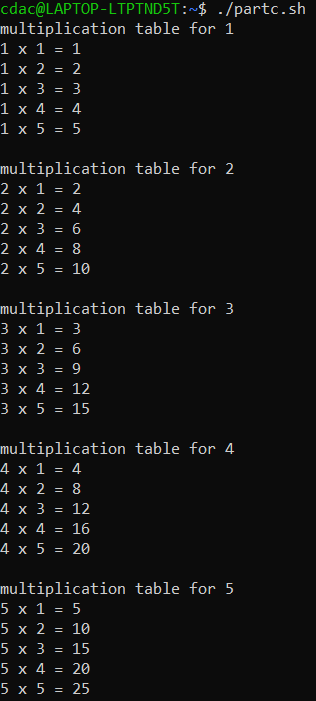
res=$(($i \* $j))

echo "$i x $j = $res"

done

echo ""

done



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.

while [ true ];

do

echo “enter a number”

read n

if [ $n -ge 0 ];

then

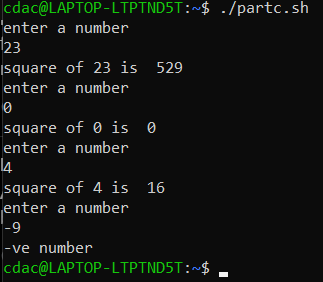
echo “square of number is” $(($n \*\* 2))

else

echo “negative number”

break

fi

done 

Part E

1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival | Burst | Completion | TAT | Waiting |
| 1 | 0 | 5 | 5 | 5 | 0 |
| 2 | 1 | 3 | 8 | 7 | 4 |
| 3 | 2 | 6 | 14 | 12 | 6 |

Avg waiting time= 10/3=3.33

2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival | Burst | Completion | TAT | Waiting |
| 1 | 0 | 3 | 3 | 3 | 0 |
| 2 | 1 | 5 | 13 | 12 | 7 |
| 3 | 2 | 1 | 4 | 2 | 1 |
| 4 | 3 | 4 | 8 | 5 | 1 |

Avg turnaround time= 22/4=5.5

3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process | Priority | Arrival | Burst | Completion | TAT | WT |
| 1 | 3 | 0 | 6 | 6 | 6 | 0 |
| 2 | 1 | 1 | 4 | 10 | 9 | 5 |
| 3 | 4 | 2 | 7 | 19 | 17 | 10 |
| 4 | 2 | 3 | 2 | 17 | 9 | 7 |

Avg TAT=10.25

4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process | Arrival | Burst | Completion | TAT |
| 1 | 0 | 4 | 8 | 8 |
| 2 | 1 | 5 | 14 | 13 |
| 3 | 2 | 2 | 6 | 4 |
| 4 | 3 | 3 | 13 | 10 |

Avg TAT=9.75

5. after forking, values of parent and child process will be 6