

CDAC MUMBAI
Concepts of Operating System
Assignment 2

Part A

Name – Harsh Khanvilkar

PG-DAC JUHU

echo "Hello, World!"

Prints Hello, World! to the terminal.

name="Productive"

Creates a variable name and assigns it the value Productive.

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 (those starting with.).

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. True - ls is used to list files and directories in a directory.
2. True - mv is used to move files and directories.
3. False - cd is used to change directories, not copy files and directories.
4. True - pwd stands for "print working directory" and displays the current directory.
5. True - grep is used to search for patterns in files.
6. True - chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
7. True - mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

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

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 (old name newname)

Part C - Shell Scripting Questions

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh2
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh2
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh2
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh2
Enter a number: 99
You entered: 99
cdac@DESKTOP-U1COC08:~/Assignment02$ _
```

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1  sh2
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh3
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh3
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh3
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh3
sh3: line 3: =3: syntax error: operand expected (error token is "=3")
Sum:
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh3
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh3
Sum: 8
cdac@DESKTOP-U1COC08:~/Assignment02$ _
```

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

cdac@DESKTOP-U1COC08: ~/Assignment02

```
cdac@DESKTOP-U1COC08:~/Assignment02$ cd
cdac@DESKTOP-U1COC08:~$ cd Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1  sh2  sh3
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh4
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh4
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh4
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh4
Enter a number: 67
Odd
cdac@DESKTOP-U1COC08:~/Assignment02$
```

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1  sh2  sh3  sh4
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh5
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh5
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh5
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh5
1
2
3
4
5
cdac@DESKTOP-U1COC08:~/Assignment02$ _
```

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1 sh2 sh3 sh4 sh5
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh6
sh6: line 1: i: command not found
sh6: line 2: [: missing `]'
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh6
sh6: line 1: i: command not found
sh6: line 2: [: missing `]'
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh6
i=1
while [ $i -le 5]; do
    echo "$i"
    ((i++))
done
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh6
sh6: line 2: [: missing `]'
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh6
1
2
3
4
5
cdac@DESKTOP-U1COC08:~/Assignment02$
```

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


```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
sh1 sh2 sh3 sh4 sh5 sh6
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh7
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh7
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh7
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh7
if [ -f "file.txt" ]; then
    echo "file exists"
else
    echo "File does not exist"
fi
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh7
File does not exist
cdac@DESKTOP-U1COC08:~/Assignment02$ touch file.txt
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh7
file exists
cdac@DESKTOP-U1COC08:~/Assignment02$
```

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
file.txt sh1 sh2 sh3 sh4 sh5 sh6 sh7
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh8
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh8
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh8
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh8
read -p "Enter a number: " num
if [ $num -gt 10 ]; then
    echo "Number is greater than 10"
else
    echo "Number is 10 or less"
fi
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh8
Enter a number: num 28
sh8: line 2: [: -gt: unary operator expected
Number is 10 or less
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh8
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh8
Enter a number: 78
Number is greater than 10
cdac@DESKTOP-U1COC08:~/Assignment02$
```

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

```

for i in {1..10}; do
    echo "$num x $i = $((num * i))"
done
echo ""
done
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh9
sh9: line 4: syntax error near unexpected token `echo'
sh9: line 4: `echo "Multiplication Table for $num:"'
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh9
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh9
for num in {1..5}; do
echo "Multiplication Table for $num:"
for i in {1..10}; do
echo "$num x $i = $((num * i))"
done
echo ""
done
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh9
Multiplication Table for 1:
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

Multiplication Table for 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Multiplication Table for 3:

```

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

```
cdac@DESKTOP-U1COC08: ~/Assignment02
2 x 9 = 18
2 x 10 = 20

Multiplication Table for 3:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Multiplication Table for 4:
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40

Multiplication Table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

cdac@DESKTOP-U1COC08:~/Assignment02$
```

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

cdac@DESKTOP-U1COC08: ~/Assignment02

```
cdac@DESKTOP-U1COC08:~$ cd Assignment02
cdac@DESKTOP-U1COC08:~/Assignment02$ ls
file.txt sh1 sh10 sh2 sh3 sh4 sh5 sh6 sh7 sh8 sh9
cdac@DESKTOP-U1COC08:~/Assignment02$ touch sh10
cdac@DESKTOP-U1COC08:~/Assignment02$ cat sh10
while true; do
    read -p "Enter a number:" num
    if [ $num -lt 0 ]; then
        break
    fi
    echo "Square: $((num * num))"
done
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh10
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh10
Enter a number (neagtive to quit): 98
sh10: line 13: square: command not found
Square of 98 is
Enter a number (neagtive to quit): vim sh10
sh10: line 7: [[: vim sh10: syntax error in expression (error token is "sh10")
sh10: line 13: vim sh10: syntax error in expression (error token is "sh10")
cdac@DESKTOP-U1COC08:~/Assignment02$ vim sh10
cdac@DESKTOP-U1COC08:~/Assignment02$ bash sh10
Enter a number (negative to quit): 98
Square of 98 is 9604
Enter a number (negative to quit): 34
Square of 34 is 1156
Enter a number (negative to quit): 67
Square of 67 is 4489
Enter a number (negative to quit): 49
Square of 49 is 2401
Enter a number (negative to quit): -30
Negative number entered. Exiting...
cdac@DESKTOP-U1COC08:~/Assignment02$
```

Part E

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

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

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

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Process	Arrival time	Burst time	Completion time	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

Process	AT	BT	CT	TAT	WT
P ₁	0	3	3	3	0
P ₂	1	5	13	12	7
P ₃	2	1	4	2	1
P ₄	3	4	8	5	1

P ₁	P ₃	P ₄	P ₂	Average TAT
0	3	4	8	13
				$\frac{22}{4}$
				$= 5.5$

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Calculate the average waiting time using Priority Scheduling.

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

$$\text{Avg WT} = \frac{0+5+10+7}{4} = 5.5$$

Gantt chart: P₁ P₂ P₄ P₃
 0 6 10 12 19

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

Calculate the average turnaround time using Round Robin scheduling.

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

$$\text{Average TAT} = \frac{35}{4} = 8.75$$

Ready Queue: P₁ P₂ P₃ P₁ P₄ P₂ P₄ P₂
 0 2 4 6 8 10 12 14

Gantt chart: P₁ P₂ P₃ P₁ P₄ P₂ P₄ P₂
 0 2 4 6 8 10 12 14

$x = 5$
 fork()

C1 P
 $x = 6$ $x = 6$

6. 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?

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 separate memory space and contain `x=5`.

Step 3:- After `fork()` execution:-
Both process will execute same step.
i.e. `x=x+1;`

since they have separate memory copies the changes do not affect across process.

Step 4:- Final value of `x`.

parent : value of `x=6`.

child :- value of `x=6`.