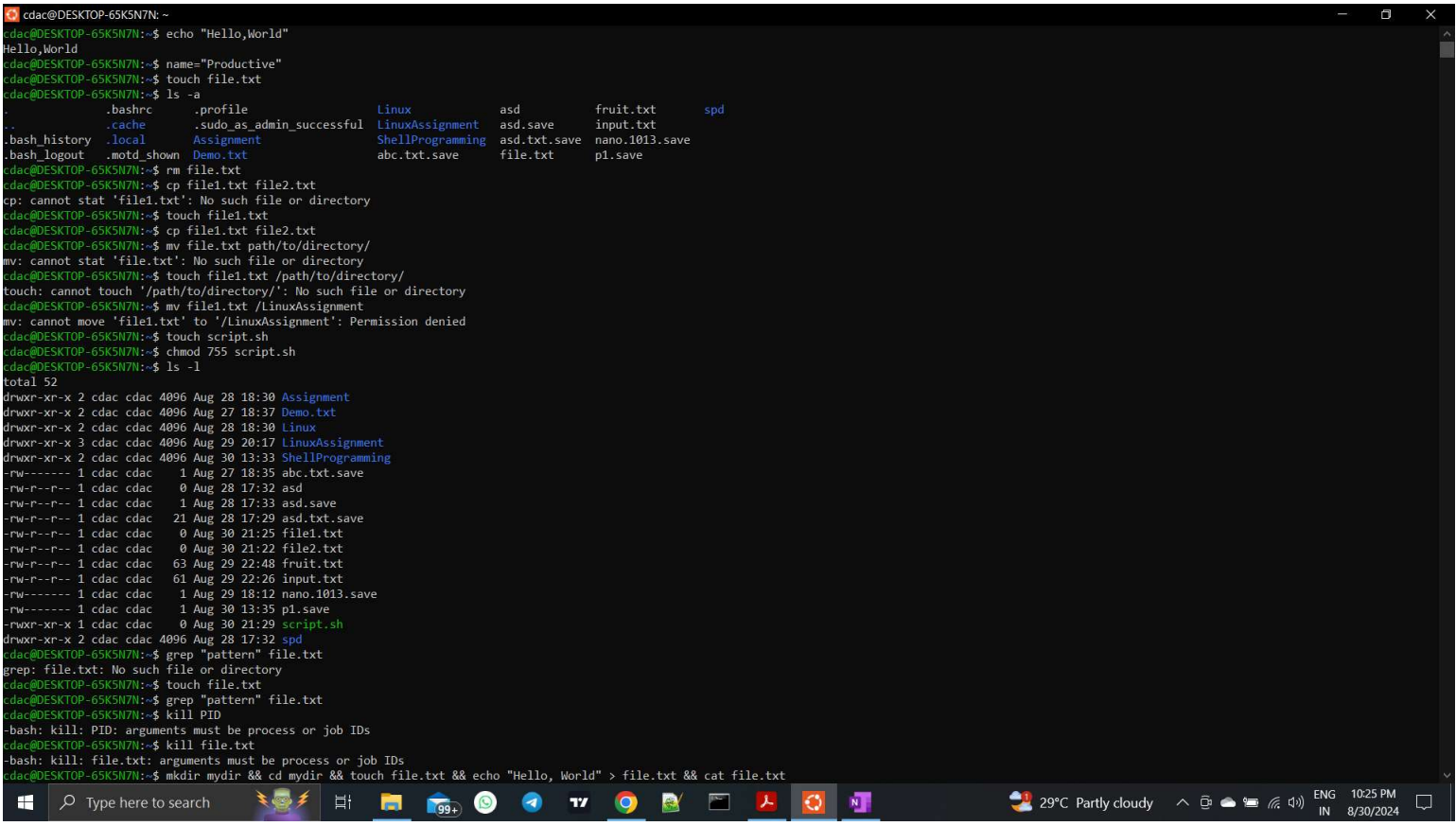
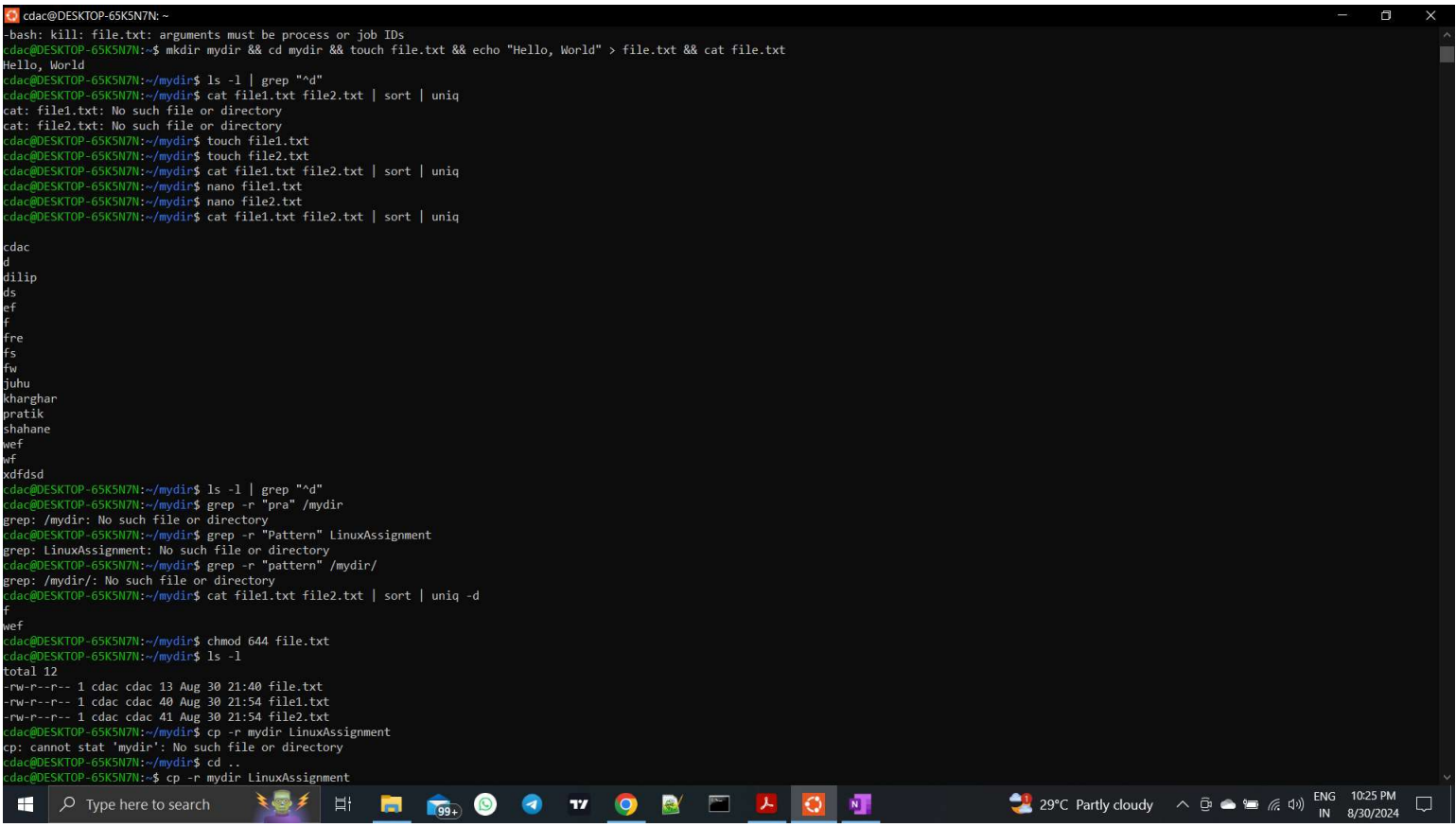
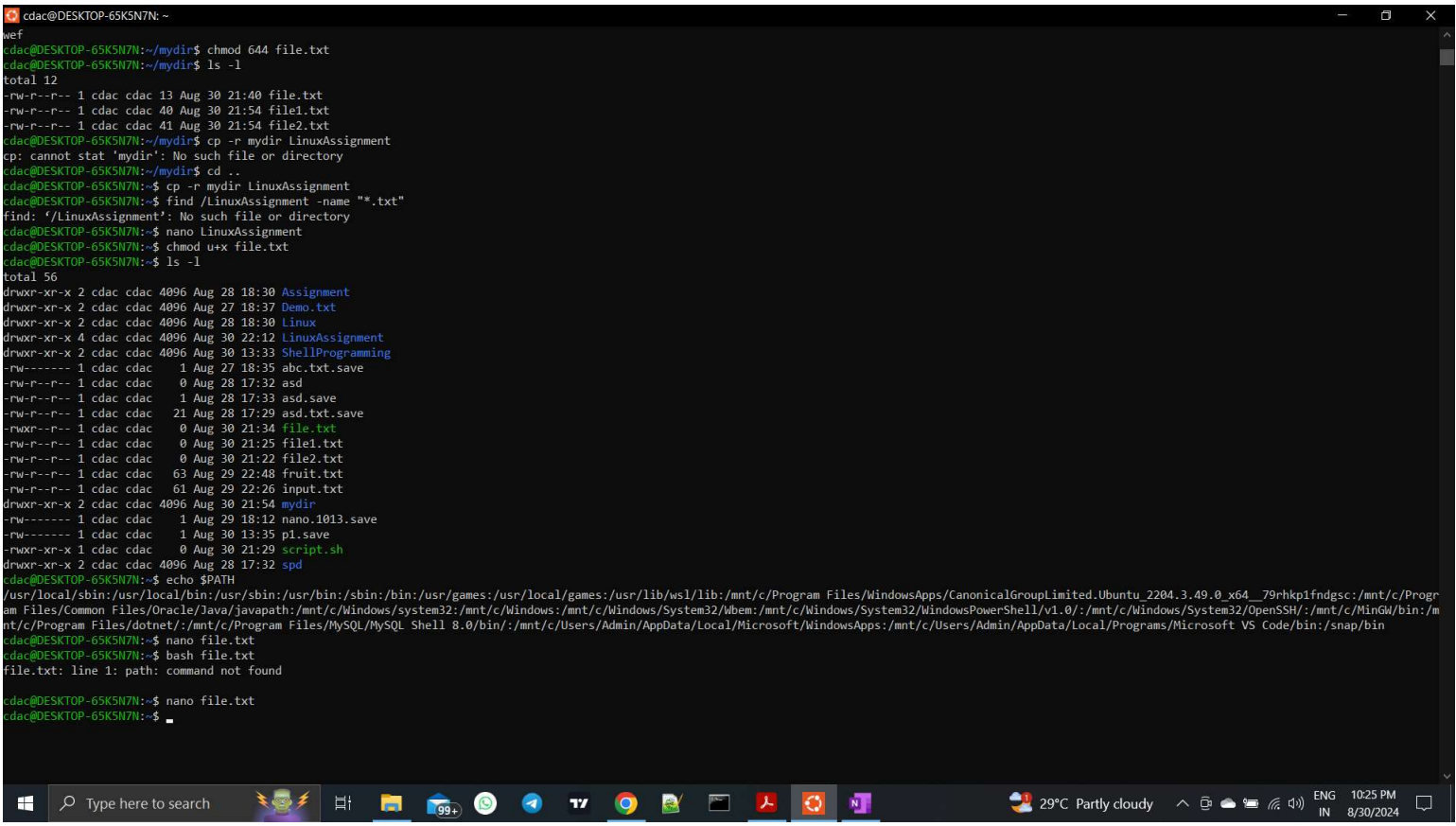
Linux Assignment 2 Solution

Friday, August 30, 2024 9:09 PM

Command	Explanation
echo "Hello, World!"	Print Hello,World
name="Productive"	assigns the value "Productive" to the environment variable named name.
touch file.txt	Create file named file.txt
ls -a	list files and directories in the current directory
rm file.txt	Remove file named file.txt
cp file1.txt file2.txt	Copied file1.txt to file2.txt
mv file.txt /path/to/directory/	Moves file.txt to /path/to/directory/.
chmod 755 script.sh	Sets the permissions of the file script.sh to 755 • **7**: represents the permissions for the file owner. (4+2+1 i.e. full permission i.e. read, write & execute) • **5**: represents the permissions for the group associated with the file. (only 4+1 i.e. read and execute) • **5**: represents the permissions for others (everyone else).
grep "pattern" file.txt	To search word pattern in file.txt
kill PID	TO kill a process from linux command line with its Process ID
mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt	Made new directory "mydir" && operator is used as it will operate only when previous commands are executed. Then we changed current working directory to mydir. Then we created file.txt, then we printed " Hello, World" and redirected it to file.txt. And then we displayed the content of the file.txt.
ls -l | grep ".txt"	We used pipe operator (|) to takes the output of the command on its left (All files & their information) and uses it as input for the command on its right (search for the word .txt)
cat file1.txt file2.txt | sort | uniq	First it will display the contents of file1.txt followed by the contents of file2.txt in sequence. Then pipe is used to **sort** (Sorts lines of text in ascending order). Then using pipe we used uniq to Remove any duplicate lines from the sorted list, leaving only unique lines.
ls -l | grep "^d"	We used pipe operator (|) to takes the output of the command on its left (All files & their information) and uses it as input for the command on its right (search for the word ^d)
grep -r "pattern" /path/to/directory/	Search for "pattern" in (-r represents recursively i.e. into all files of a specied directory & its subdirectories)
cat file1.txt file2.txt | sort | uniq –d	First it will display the contents of file1.txt followed by the contents of file2.txt in sequence. Then pipe is used & then we used **sort** (Sorts lines of text in ascending order). Then using pipe we used uniq to Remove any duplicate lines from the sorted list, leaving only unique lines. -d option tells uniq to only print lines that are duplicated (i.e., lines that appear more than once in the input).
chmod 644 file.txt	Sets the permissions of the file script.sh to 644 • **Owner (User)**: 6 i.e. 4+2(read and write) • **Group**: 4 (read only) • **Others**: 4 (read only)
cp -r source_directory destination_directory	Copied directory from source recursively to destinatin directory
find /path/to/search -name "*.txt"	To find the file / directory which has pattern like .txt
chmod u+x file.txt	User (u) will be granted permission to execute(+x) the file as a program or script of file.txt
echo $PATH	Print the assigned value of the variable path. $PATH is a environment variable that contains a colon-separated list of directories.







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. = False (chmod)
2. cpy is used to copy files and directories. = False (cp)
3. mkfile is used to create a new file. = False (touch)
4. catx is used to concatenate files. = False (cat)
5. rn is used to rename files. = False (mv)

Questions

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

```
echo Hello World
```

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

```
name ="CDAC Mumbai"
echo $name
```

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

```
A=2
echo $A
```

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

```
num1=5
num2=3
result=$((num1 + num2))
echo "The result of adding $num1 and $num2 is: $result"
```

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

```
a=10
if ((a%2==0))
then
echo number is even
else
echo number is odd
fi
```

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

```
a=0
for a in 1 2 3 4 5
do
echo $a
done
```

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

```
A=1
while [ $A -lt 5 ]
do
echo $A
((A++))
done
```

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".

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

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.

```
a=12
if (($a > 10))
then
echo Number is greater than 10
else
 echo Number is less than 10
fi
```

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.
#!/bin/bash

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.

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					P3 END
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Averrage Waiting time of P1, P2 & P3 = 10/3=3.3334

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			P3	P4				P2				P2 END	
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Avg TAT =3+12+2+5=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		P3						P3end		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Avg waiting time = 0+5+10+7 = 22/4 = 5.5

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.

P1	P2	P3	P4	P1	P2	P4 END AT 13	P2 END AT 15
0	2	4	6	8	10	12	14

Avg TAT = 10+14+4+10 = 38 /4 = 9.5

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 =

Before Forking Parent Process: x = 5 then **After Forking** Both Parent and Child processes have x = 5 (since the fork duplicates the parent's memory space). & incremening both by 1 will give final value of X in both parent & child process = 6