CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  - Just Print Same as it
  - Output

```
arati@Arati:~$ echo "Hello, World!"
Hello, World!
```

- name="Productive"
  - in that value is stored in variable
  - it also called as a value stored in shell variable
  - value Productive is stored in variable name
  - Output

```
arati@Arati:~$ name="Productive"
arati@Arati:~$ echo $name
Productive
```

- touch file.txt
  - It just touch the file.txt means create the file
  - Output

```
arati@Arati:~$ touch file.txt
arati@Arati:~$
```

- ls -a
  - It List all the files and directory of that directory
  - Output

```
arati@Arati:~/LinuxAssignment$ ls -a
.           docs           file.txt    input.txt
..          duplicate.txt  file1.txt   numbers.txt
data.txt    duplicate.zip  fruit.txt   output.txt
```

- rm file.txt
  - remove the file file.txt from directory
  - Output

```
arati@Arati:~/LinuxAssignment$ rm file.txt
arati@Arati:~/LinuxAssignment$ ls -a
.          docs           file1.txt  numbers.txt
..         duplicate.txt  fruit.txt  output.txt
data.txt   duplicate.zip  input.txt
```

- cp file1.txt file2.txt
  - In this command copy file1.txt to file2.txt
  - Output

```
arati@Arati:~/LinuxAssignment$ cp file1.txt file2.txt
arati@Arati:~/LinuxAssignment$ cat file2.txt
Hello, EveryOne Cdac Students
arati@Arati:~/LinuxAssignment$
```

- mv file.txt /path/to/directory/
  - in that move file to the another directory
  - Output

```
arati@Arati:~/LinuxAssignment$ mv file.txt docs/
arati@Arati:~/LinuxAssignment$ cd docs/
arati@Arati:~/LinuxAssignment/docs$ ls
duplicate.txt  file.txt  file2.txt
```

- chmod 755 script.sh
  - change the permission it gives only read and execute permission to group and other user
  - Output

```
arati@Arati:~/LinuxAssignment$ touch script.sh
arati@Arati:~/LinuxAssignment$ chmod 755 script.sh
arati@Arati:~/LinuxAssignment$ touch script.sh
arati@Arati:~/LinuxAssignment$ ls -l script.sh
-rwxr-xr-x 1 arati arati 0 Feb 28 13:42 script.sh
```

  -
  -

- grep "pattern" file.txt
  - It highlight the word which is available in that file
  - Output

```
arati@Arati:~/LinuxAssignment$ grep "Cdac" file1.txt
Hello, EveryOne Cdac Students
```

- kill PID
  - this command is used to kill the process by using PID
  - Output

```
arati@Arati:~/LinuxAssignment$ kill PID
-bash: kill: PID: arguments must be process or job IDs
```

- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
    - make directory mydir and change directory to mydir and then create file.txt then echo statement “Hello, World!” this will redirect to file.txt file and last print that text from file.txt
    - Output

```
arati@Arati:~/LinuxAssignment$ mkdir mydir && cd mydir
&& touch file.txt && echo "Hello, World!" > file.txt &&
 cat file.txt
Hello, World!
```

- ls -l | grep ".txt"
    - in that list all the .txt file with detaails
    - Output

```
arati@Arati:~/LinuxAssignment$ ls -l | grep ".txt"
-rw-r--r-- 1 arati arati 1663 Feb 27 13:18 data.txt
-rw-r--r-- 1 arati arati  141 Feb 27 13:50 duplicate.txt
-rwxr--r-- 1 arati arati   30 Feb 27 16:48 file1.txt
-rwxr--r-- 1 arati arati   30 Feb 28 12:13 file2.txt
-rw-r--r-- 1 arati arati  152 Feb 27 13:58 fruit.txt
-rw-r--r-- 1 arati arati   22 Feb 27 13:43 input.txt
-rw-r--r-- 1 arati arati 1663 Feb 27 13:31 numbers.txt
-rw-r--r-- 1 arati arati   22 Feb 27 13:44 output.txt
```

- cat file1.txt file2.txt | sort | uniq
    - Concatenate both files file1.txt and file2.txt then sort it and print only uniq data
    - Output

```
arati@Arati:~/LinuxAssignment$ cat file1.txt file2.txt |
 sort | uniq
Apple
Apricot
Avocado
Banana
Banana
Blueberry
Cherry
Grapefruit
Hello, EveryOne Cdac Students
Kiwi
Mango
Orange
Papaya
Pineapple
Strawberry
Watermelon
```

- ls -l | grep "^d"
  - In that list all directories with details
  - Output

```
arati@Arati:~/LinuxAssignment$ ls -l | grep "^d"
drwxr-xr-x 2 arati arati 4096 Feb 28 13:38 docs
drwxr-xr-x 2 arati arati 4096 Feb 28 13:53 mydir
```

- grep -r "pattern" /path/to/directory/
  - print the all kiwi/pattern form the file1.txt no matter how many time they show
  - Output

```
arati@Arati:~/LinuxAssignment$ grep -r "Kiwi" file1.txt
Kiwi
Kiwi
arati@Arati:~/LinuxAssignment$
```

- cat file1.txt file2.txt | sort | uniq –d
  - It print only duplicate data with sorting with concat file from file1.txt to file2.txt
  - Output

```
arati@Arati:~/LinuxAssignment$ cat file1.txt file2.txt |
 sort | uniq -d
Apple
Cherry
Kiwi
Strawberry
```

- chmod 644 file.txt
  - In that change permission of read, write, execute of file for owner, group and other user
  - Output

```
arati@Arati:~/LinuxAssignment$ ls -l file1.txt
-rwxr--r-- 1 arati arati 152 Feb 28 14:01 file1.txt
arati@Arati:~/LinuxAssignment$ chmod 644 file1.txt
arati@Arati:~/LinuxAssignment$ ls -l file1.txt
-rw-r--r-- 1 arati arati 152 Feb 28 14:01 file1.txt
```

- cp -r source_directory destination_directory
  - copy recursive file1.txt to file.txt
  - Output

```
arati@Arati:~/LinuxAssignment$ cp -r file1.txt file.txt
arati@Arati:~/LinuxAssignment$ cat file.txt
Banana
Banana
Apple
Apple
Strawberry
Strawberry
Kiwi
Cherry
Avocado
```

- find /path/to/search -name "*.txt"
  - find all directories end with .txt
  - Output

```
arati@Arati:~/LinuxAssignment$ find docs "*.txt"
docs
docs/duplicate.txt
docs/file.txt
docs/file2.txt
find: ‘*.txt’: No such file or directory
```

- chmod u+x file.txt
  - Add permission to owner for execute
  - Output

```
arati@Arati:~/LinuxAssignment$ ls -l file.txt
-rw-r--r-- 1 arati arati 152 Feb 28 15:28 file.txt
arati@Arati:~/LinuxAssignment$ chmod u+x file.txt
arati@Arati:~/LinuxAssignment$ ls -l file.txt
-rwxr--r-- 1 arati arati 152 Feb 28 15:28 file.txt
```

- echo $PATH
  - print path of system directories
  - Output

```
arati@Arati:~/LinuxAssignment$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c
/Program Files/WindowsApps/MicrosoftCorporationII.Window
sSubsystemForLinux_2.4.11.0_x64__8wekyb3d8bbwe:/mnt/c/Pr
ogram Files/Common Files/Oracle/Java/javapath:/mnt/c/Win
```

Part B

Identify True or False:

1. ls is used to list files and directories in a directory.

- Answer: true it is use for display the files and directories from directory

2. mv is used to move files and directories.

- Answer: true it is use to move directories or files

3. cd is used to copy files and directories.

- Answer: false it is use for change directories from directory

4. pwd stands for "print working directory" and displays the current directory.

- Answer: true it is use for display the path of current directory

5. grep is used to search for patterns in files.

- Answer: true it is use for search patterns from files

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

- Answer: true it is use for change the permission of owner, group and other user, It changes owners permission and gives all read write execute

permission, also change group and other user permission of read and execute.

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

- Answer: make directory1 if not available or if available then not create and create directory2 inside the directory1

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

- Answer: true, it is remove file.txt forcefully

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.

- Answer : it is incorrect command and correct command is **chmod**

2. cpy is used to copy files and directories.

- Answer : It is incorrect command and correct command is **cp**

3. mkfile is used to create a new file.

- Answer : It is incorrect command and correct command is **touch** for making new files and **mkdir** for making new directory

4. catx is used to concatenate files.

- Answer : It is incorrect command and correct command is **cat**

5. rn is used to rename files.

- Answer : It is incorrect command and correct command is **mv** move file data or file there is no such rename command

Part C

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

Answer:

arati@Arati:~/LinuxAssignment$ nano sh

```
echo "Hello, World!"
```

arati@Arati:~/LinuxAssignment$ bash sh

Hello, World!

arati@Arati:~/LinuxAssignment$

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

Answer:

arati@Arati:~/LinuxAssignment$ nano sh1

name="CDAC Mumbai"

echo $name

arati@Arati:~/LinuxAssignment$ bash sh1

CDAC Mumbai

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

Answer :

arati@Arati:~/LinuxAssignment$ nano sh2

echo $1

echo $2

arati@Arati:~/LinuxAssignment$ bash sh2 10 20

10

20

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

Answer:

- echo -n "Enter the first number : "
- read num1.
- echo -n "Enter the second number : "
- read num2.
- sum=`expr $num1 + $num2`
- echo "sum of two value is $sum"

Output:

arati@Arati:~/LinuxAssignment$ bash sh3

Enter the first number :
12
Enter the second number :
12
sum of two value is 24

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

Answer:

```
echo "Enter a number:"
read n
if [ `expr $n % 2` == 0 ]
then
    echo "$n is even"
else
    echo "$n is Odd"
fi
```

Output:

```
arati@Arati:~/LinuxAssignment$ bash sh4
Enter a number:
2
2 is even
Enter a number:
13
13 is Odd
```

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

Answer:

```
a=0
for a in {1..5};
do
echo $a
done
```

Output:

arati@Arati:~/LinuxAssignment$ bash sh5

1

2

3

4

5

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

Answer:

```
i=1
while [ $i -le 5 ]
do
   echo $i
   i=` expr $i + 1 `
done
```

Output:

arati@Arati:~/LinuxAssignment$ bash sh5

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".

Answer:

```
if test -f "file.txt" ;
then
echo "File is exist"
else
echo "File is not exist"
```

fi

Output:

arati@Arati:~/LinuxAssignment$ bash sh7

File is 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.

Answer:

```
echo Enter the value of x:
read x
if [ $x -gt 10 ]; then
  echo “$x is greater than 10”
else
  echo “$x is less than 10”
fi
```

Output:

Enter the value of x:

12

“12 is greater than 10”

arati@Arati:~/LinuxAssignment$ bash sh8

Enter the value of x:

5

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.

Answer:

```
i=1
for i in {1..5}
do
```

```
j=1
for j in {1..5}
do
res=` expr $j \* $i`
echo  -n "$j * $i = $res       "
done
echo
done
```

Output:

```
arati@Arati:~/LinuxAssignment$ bash sh9
1 * 1 = 1      2 * 1 = 2      3 * 1 = 3      4 * 1 = 4      5 * 1 = 5
1 * 2 = 2      2 * 2 = 4      3 * 2 = 6      4 * 2 = 8      5 * 2 = 10
1 * 3 = 3      2 * 3 = 6      3 * 3 = 9      4 * 3 = 12     5 * 3 = 15
1 * 4 = 4      2 * 4 = 8      3 * 4 = 12     4 * 4 = 16     5 * 4 = 20
1 * 5 = 5      2 * 5 = 10     3 * 5 = 15     4 * 5 = 20     5 * 5 = 25
arati@Arati:~/LinuxAssignment$
```

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.

Answer:

```
echo Enter the number
read i
while [ $i -ge 0 ];
do
 ((i++))
echo Enter the number
read i
if [ $i -le 0 ];
then
break;
else
res=` expr $i \* $i`
```

```
echo Square of $i is $res
fi
done
```

Output:

```
Enter the number
4
Enter the number
5
Square of 5 is 25
Enter the number
-1
```

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.

| p1 | p2 | p3 |
0 5 8 14

	Waiting Time
P1 | 0 | 5	0
P2 | 1 | 3	4
P3 | 2 | 6	6

Total = 10

Total Average Waiting Time = 10/3 = 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.

| p1 | p1 | p3 | p4 | p2 |
0 1 3 4 8 13

			W T	RT	TAT
P1	0	3	0	0	3
P2	1	5	7	8	12
P3	2	1	1	3	2
P4	3	4	1	4	5
			9	15	22

Total average turnaround time = 22/4 = 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.

| p1 | p2 | p4 | p1 | p3 |
0 1 5 7 12 19

	AT	BT	P	RT	WT
P1	0	6	3	0	6
P2	1	4	1	1	0
P3	2	7	4	12	10
P4	3	2	2	5	2

18

Total average waiting time = 18/4 = 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	2 = 2
P2	1	5	2 = 3
P3	2	2	2 =0
P4	3	3	2 =1

Calculate the average turnaround time using Round Robin scheduling.

Answer:

| p1 | p2 | p3 | p4 | p1 | p2 | p4 | p2 |
0 2 4 6 8 10 12 13 14

	RT	WT	CT	TAT
P1 | 0 | 4	0	6	10	10
P2 | 1 | 5	2	8	14	13
P3 | 2 | 2	4	2	6	4
P4 | 3 | 3	6	7	8	10

Total average turnaround time = 37/4 =9.25

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 :

- When fork () system call create it will create replication of that process that means their 1 child process and 1 main process. It means parent process value is 5 but after fork() it will increment by 1means value of x becomes 6
- Final value of x becomes 6.