CDAC MUMBAI

Concepts of Operating System

Assignment 2

Name – Darshan Dhongade

PG-DAC KHARGHAR

Part A

Q. echo "Hello, World!"

Prints Hello, World! to the terminal.

Q. name="Productive"

Creates a variable name and assigns it the value Productive.

Q. touch file.txt

Creates an empty file named file.txt or updates its timestamp if it already exists.

Q. ls -a

Lists all files and directories in the current directory, including hidden ones (those starting with.).

Q. rm file.txt

Removes the file file.txt permanently**.**

Q. cp file1.txt file2.txt

Copies file1.txt to file2.txt. If file2.txtexists, it will be overwritten.

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

Moves file.txt to the specified directory.

Q. chmod 755 script.sh

Grants the owner full permissions (read, write, execute) and gives others read and execute permissions on script.sh.

Q. grep "pattern" file.txt

Searches for occurrences of "pattern" in file.txt and prints matching lines.

Q. kill PID

Terminates the process with the specified Process ID (PID).

Q. 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

Q. ls -l | grep ".txt"

Lists files in long format and filters only those containing ". Txt" in their names.

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

Concatenates file1.txtand file2.txt, sorts them, and removes duplicate lines.

Q. ls -l | grep "^d"

Lists directories (entries starting with d in long format output).

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

Searches for "pattern" recursively in all files under /path/to/directory/ .

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

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

Q. chmod 644 file.txt

Grants the owner read and write permissions, while others get read-only access tofile.txt .

Q. cp -r source_directory destination_directory

Recursively copies source_directoryto destination_directory, preserving contents.

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

Finds all .txtfiles in/path/to/searchand its subdirectories.

Q. chmod u+x file.txt

Gives the owner (u) execute permission on file.txt.

Q. 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.

Identify the incorrect commands

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

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh1
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ ls
hello  sh1
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh1
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ cat sh1
echo Hello World
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh1
Hello World
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ ls
hello  sh1  sh2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ cat sh2
echo Enter the String
read name
echo $name
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh2
Enter the String
CDAC Mumbai
CDAC Mumbai
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh3
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nono sh3
Command 'nono' not found, did you mean:
  command 'nano' from snap nano (7.2+pkg-4057)
  command 'nino' from snap nino (1.3.1)
  command 'nodo' from snap nodo (master)
  command 'nano' from deb nano (7.2-2ubuntu0.1)
  command 'mono' from deb mono-runtime (6.8.0.105+dfsg-3.5ubuntu1)
  command 'nona' from deb hugin-tools (2023.0.0+dfsg-1)
See 'snap info <snapname>' for additional versions.
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh3
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh3
Enter a number
5
5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh4
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh4
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh4
Enter 1st number
20
Enter 2nd number
10
sh4: line 6: Echo: command not found
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh4
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh4
Enter 1st number
20
Enter 2nd number
10
sh4: line 6: Echo: command not found
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh4
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh4
Enter 1st number
20
Enter 2nd number
15
Addition of 2 given numbers is 35
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh5
enter a number
10
The given number is odd
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh5
enter a number
10
sh5: line 3: ((: % 2 == 0: syntax error: operand expected (error token is "% 2 == 0")
The given number is odd
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh5
enter a number
10
sh5: line 3: ((: % 2 == 0: syntax error: operand expected (error token is "% 2 == 0")
The given number is odd
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh5
enter a number
10
The given number is even
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh6
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh6
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh6
1
2
3
4
5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ cat sh7
num=1

while [ $num -le 5 ]
do
        echo $num
        ((num++))
done
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh7
1
2
3
4
5
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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".

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh8
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh8
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh8
sh8: line 4: unexpected EOF while looking for matching `''
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh8
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh8
file does not exists
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ ls
hello  sh1  sh2  sh3  sh4  sh5  sh6  sh7  sh8
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch file1.txt
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh8
File exists
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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.

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ touch sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh9
enter a number
5
sh9: line 3: [: missing `]'
number is bigger than 10
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh9
enter a number
5
sh9: line 3: [: num: integer expression expected
number is bigger than 10
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh9
enter a number
5
sh9: line 3: [: -lt: unary operator expected
number is bigger than 10
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh99
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ nano sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh9
enter a number
5
Given number is less than 10
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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.

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ ls
file1.txt  hello  sh1  sh10  sh2  sh3  sh4  sh5  sh6  sh7  sh8  sh9
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ cat sh10
for num in {1..5}; do
        echo "Multiplication Table for $num"
        for i in {1..10}; do
                echo "$num x $i = $((num * i))"
        done
        echo ""
done
darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$ bash sh10
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
```

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
2 x 7 = 14
2 x 8 = 16
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
```

```
darshan@DESKTOP-B7Q28RG: ~/Linux_Assignment_2
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

darshan@DESKTOP-B7Q28RG:~/Linux_Assignment_2$
```

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.

Process	Arrival Time	Burst Time	Complet^n Time	TAT	WT
P_1	0	5	5	5	0
P_2	1	3	8	7	4
P_3	2	6	14	12	6
					$\boxed{3.33}$

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	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

0 3 4 8 13

Average TAT $= \frac{22}{4} = 5.5$

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	Arrival Time	Burst Time	Priority	Complet^n Time	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

Average WT $= \frac{5+10+7}{4} = 5.5$

P_1	P_2	P_4	P_3

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	Arrival Time	Burst Time	Compl^n Time	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

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

$= 8.75$

Ready Queue

~~P_1~~	~~P_2~~	~~P_3~~	~~P_1~~	~~P_4~~	~~P_2~~	P_4	P_2

0 2 3 3 1 1

Gantt Chart

P_1	P_2	P_3	P_1	P_4	P_2	P_4	P_2

0 2 4 6 8 10 12 13 14

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?

Q5.

Step1: Before fork() is called

int x=5;

Step 2: Calling fork()

- fork() system call create a new child
- Both parent & child have seperate memory space and contain x=5.

Step3: After fork() execution:

- Both process will execute same step i.e x=x+1;
- since they have seperate memory copies the chage do not affect across process.

Step4: Final value of x

parent : value of x=6

child : value of x=6

Even though both process increment x
It is done in their own independent memory space.
So the final value remain 6 in both process.