

CDAC MUMBAI Concepts of Operating System

Assignment 2

Part A

echo "Hello, World!"

- Prints "Hello, World!" to the terminal.

name="Productive"

- Assigns the string "Productive" to the variable name in the current shell session.

touch file.txt

- Creates an empty file named file.txt if it doesn't exist or updates its timestamp if it does.

ls -a

- Lists all files and directories, including hidden ones (those that start with .).

rm file.txt

- Deletes file.txt. (Irreversible unless using a recovery tool.)

cp file1.txt file2.txt

- Copies file1.txt to file2.txt.

mv file.txt /path/to/directory/

- Moves file.txt to the specified directory.

chmod 755 script.sh

- Grants read, write, and execute (rwx) permissions to the owner, and read and execute (r-x) to group and others for script.sh.

grep "pattern" file.txt

- Searches for lines in file.txt that contain "pattern" and prints them.

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 named mydir, navigates into it, creates an empty file file.txt, writes "Hello, World!" into file.txt, then displays its content.

ls -l | grep ".txt"

- Lists files in long format (ls -l) and filters the output to show only files containing .txt.

cat file1.txt file2.txt | sort | uniq

- Concatenates file1.txt and file2.txt, sorts the combined lines, and removes duplicate lines.

ls -l | grep "^d"

- Lists files in long format and filters only directories (lines starting with d).

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

- Recursively searches for "pattern" in all files inside /path/to/directory/.

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

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

chmod 644 file.txt

- Sets file.txt permissions so the owner can read and write (rw-), while group and others can only read (r--).

cp -r source_directory destination_directory

- Recursively copies source_directory and its contents to destination_directory.

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

- Searches for all .txt files under /path/to/search.

chmod u+x file.txt

- Grants execute (x) permission to the file owner (u).

echo \$PATH

- Displays the directories in the system's PATH environment variable, which determines where executable files are searched for.

Part B

Identify True or False:

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

Ans: True – ls is used to list files and directories in a directory.

2. mv is used to move files and directories.

Ans: True – mv is used to move (or rename) files and directories.

3. cd is used to copy files and directories.

Ans: False – cd is used to change directories, not copy files. (cp is used for copying files and directories.)

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

Ans: True – pwd stands for "print working directory" and displays the current directory path.

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

Ans: True – grep searches for patterns in files and prints matching lines.

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

Ans: True – chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to the group and others.

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

Ans: True – mkdir -p directory1/directory2 creates directory1 (if it does not exist) and then directory2 inside it.

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

Ans: True – rm -rf file.txt deletes a file forcefully (-f) and recursively (-r for directories), without asking for confirmation.

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.

Ans: Incorrect: chmodx → The correct command is **chmod** (used to change file permissions).

2. cpy is used to copy files and directories.

Ans: Incorrect: cpy → The correct command is **cp** (used to copy files and directories).

3. mkfile is used to create a new file.

Ans: Incorrect: `mkfile` → The correct command is `touch filename` (creates an empty file).

4. `catx` is used to concatenate files.

Ans: Incorrect: `catx` → The correct command is `cat` (used to concatenate and display file contents).

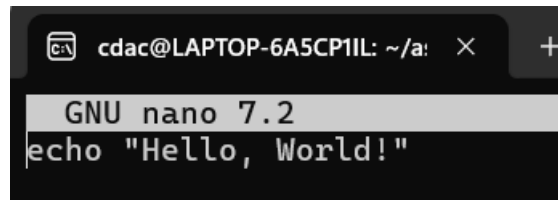
5. `rn` is used to rename files.

Ans: Incorrect: `rn` → The correct command is `mv oldname newname` (used to rename files and directories).

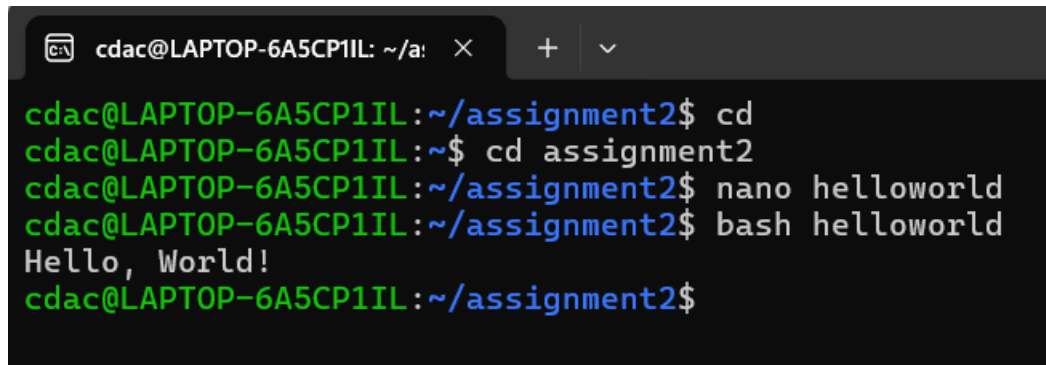
Part C

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

Ans:



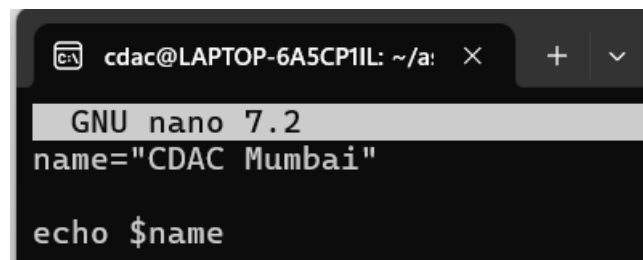
```
cdac@LAPTOP-6A5CP1IL: ~/a: × +
GNU nano 7.2
echo "Hello, World!"
```



```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
cdac@LAPTOP-6A5CP1IL:~/assignment2$ cd
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano helloworld
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash helloworld
Hello, World!
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

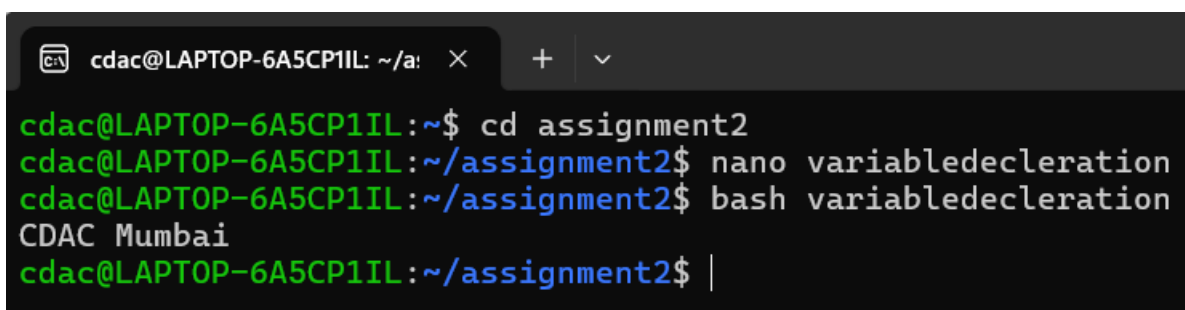
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

Ans:



```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
GNU nano 7.2
name="CDAC Mumbai"

echo $name
```



```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano variabledecleration
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash variabledecleration
CDAC Mumbai
cdac@LAPTOP-6A5CP1IL:~/assignment2$ |
```

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

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
GNU nano 7.2
echo Enter a input:
read n

echo Printing the input: $n
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano input
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash input
Enter a input:
12
Printing the input: 12
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

Question 4: Write a shell script that uses a for loop to print numbers from 1 to 5

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
GNU nano 7.2
for i in {1..5}
do
    echo $i
done
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + ∨
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano range1
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash range1
1
2
3
4
5
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

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

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
GNU nano 7.2
echo Enter a number:
read num

for i in $num
do
    if(( num%2 == 0 ))
    then
        echo Even
    else
        echo Odd
    fi
done
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano evenodd
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash evenodd
Enter a number:
12
Even
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash evenodd
Enter a number:
10
Even
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash evenodd
Enter a number:
5
Odd
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

Question 6: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
GNU nano 7.2
a=5
b=3
sum=$(( a+b ))

echo sum of first and second number is: $sum
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano addition
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash addition
sum of first and second number is: 8
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

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

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
GNU nano 7.2
num=1

while [ $num -le 5 ]
do
    echo $num
    ((num++))
done
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano range2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash range2
1
2
3
4
5
cdac@LAPTOP-6A5CP1IL:~/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".

Ans:

```
GNU nano 7.2
if [ -e file.txt ]
then
    echo File exists
else
    echo File does not exist
fi
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: X + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano file1.txt
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash file1.txt
File does not exist
cdac@LAPTOP-6A5CP1IL:~/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.

Ans:

```
GNU nano 7.2
echo Enter a number:
read number

if [ $number -gt 10 ]
then
    echo the number $number is greater that 10.
else
    echo the number $number is not greater than 10.
fi
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano greatest
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash greatest
Enter a number:
15
the number 15 is greater that 10.
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash greatest
Enter a number:
5
the number 5 is not less than 10.
cdac@LAPTOP-6A5CP1IL:~/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.

Ans:

```
GNU nano 7.2
echo "Multiplication Table (1 to 5)"
echo "-----"

for(( i=1; i<=5; i++ ))
do
    for(( j=1; j<=5; j++ ))
    do
        printf "%4d" $((i*j))
    done
    echo
done
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: × + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano multiplicationtable
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash multiplicationtable
Multiplication Table (1 to 5)
-----
 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@LAPTOP-6A5CP1IL:~/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.

Ans:

```
cdac@LAPTOP-6A5CP1IL: ~/a: X + v
GNU nano 7.2 square
while true
do
  echo "Enter a number(Enter a negative number to exit) :"
  read num

  if [ $num -lt 0 ]
  then
    echo "Negative number is entered. Exiting..."
    break
  fi
  square=$((num*num))
  echo "Square of $num is: $square"
done
```

```
cdac@LAPTOP-6A5CP1IL: ~/a: X + v
cdac@LAPTOP-6A5CP1IL:~$ cd assignment2
cdac@LAPTOP-6A5CP1IL:~/assignment2$ nano square
cdac@LAPTOP-6A5CP1IL:~/assignment2$ bash square
Enter a number(Enter a negative number to exit) :
10
Square of 10 is: 100
Enter a number(Enter a negative number to exit) :
-12
Negative number is entered. Exiting...
cdac@LAPTOP-6A5CP1IL:~/assignment2$
```

Part D

Assignment No.-2

Name → Nikita Kantilal Patil KH

Page No.

Date 28.02.25

Q1 FCFS

Process	Arrival	Burst	Waiting Time
P ₁	0	5	0
P ₂	1	3	4
P ₃	2	6	6

Gantt chart

P ₁	P ₂	P ₃	<u>14</u>
0	5	8	

$$\text{Avg. Waiting Time} = \frac{(0+4+6)}{3} = \frac{10}{3} = 3.33$$

Q2 SJF (Non-Preemptive)

Process	Arrival	Burst	Waiting	TAT
P ₁	0	3	0	3
P ₂	1	5	7	12
P ₃	2	1	1	2
P ₄	3	4	1	5

Gantt chart -

P ₁	P ₃	P ₄	<u>13</u>
0	3-4	8	P ₂

$$\text{Avg. TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$$

Q3 Priority Scheduling (Non-preemptive)

Process	Arrival	Burst	Priority	Waiting Time
P ₁	0	6	3	0
P ₂	1	4	1	5
P ₃	2	7	4	10
P ₄	3	2	2	7

Gantt chart (Non-Preemptive)

P₁ P₂ 10 12 9
0 6 P₄ P₃

$$\text{Avg. Waiting Time} = \frac{22}{4} = \underline{\underline{5.5}}$$

Gantt chart (preemptive)

P₁ P₂ P₄ 7 12 19
0 2 5 P₁ P₃

$$\text{Avg. Waiting time} = \frac{16}{4}$$

$$= \underline{\underline{4.5}}$$

Waiting time

6

6

2

10

Q4 Round Robin - quantum = 2 units

Process	Arrival	Burst	Waiting Time	TAT
P ₁	0	4	6	10
P ₂	1	5	8	13
P ₃	2	2	2	4
P ₄	3	3	7	10

Gantt chart -

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

$$\text{Avg. TAT} = \frac{(10+13+4+10)}{4} = \frac{37}{4} = 9.25$$

Q5 Fork() → x = 5

- When the fork() system call is used, it creates a child process that has its own copy of the parent's memory.

- Before fork() → x = 5

→ After fork() → parent = 6 child = 6

→ In parent process, x = 6.

In child process, x = 6.