

Assignment – 2

COS

PART – A

What will the following commands do?

- **echo “Hello, World!”**

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

- **name="Productive"**

It assigns the string productive to name variable which can be printed by echo.

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

- **touch file.txt**

It creates a file named file.txt on which we can write our instructions.

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

- **ls -a**

It lists all files and directories in the current directory, including hidden files.

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

- **rm file.txt**

It removes the file named file.txt

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

- **cp file1.txt file2.txt**

It copies the contents of file 1 into file 2.

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ touch file1.txt  
cdac@LAPTOP-BFJCJVNG:~$ touch file2.txt  
cdac@LAPTOP-BFJCJVNG:~$ cp file1.txt file2.txt  
cdac@LAPTOP-BFJCJVNG:~$ ls  
LinuxAssignment file1.txt file2.txt  
cdac@LAPTOP-BFJCJVNG:~$
```

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

It moves a file to a particular directory.

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~$ touch file.txt  
cdac@LAPTOP-BFJCJVNG:~$ mv file.txt /home/cdac/LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~$ ls  
LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~$ cd LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ ls  
data.txt docs duplicate.txt file.txt file1.txt  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

- **chmod 755 script.sh**

It gives read, write, execute permission to the main user and only read and execute permission to group and other user with respect to script.sh file.

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ touch script.sh  
cdac@LAPTOP-BFJCJVNG:~$ chmod 755 script.sh  
cdac@LAPTOP-BFJCJVNG:~$ ls -l  
total 0  
drwxr-xr-x 1 cdac cdac 4096 Mar  1 17:35 LinuxAssignment  
-rwxr-xr-x 1 cdac cdac   0 Mar  1 17:43 script.sh  
cdac@LAPTOP-BFJCJVNG:~$
```

- **grep “pattern” file.txt**

It searches for pattern word inside the file.txt

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ nano file.txt  
cdac@LAPTOP-BFJCJVNG:~$ grep "pattern" file.txt  
There is pattern in everything every pattern tells something about it's pattern.  
cdac@LAPTOP-BFJCJVNG:~$
```

- **kill PID**

It is used to terminate a process with a specific Process ID (PID).

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

Creates a new directory named mydir, moves into newly created directory, creates an empty file named file.txt, writes “Hello, World!” into file.txt, displays the content of file.txt in the terminal.

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

- **ls -l | grep “.txt”**

It lists all .txt files in the current directory with detailed information

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ ls -l | grep ".txt"  
-rw-r--r-- 1 cdac cdac 27 Feb 28 19:26 data.txt  
-rw-r--r-- 1 cdac cdac 72 Feb 28 19:31 duplicate.txt  
-rw-r--r-- 1 cdac cdac 0 Mar 1 17:35 file.txt  
-rw-r--r-- 1 cdac cdac 79 Feb 28 19:20 file1.txt  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

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

It sorts the file1.txt and file2.txt by clubbing all the duplications and uniq will remove the duplications.

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ cat file1.txt file2.txt | sort | uniq  
Bye  
Hello  
Hello Aditya sir  
Hello CDAC Mumbai Kharghar  
Hello Malkeet sir  
Hello Ravi Yadav  
Hi  
Juhu  
Kharghar  
Mumbai  
Nasik  
Pune  
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

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

It is used to show current directories with the user.

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ ls -l | grep "^d"
drwxr-xr-x 1 cdac cdac 4096 Feb 28 08:55 docs
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

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

It is used to search for a specific pattern inside all files within a given directory and its subdirectories.

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

is used to find duplicate lines that appear in both file1.txt and file2.txt

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ cat file1.txt file2.txt | sort | uniq -d
uniq-d: command not found
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ ls
data.txt  docs  duplicate.txt  file.txt  file1.txt  file2.txt
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ nano file1.txt
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ nano file2.txt
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ cat file1.txt file2.txt | sort | uniq -d
Hello
Mumbai
Nasik
Pune
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

- **chmod 644 file.txt**

It changes the file permission of the user to read and write, and of group and other user to only read.

```
cdac@LAPTOP-BFJCJVNG: ~/LinuxAssignment
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ chmod 644 file.txt
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$ ls -l
total 0
-rw-r--r-- 1 cdac cdac 27 Feb 28 19:26 data.txt
drwxr-xr-x 1 cdac cdac 4096 Feb 28 08:55 docs
-rw-r--r-- 1 cdac cdac 72 Feb 28 19:31 duplicate.txt
-rw-r--r-- 1 cdac cdac 0 Mar 1 17:35 file.txt
-rw-r--r-- 1 cdac cdac 100 Mar 1 18:11 file1.txt
-rw-r--r-- 1 cdac cdac 61 Mar 1 18:25 file2.txt
cdac@LAPTOP-BFJCJVNG:~/LinuxAssignment$
```

- **cp -r source_directory destination_directory**

It is used to copy a directory and its contents (including subdirectories and files) to another location in Linux.

```
cdac@LAPTOP-BFJCJVNG: ~/mydir
cdac@LAPTOP-BFJCJVNG:~$ ls
LinuxAssignment file.txt mydir script.sh
cdac@LAPTOP-BFJCJVNG:~$ cp -r /home/cdac/LinuxAssignment/file.txt /home/cdac/mydir/file1.txt
cdac@LAPTOP-BFJCJVNG:~$ cat file1.txt
cat: file1.txt: No such file or directory
cdac@LAPTOP-BFJCJVNG:~$ cd mydir
cdac@LAPTOP-BFJCJVNG:~/mydir$ ls
file.txt file1.txt
cdac@LAPTOP-BFJCJVNG:~/mydir$
```

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

It will find .txt file in the given directory.

```
cdac@LAPTOP-BFJCJVNG: ~/mydir
cdac@LAPTOP-BFJCJVNG:~/mydir$ find /home/cdac/mydir -name "*.txt"
/home/cdac/mydir/file.txt
/home/cdac/mydir/file1.txt
cdac@LAPTOP-BFJCJVNG:~/mydir$
```

- **chmod u+x file.txt**

It gives user the permission to execute with respect to file.txt.

```
cdac@LAPTOP-BFJCJVNG: ~/mydir
cdac@LAPTOP-BFJCJVNG:~/mydir$ chmod u+x file.txt
cdac@LAPTOP-BFJCJVNG:~/mydir$ ls -l
total 0
-rwxr--r-- 1 cdac cdac 14 Mar  1 18:02 file.txt
-rw-r--r-- 1 cdac cdac  0 Mar  2 11:47 file1.txt
cdac@LAPTOP-BFJCJVNG:~/mydir$
```

- **echo \$PATH**

It will print the environment variable that stores a list of directories where executable files (commands, scripts, programs) are located.

```
cdac@LAPTOP-BFJCJVNG: ~
avapath:/mnt/c/Program Files (x86)/Intel/iCLS Client:/mnt/c/Program Files/Intel/iCLS Client:/mnt/c/windows/system32:/mnt/c/windows:/mnt/c/windows/System32/Wbem:/mnt/c/windows/System32/WindowsPowerShell/v1.0:/mnt/c/Program Files (x86)/Intel/Intel(R) Management Engine Components/DAL:/mnt/c/Program Files/Intel/Intel(R) Management Engine Components/DAL:/mnt/c/Program Files (x86)/Intel/Intel(R) Management Engine Components/IPT:/mnt/c/Program Files/Intel/Intel(R) Management Engine Components/IPT:/mnt/c/Program Files (x86)/NVIDIA Corporation/PhysX/Common:/mnt/c/Program Files/Intel/WiFi/bin:/mnt/c/Program Files/Common Files/Intel/WirelessCommon:/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/MinGW/bin:/mnt/c/TDM-GCC-64/bin:/mnt/c/Users/Ravi Yadav/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/Ravi Yadav/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/Ravi Yadav/AppData/Local/GitHubDesktop/bin:/snap/bin
cdac@LAPTOP-BFJCJVNG:~$
```

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. (True)
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) → **chmod** (Correct)
2. **cpy** is used to copy files and directories. (Incorrect) → **cp** (Correct)
3. **mkfile** is used to create a new file. (Correct)
4. **catx** is used to concatenate files. (Incorrect) → **cat** (Correct)
5. **rn** is used to rename files. (Correct)

PART – C

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

```
cdac@LAPTOP-BFJCJVN: ~  
cdac@LAPTOP-BFJCJVN:~$ nano hello.sh  
cdac@LAPTOP-BFJCJVN:~$ bash hello.sh  
Hello, World!  
cdac@LAPTOP-BFJCJVN:~$
```

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

```
cdac@LAPTOP-BFJCJVN: ~  
cdac@LAPTOP-BFJCJVN:~$ cat name.sh  
name="CDAC Mumbai"  
echo "$name"  
cdac@LAPTOP-BFJCJVN:~$ bash name.sh  
CDAC Mumbai  
cdac@LAPTOP-BFJCJVN:~$
```

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

```
cdac@LAPTOP-BFJCJVN: ~  
cdac@LAPTOP-BFJCJVN:~$ cat num.sh  
echo Enter a number  
read number  
echo "Your entered number is : $number"  
cdac@LAPTOP-BFJCJVN:~$ bash num.sh  
Enter a number  
28  
Your entered number is : 28  
cdac@LAPTOP-BFJCJVN:~$
```

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

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ cat sum.sh  
echo Enter a number1  
read number1  
echo Enter a number2  
read number2  
sum=$((number1 + number2))  
echo "The sum of number1 and number2 : $sum"  
cdac@LAPTOP-BFJCJVNG:~$ bash sum.sh  
Enter a number1  
5  
Enter a number2  
3  
The sum of number1 and number2 : 8  
cdac@LAPTOP-BFJCJVNG:~$
```

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

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ cat odd.sh  
echo "Enter a number"  
read num  
if ((num % 2 == 0)); then  
echo "Even"  
else  
echo "Odd"  
fi  
cdac@LAPTOP-BFJCJVNG:~$ bash odd.sh  
Enter a number  
8  
Even  
cdac@LAPTOP-BFJCJVNG:~$
```

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

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ cat for.sh  
for i in {1..5}; do  
echo $i  
done  
cdac@LAPTOP-BFJCJVNG:~$ bash for.sh  
1  
2  
3  
4  
5  
cdac@LAPTOP-BFJCJVNG:~$
```


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

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ rm for.sh  
cdac@LAPTOP-BFJCJVNG:~$ nano while.sh  
cdac@LAPTOP-BFJCJVNG:~$ cat while.sh  
i=1  
while [ $i -le 5 ]; do  
echo $i  
((i++))  
done  
cdac@LAPTOP-BFJCJVNG:~$ bash while.sh  
1  
2  
3  
4  
5  
cdac@LAPTOP-BFJCJVNG:~$
```

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".

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ nano exist.sh  
cdac@LAPTOP-BFJCJVNG:~$ cat exist.sh  
if [ -f "file.txt" ]; then  
echo "File exists"  
else  
echo "File does not exist"  
fi  
cdac@LAPTOP-BFJCJVNG:~$ bash exist.sh  
File exists  
cdac@LAPTOP-BFJCJVNG:~$
```

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.

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ nano greater.sh  
cdac@LAPTOP-BFJCJVNG:~$ cat greater.sh  
echo "Enter a number: "  
read num  
if [ "$num" -gt 10 ]; then  
echo "The number is greater than 10"  
else  
echo "The number is not greater than 10"  
fi  
cdac@LAPTOP-BFJCJVNG:~$ bash greater.sh  
Enter a number:  
15  
The number is greater than 10  
cdac@LAPTOP-BFJCJVNG:~$
```

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-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ nano multi.sh  
cdac@LAPTOP-BFJCJVNG:~$ bash multi.sh  
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  
6 12 18 24 30  
7 14 21 28 35  
8 16 24 32 40  
9 18 27 36 45  
10 20 30 40 50  
cdac@LAPTOP-BFJCJVNG:~$
```

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.

```
cdac@LAPTOP-BFJCJVNG: ~  
cdac@LAPTOP-BFJCJVNG:~$ nano last.sh  
cdac@LAPTOP-BFJCJVNG:~$ cat last.sh  
while true; do  
echo "Enter a number (negative to exit);"  
read num  
if [ "$num" -lt 0 ]; then  
echo "Negative number entered. Exiting.."  
break  
fi  
square=$((num * num))  
echo "Square of $num is: $square"  
done  
cdac@LAPTOP-BFJCJVNG:~$ bash last.sh  
Enter a number (negative to exit);  
3  
Square of 3 is: 9  
Enter a number (negative to exit);  
5  
Square of 5 is: 25  
Enter a number (negative to exit);  
8  
Square of 8 is: 64  
Enter a number (negative to exit);  
-2  
Negative number entered. Exiting..  
cdac@LAPTOP-BFJCJVNG:~$
```

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.

Answer 1) Avg. Waiting Time = 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.

Answer 2) Avg Turnaround Time = 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.

Answer 3) Avg. Waiting Time = 7.75.

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.

Answer 4) Avg. Turnaround time = 9.

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?

Answer 5)

Let $x = 5$.

Call `fork()`.

Both parent and child increase x by 1.

child $x = 6$.

parent $x = 6$.

Final values of x :

child process, $x = 6$.

parent process, $x = 6$.