# CDAC MUMBAI

# Concepts of Operating System

# Assignment 2

## Part A

**What will the following commands do?**

**echo "Hello, World!"**

It will print Hello, World!

**name="Productive"**

**It will assign a value to variable.**

cdac@DESKTOP-HFLCO3Q:~$ name="Productive"

cdac@DESKTOP-HFLCO3Q:~$ echo $name

Productive

**touch file.txt**

It will create a file called file.txt.

**ls -a**

**It will list all contain including hiden contain.**

**rm file.txt**

**It will remove file.txt**

**cp file1.txt file2.txt**

**It will copy contain of file1.txt to file2.txt**

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

**It will move file.txt to directory.**

**chmod 755 script.sh**

**It will give permission to owner, group, other.**

**grep "pattern" file.txt**

**It will display word “pattern” if it exist in a file.txt otjerwise it won’t display anything.**

**kill PID**

**It will terminate process with ID.**

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

**it will first print Hello, World! Then it will change directory to mydir.**

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

**It will list out all the files having extension .txt**

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

**It will display all the contain of file1.txt & file2.txt but if there is any duplicate character or line then that will print or display together.**

**cdac@DESKTOP-HFLCO3Q:~$ cat a.txt b.txt | sort | uniq**

**Apaar**

**Apaar**

**Ayush Rajput**

**Deepak**

**Hemant**

**Lucky**

**Pranjal Thakral**

**cdac@DESKTOP-HFLCO3Q:~$ cat a.txt b.txt**

**Apaar**

**Ayush Rajput**

**Deepak**

**Hemant**

**Apaar**

**Hemant**

**Lucky**

**Pranjal Thakral**

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

**it will display all the files whose name starting with letter d.**

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

**It will search for a word pattern in a directory.**

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

**It will print a duplicate word or line which is present in both the files.**

**cdac@DESKTOP-HFLCO3Q:~$ cat a.txt**

**Apaar Ayush Rajput Deepak Hemant**

**cdac@DESKTOP-HFLCO3Q:~$ cat b.txt**

**Apaar Hemant Lucky Pranjal Thakral**

**cdac@DESKTOP-HFLCO3Q:~$ cat a.txt b.txt | sort | uniq -d**

**Apaar Hemant**

**chmod 644 file.txt**

**It will give permission to file as owner, group, and other.**

**cdac@DESKTOP-HFLCO3Q:~$ chmod 644 file.txt**

**cdac@DESKTOP-HFLCO3Q:~$ ls -l file.txt**

**-rw-r--r-- 1 cdac cdac 0 Aug 29 19:57 file.txt**

**cp -r source_directory destination_directory**

**It will copy source_directory to destination_directory containing all its contain and files.**

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

**It will display all the files having extension .txt**

**chmod u+x file.txt**

**It will give permission to owner to execute file as program.**

**echo $PATH**

**It will display current value of path.**

# 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**

**cd used to change 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 Incorrect Commands:**

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

**Correct command is chmod.**

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

   **Correct command is cp.**

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

   **Correct command is touch or nano.**

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

   **cat is used to concatenate file not catx.**

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

   **Mv is used to rename or move.**

# Part C

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

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a
root@DESKTOP-HFLCO3Q:~# bash a
Hello, World!
root@DESKTOP-HFLCO3Q:~#
```

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                                  a
echo "Hello, World!"

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

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

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a
root@DESKTOP-HFLCO3Q:~# bash a
CDAC Mumbai
root@DESKTOP-HFLCO3Q:~#
```

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                                  a
name="CDAC Mumbai"
echo $name

                          [ Read 2 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

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

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                                a
echo Enter a number
read num
echo Entered number is : $num

                              [ Read 3 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a
root@DESKTOP-HFLCO3Q:~# bash a
Enter a number
18
Entered number is : 18
root@DESKTOP-HFLCO3Q:~#
```

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

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                                a
num1=18
num2=7
sum=0
sum=`expr $num1 + $num2`
echo Result is : $sum

                              [ Read 5 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a
root@DESKTOP-HFLCO3Q:~# bash a
Result is : 25
root@DESKTOP-HFLCO3Q:~#
```

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

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
echo Enter a number :
read num
r=`expr $num % 2`
if [ $r -eq 0 ]
then
echo Even
else
echo Odd
fi

^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
Enter a number :
6
Even
root@DESKTOP-HFLCO3Q:~# bash a.sh
Enter a number :
3
Odd
root@DESKTOP-HFLCO3Q:~#
```

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

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
num=0
for num in 1 2 3 4 5
do
echo $num
done

                          [ Read 5 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
1
2
3
4
5
root@DESKTOP-HFLCO3Q:~#
```

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

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
num=1
while [ $num -lt 6 ]
do
echo $num
num=`expr $num + 1`
done

                          [ Read 6 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
1
2
3
4
5
root@DESKTOP-HFLCO3Q:~#
```

**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".**

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
if [ -f "file.txt" ]
then
echo File exists
else
echo File does not exist
fi

                          [ Read 6 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
File does not exist
root@DESKTOP-HFLCO3Q:~# touch file.txt
root@DESKTOP-HFLCO3Q:~# bash a.sh
File exists
root@DESKTOP-HFLCO3Q:~#
```

**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.**

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
num=6
if [ $num -gt 10 ]
then
echo $num is greater than 10.
else
echo $num is less than 10.
fi

                          [ Read 7 lines ]
^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
6 is less than 10.
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
16 is greater than 10.
root@DESKTOP-HFLCO3Q:~#
```

**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.**

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
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
root@DESKTOP-HFLCO3Q:~#
```

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                    a.sh
for i in 1 2 3 4 5
do
        for j in 1 2 3 4 5
        do
        echo -n "$((i * j)) "
        done
echo
done

                    [ Read 9 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```

**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.**

```
root@DESKTOP-HFLCO3Q: ~
  GNU nano 6.2                          a.sh
while true
do
        echo "Enter a number ( enter negative no. to terminate)"
        read num
        if [ "$num" -lt 0 ]
        then
                break
        fi
        echo "Square of $num is : $((num * num))"
done

                              [ Read 10 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

```
root@DESKTOP-HFLCO3Q: ~
root@DESKTOP-HFLCO3Q:~# nano a.sh
root@DESKTOP-HFLCO3Q:~# bash a.sh
Enter a number ( enter negative no. to terminate)
2
Square of 2 is : 4
Enter a number ( enter negative no. to terminate)
6
Square of 6 is : 36
Enter a number ( enter negative no. to terminate)
7
Square of 7 is : 49
Enter a number ( enter negative no. to terminate)
9
Square of 9 is : 81
Enter a number ( enter negative no. to terminate)
12
Square of 12 is : 144
Enter a number ( enter negative no. to terminate)
-6
root@DESKTOP-HFLCO3Q:~#
```

# Part E

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

**| Process | 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.**

| Process | Arrival Time | Burst Time | Wait Time | TAT |
|---|---|---|---|---|
| P1 | 0 | 5 | 0 | 5 |
| P2 | 1 | 3 | 4 | 7 |
| P3 | 2 | 6 | 6 | 12 |

**Average Wait Time = 3.3**

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

**| Process | Arrival Time | Burst Time |**

**| P1 | 0 | 3 |** **Calculate the average turnaround time**

**| P2 | 1 | 5 |** **using Shortest Job First (SJF) scheduling.**

**| P3 | 2 | 1 |**

**| P4 | 3 | 4 |**

| Process | A T | B T | Wait Time | CT | TAT |

|---|---|---|---|---|---|

| P1 | 0 | 3 | 0 | 3 | 3

| P2 | 1 | 5 | 7 | 13 | 12

| P3 | 2 | 1 | 1 | 4 | 2

| P4 | 3 | 4 | 1 | 8 | 5

P1 P3 P4 P2

0 3 4 8 13

Average TAT = 5.5

**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.**

| Process | A T | B T | Priority |Wait Time | CT | TAT |

|---|---|---|---|---|---|---|

| P1 | 0 | 6 | 3 | 7 | 13 | 13

| P2 | 1 | 4 | 1 | 0 | 5 | 4

| P3 | 2 | 7 | 4 | 11 | 20 | 18

| P4 | 3 | 2 | 2 | 2 | 7 | 4

Average Wait Time = 5

**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.**

| Process | A T | B T | Wait Time | CT | TAT |
|---|---|---|---|---|---|
| P1 | 0 | 4 | 12 | 10 | 10 |
| P2 | 1 | 5 | 14 | 15 | 14 |
| P3 | 2 | 2 | 2 | 6 | 4 |
| P4 | 3 | 3 | 8 | 13 | 10 |

Average TAT = 9.5

**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?**

**Before fork() call**

Value of parent = 5

Value of child = 5

**After fork() call**

Value of parent = 6 (5+1)

Value of child = 6 (5+1)

**Final value of both parent and child is 6.**