Concepts Of Operating System

Assignment-2

Part - A

1. echo "Hello, World!"

```
cdac@LAPTOP-PLD6211J:~$ echo "Hello, World!"
Hello, World!
```

2. name="Productive"

```
cdac@LAPTOP-PLD6211J:~$ name="Productive"
cdac@LAPTOP-PLD6211J:~$ echo $name
Productive
```

3. touch file.txt

```
cdac@LAPTOP-PLD6211J:~$ touch file.txt
cdac@LAPTOP-PLD6211J:~$ ls
LinuxAssignment  file.txt
```

4. ls -a

```
cdac@LAPTOP-PLD6211J:~$ ls -a
.               .bash_logout   .local        .sudo_as_admin_successful
..              .bashrc        .motd_shown   LinuxAssignment
.bash_history   .cache         .profile      file.txt
```

5. rm file.txt

```
cdac@LAPTOP-PLD6211J:~$ ls
LinuxAssignment  file.txt
cdac@LAPTOP-PLD6211J:~$ rm file.txt
cdac@LAPTOP-PLD6211J:~$ ls
LinuxAssignment
```

6. cp file1.txt file2.txt

```
cdac@LAPTOP-PLD6211J:~$ nano file1.txt
cdac@LAPTOP-PLD6211J:~$ cat file1.txt
Chitransh Mrigank Singh
cdac@LAPTOP-PLD6211J:~$ touch file2.txt
cdac@LAPTOP-PLD6211J:~$ cat file2.txt
cdac@LAPTOP-PLD6211J:~$ cp file1.txt file2.txt
cdac@LAPTOP-PLD6211J:~$ cat file2.txt
Chitransh Mrigank Singh
```

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

```
cdac@LAPTOP-PLD6211J:~$ pwd
/home/cdac
cdac@LAPTOP-PLD6211J:~$ ls
LinuxAssignment  Mumbai  file.txt  file1.txt  file2.txt
cdac@LAPTOP-PLD6211J:~$ mv file.txt /home/cdac/Mumbai
cdac@LAPTOP-PLD6211J:~$ cd Mumbai
cdac@LAPTOP-PLD6211J:~/Mumbai$ ls
file.txt
```

8. chmod 755 script.sh

chmod 755 command in Linux grants the owner of a file or directory full permissions, while giving the group and others read and execute permissions.

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ touch script.sh
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ ls -l
total 0
-rw-r--r-- 1 cdac cdac 0 Aug 30 15:59 script.sh
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ chmod 755 script.sh
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ ls -l
total 0
-rwxr-xr-x 1 cdac cdac 0 Aug 30 15:59 script.sh
```

9. grep "pattern" file.txt

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ nano file.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ grep "pattern" file.txt
pattern 123
12pattern34
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ cat file.txt
pattern 123
classwork
2024
12pattern34
```

10. kill PID

This command terminate the process with the given PID

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

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ mkdir mydir && cd mydir && touch file.txt &&
echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2/mydir$ ls
file.txt
```

This creates “mydir” directory and enter into it. The it creates a file named “file.txt”. The output echo statement is redirected to “file.txt” and also displayed on the console.

12. ls -l | grep ".txt"

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment$ ls
data.txt  docs.zip       extract    fruit.txt  numbers.txt
docs      duplicate.txt  file1.txt  input.txt  output.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment$ ls -l | grep ".txt"
-rw-rw-r-- 1 cdac cdac  100 Aug 29 11:03 data.txt
-rw-rw-r-- 1 cdac cdac  132 Aug 29 13:04 duplicate.txt
-rw-r--r-- 1 cdac cdac   62 Aug 29 06:50 file1.txt
-rw-rw-r-- 1 cdac cdac   55 Aug 29 20:54 fruit.txt
-rw-rw-r-- 1 cdac cdac   43 Aug 29 19:43 input.txt
-rw-rw-r-- 1 cdac cdac   68 Aug 29 11:17 numbers.txt
-rw-rw-r-- 1 cdac cdac   43 Aug 29 19:43 output.txt
```

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

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ nano file1.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ cat file1.txt
cat
dog
horse
cat
monkey
monkey
dog
```

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ nano file2.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ cat file2.txt
tiger
lion
deer
tiger
tiger
deer
leopard
```

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ cat file1.txt file2.txt | sort | uniq
cat
deer
dog
horse
leopard
lion
monkey
tiger
```

14. ls -l | grep "^d"

Shows all directories in the current directory. That's because the ls -l adds a d in the beginning of the directories info.

```
cdac@LAPTOP-PLD6211J:~$ cd /
cdac@LAPTOP-PLD6211J:/$ ls -l | grep "^d"
drwxr-xr-x   2 root root    4096 Apr 18  2022 boot
drwxr-xr-x  16 root root    3560 Aug 30 15:00 dev
drwxr-xr-x  73 root root    4096 Aug 30 15:00 etc
drwxr-xr-x   4 root root    4096 Aug 29 08:08 home
drwx------   2 root root   16384 Aug 28 14:19 lost+found
drwxr-xr-x   2 root root    4096 Nov 23  2023 media
drwxr-xr-x   6 root root    4096 Aug 28 14:19 mnt
drwxr-xr-x   2 root root    4096 Nov 23  2023 opt
dr-xr-xr-x 211 root root       0 Aug 30 15:00 proc
drwx------   5 root root    4096 Aug 28 19:04 root
drwxr-xr-x  18 root root     540 Aug 30 15:00 run
drwxr-xr-x   8 root root    4096 Nov 23  2023 snap
drwxr-xr-x   2 root root    4096 Nov 23  2023 srv
dr-xr-xr-x  11 root root       0 Aug 30 15:00 sys
drwxrwxrwt  10 root root    4096 Aug 30 15:10 tmp
drwxr-xr-x  14 root root    4096 Nov 23  2023 usr
drwxr-xr-x  13 root root    4096 Nov 23  2023 var
```

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

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ grep -r monkey /home/cdac/LinuxAssignment2
/home/cdac/LinuxAssignment2/file1.txt:monkey
/home/cdac/LinuxAssignment2/file1.txt:monkey
```

16. chmod 644 file.txt

The chmod 644 command in Linux sets file permissions to give the owner read and write access, and read-only access to everyone else.

```
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ chmod 755 file.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ ls -l
total 12
-rwxr-xr-x 1 cdac cdac    0 Aug 30 20:28 file.txt
-rw-r--r-- 1 cdac cdac   36 Aug 30 17:32 file1.txt
-rw-r--r-- 1 cdac cdac   41 Aug 30 17:42 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Aug 30 16:20 mydir
-rwxr-xr-x 1 cdac cdac    0 Aug 30 15:59 script.sh
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ chmod 644 file.txt
cdac@LAPTOP-PLD6211J:~/LinuxAssignment2$ ls -l
total 12
-rw-r--r-- 1 cdac cdac    0 Aug 30 20:28 file.txt
-rw-r--r-- 1 cdac cdac   36 Aug 30 17:32 file1.txt
-rw-r--r-- 1 cdac cdac   41 Aug 30 17:42 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Aug 30 16:20 mydir
-rwxr-xr-x 1 cdac cdac    0 Aug 30 15:59 script.sh
```

17. cp -r source_directory destination_directory

```
cdac@LAPTOP-PLD6211J:~$ pwd
/home/cdac
cdac@LAPTOP-PLD6211J:~$ ls
LinuxAssignment  LinuxAssignment2  Mumbai  ShellProgramming  file1.txt  file2.txt
cdac@LAPTOP-PLD6211J:~$ cp -r /home/cdac/LinuxAssignment /home/cdac/Mumbai
cdac@LAPTOP-PLD6211J:~$ cd Mumbai
cdac@LAPTOP-PLD6211J:~/Mumbai$ ls
```

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

```
cdac@LAPTOP-PLD6211J:~$ find /home/cdac/LinuxAssignment -name "*.txt"
/home/cdac/LinuxAssignment/docs/file2.txt
/home/cdac/LinuxAssignment/output.txt
/home/cdac/LinuxAssignment/extract/docs/file2.txt
/home/cdac/LinuxAssignment/extract/file1.txt
/home/cdac/LinuxAssignment/numbers.txt
/home/cdac/LinuxAssignment/duplicate.txt
/home/cdac/LinuxAssignment/data.txt
/home/cdac/LinuxAssignment/file1.txt
/home/cdac/LinuxAssignment/input.txt
/home/cdac/LinuxAssignment/fruit.txt
```

19. chmod u+x file.txt

```
cdac@LAPTOP-PLD6211J:~$ touch file.txt
cdac@LAPTOP-PLD6211J:~$ ls -l
total 24
drwxr-xr-x 4 cdac cdac 4096 Aug 29 21:30 LinuxAssignment
drwxr-xr-x 3 cdac cdac 4096 Aug 30 20:44 LinuxAssignment2
drwxrwxr-x 3 cdac cdac 4096 Aug 30 20:52 Mumbai
drwxr-xr-x 2 cdac cdac 4096 Aug 30 15:48 ShellProgramming
-rw-r--r-- 1 cdac cdac    0 Aug 30 21:01 file.txt
-rw-rw-r-- 1 cdac cdac   24 Aug 29 23:47 file1.txt
-rw-rw-r-- 1 cdac cdac   24 Aug 29 23:48 file2.txt
cdac@LAPTOP-PLD6211J:~$ chmod u+x file.txt
cdac@LAPTOP-PLD6211J:~$ ls -l
total 24
drwxr-xr-x 4 cdac cdac 4096 Aug 29 21:30 LinuxAssignment
drwxr-xr-x 3 cdac cdac 4096 Aug 30 20:44 LinuxAssignment2
drwxrwxr-x 3 cdac cdac 4096 Aug 30 20:52 Mumbai
drwxr-xr-x 2 cdac cdac 4096 Aug 30 15:48 ShellProgramming
-rwxr--r-- 1 cdac cdac    0 Aug 30 21:01 file.txt
-rw-rw-r-- 1 cdac cdac   24 Aug 29 23:47 file1.txt
-rw-rw-r-- 1 cdac cdac   24 Aug 29 23:48 file2.txt
```

20. echo $PATH

```
cdac@LAPTOP-PLD6211J:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/gam
es:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/wind
ows/system32:/mnt/c/windows:/mnt/c/windows/System32/Wbem:/mnt/c/windows/System32/Windo
wsPowerShell/v1.0/:/mnt/c/windows/System32/OpenSSH/:/mnt/c/Program Files (x86)/NVIDIA
Corporation/PhysX/Common:/mnt/c/Program Files/NVIDIA Corporation/NVIDIA NvDLISR:/mnt/c
/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/
WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/ProgramData/chocolatey
/bin:/mnt/c/Program Files/MySQL/MySQL Server 8.0/bin:/mnt/c/Program Files/Java/jdk-11.
0.12/bin:/mnt/c/Program Files/Git/cmd:/mnt/c/Program Files/dotnet/:/mnt/c/Program File
s/Microsoft SQL Server/150/Tools/Binn/:/mnt/c/Program Files/HP/HP One Agent:/mnt/c/Pro
gram Files/nodejs/:/mnt/c/Program Files/MySQL/MySQL Shell 8.0/bin/:/mnt/c/Users/chitr/
AppData/Local/Microsoft/WindowsApps:/mnt/c/Program Files/JetBrains/IntelliJ IDEA Commu
nity Edition 2022.3.2/bin:/mnt/d/Program Files/Microsoft VS Code/bin:/mnt/c/Users/chit
r/.dotnet/tools:/mnt/c/Users/chitr/AppData/Roaming/npm:/snap/bin
```

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**
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. **INCORRECT**
2. cpy is used to copy files and directories. **INCORRECT**
3. mkfile is used to create a new file. **INCORRECT**
4. catx is used to concatenate files. **INCORRECT**
5. rn is used to rename files. **CORRECT**

Part-C

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

```
echo Hello! World!
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a1
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a1
Hello! World!
```

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

```
echo Enter name
read name
echo name = $name
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a2
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a2
Enter name
CDAC Mumbai
name = CDAC Mumbai
```

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

```
echo Enter a number
read num
echo You entered: $num
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a3
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a3
Enter a number
9
You entered: 9
```

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

```
echo Enter num1
read num1
echo Enter num2
read num2
sum=$(( $num1 + $num2 ))
echo Sum of $num1 and $num2 is $sum
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a4
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a4
Enter num1
5
Enter num2
3
Sum of 5 and 3 is 8
```

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

```
echo Enter num
read num
if [ $(( $num % 2)) -eq 0 ]
then
        echo Even
else
        echo Odd
fi
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a5
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a5
Enter num
5
Odd
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a5
Enter num
12
Even
```

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

```
for (( i=1; i <= 5; i++ ))
do
        echo $i
done
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a6
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a6
1
2
3
4
5
```

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

```
a=1
while [ $a -le 5 ]
do
        echo $a
        a=$(( $a + 1 ))
done
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a7
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a7
1
2
3
4
5
```

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".

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

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a8
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ ls
a1  a2  a3  a4  a5  a6  a7  a8  apple  banana  p1  p10
p11  p12  p13  p14  p2  p3  p4  p5  p6  p7  p8  p9
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a8
File does not exist
```

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.

```
echo Enter num
read num
if [ $num -gt 10 ]
then
        echo $num is greater than 10
else
        echo $num is less than 10
fi
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a9
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a9
Enter num
11
11 is greater than 10
```

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.

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a10
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a10
--------------------
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
--------------------
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
--------------------
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
--------------------
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
--------------------
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
```

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.

```
while true
do
        echo "Enter a number (enter a negative number to exit): "
        read num

        if [ $num -lt 0 ]
        then
                break
        else
                echo $(( num * num ))
        fi
done
echo Exited because you entered a negative number
```

```
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ nano a11
cdac@LAPTOP-PLD6211J:~/ShellProgramming$ bash a11
Enter a number (enter a negative number to exit):
9
81
Enter a number (enter a negative number to exit):
3
9
Enter a number (enter a negative number to exit):
-1
Exited because you entered a negative number
```

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
	P1	0	5	0
	P2	1	3	5-1=4
	P3	2	6	8-2=6
	Gantt Chart			
	P1	P2	P3	
0	5	8	14	
	Waiting Time = Allocation Time - Arrival Time			
	Avg WT = (0+4+6)/3 =3.33			

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

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

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

	Process	Arrival Time	Burst Time	TAT
	P1	0	3	3
	P2	1	5	4-2=2
	P3	2	1	8-3=5
	P4	3	4	13-3=10
	Gantt Chart			
	P1	P3	P4	P2
0	3	4	8	13
	Turn Around Time = Completion Time - Arrival Time			
	Avg TAT = (3+2+5+10)/4 = 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	Arrival Time	Burst Time	Priority	Wait
	P1	0	6	3	0+6=6
	P2	1	4	1	0
	P3	2	7	4	12-2=10
	P4	3	2	2	5-3=2

Gantt Chart

	P1	P2	P4	P1	P3
0	1	5	7	12	19

Waiting Time = Allocation Time - Arrival Time

Avg WT = (6+0+10+2)/4 = 4.5

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	Arrival Time	Burst Time	TAT
	P1	0	4	10-0=10
	P2	1	5	17-1=18
	P3	2	2	6-2=4
	P4	3	3	18-3=15

Gantt Chart

	P1	P2	P3	P4	P1	P2	P4	P2	P4
0	2	4	6	8	10	14	16	17	18

Turn Around Time - Completion Time - Arrival Time

Avg TAT = (10+18+4+15)/4 = 11.75

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?

Final value of x for both child parent and child process will be 6.