Name: Shankar Sitaram Parab

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- **echo "Hello, World!"**
  - The echo command used to display. In this case it prints "Hello, World!".

- **name="Productive"**
  - It used assign the string productive to variable name. in other word we can say that the variable name now store the string Productive.

- **touch file.txt**
  - It create a empty file with named file.txt if it doesn't already exist or updates the file timestamps if it exist.

- **ls -a**
  - The command ls -a listed all the file and directories in the current directory. Where option -a listed all the files and directory including hidden files.

- **rm file.txt**
  - In a Linux it removes the file.txt from the current directory.

- **cp file1.txt file2.txt**
  - The cp command copies the contents of file1.txt to file2.txt if file2.txt doesn't exist it will be created and if exist content will be copy in file1.txt

- **mv file.txt /path/to/directory/**
  - It moves the file file.txt to the specified directory /path/to/directory/. The file will no longer be in the current directory but will now reside in the target directory

- **chmod 755 script.sh**
  - chmod command change the permission of the file script.sh. Code 755 gives read, write and execute permissions to owner read and execute permissions to group and other users.

- **grep "pattern" file.txt**
  - grep searches for occurrences of the string "pattern" within the file file.txt and prints each line that contains the matching string to the terminal.

- **kill PID**
  - kill PID terminate the process with the specified process id.

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
  - This command performs sequence of action first mkdir create directory with name mydir. && (logical AND) operator is used here which enables the user to run multiple commands in single command. cd mydir change the current directory to mydir and touch creates empty file named file.txt in mydir directory and than echo "Hello, World!" > file.txt Writes the string Hello, World! into the file.txt file, overwriting any existing content. And finally cat file.txt Displays the content of file.txt, which will be Hello, World!.

- **ls -l | grep ".txt"**
  - ls -l command lists all the files and directories in the current directory in long format showing all details after that | pipe operator take the output of the ls-l command and passes it as input to grep command where grep ".txt" filters the output to only display line that contain string .txt, which means it show the shows the details of file with the .txt.

- **cat file1.txt file2.txt | sort | uniq**
  - The cat file1.txt file2.txt command concatenates the contents of file1.txt and file2.txt and give outputs. The | (pipe operator) takes the output of the cat command and passes it as input to the sort command. The sort command sorts the combined text in order. The | (pipe operator) then takes the sorted output and passes it as input to the uniq command, which removes any duplicate lines, resulting in a list of unique, sorted lines from both files.

- **ls -l | grep "^d"**
  - The ls -l command lists all files and directories in the current directory with all details. The | pipe operator takes the output of the ls -l command and passes it as input to the grep command. The grep "^d" command filters the output to only display lines that start with the letter d. This means it shows the details of directories in current directory.

- **grep -r "pattern" /path/to/directory/**
  - The grep -r "pattern" /path/to/directory/ command recursively searches for the string pattern within all files and directories starting from the specified path /path/to/directory/.

- **cat file1.txt file2.txt | sort | uniq –d**
  - The cat file1.txt file2.txt command concatenates the contents of file1.txt and file2.txt and outputs them as a single stream of text. The | pipe operator takes the output of the cat command and passes it as input to the sort command. The sort command sorts the combined text in order. The | pipe operator then takes

the sorted output and passes it as input to the uniq -d command. The uniq -d command filters and displays only the duplicate lines from the sorted text.

- **chmod 644 file.txt**
  - The chmod commands changes the permission of the file.txt files and the code 644 gives the read and write permission to owner and only read permission to the groups and other users.

- **cp -r source_directory destination_directory**
  - The cp -r source_directory destination_directory command recursively copies the contents of source_directory to destination_directory. The -r stands for recursive meaning it will copy the entire directory tree.

- **find /path/to/search -name "*.txt"**
  - The find /path/to/search -name "*.txt" command searches for files with the .txt extension within the specified path /path/to/search. It recursively looks through all subdirectories and displays the paths of all matching files.

- **chmod u+x file.txt**
  - The chmod u+x file.txt command adds execute permissions for the owner,of the file file.txt.

- **echo $PATH**
  - The echo $PATH command prints the value of the PATH environment variable. The PATH variable contains a list of directories

Name: Shankar Sitaram Parab

Part B

Identify True or False:

- **ls is used to list files and directories in a directory.**
  - True.
- **mv is used to move files and directories.**
  - True.
- **cd is used to copy files and directories.**
  - False, cd is used to change directories, not to copy files and directories.
- **pwd stands for "print working directory" and displays the current directory.**
  - True.
- **grep is used to search for patterns in files.**
  - True
- **chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.**
  - True
- **mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.**
  - True
- **rm -rf file.txt deletes a file forcefully without confirmation.**
  - True

Identify the Incorrect Commands:

1. **chmodx is used to change file permissions.**
   - chmod is used to change file permissions.

2. **cpy is used to copy files and directories.**
   - cp is used to copy files and directories.

3. **mkfile is used to create a new file.**
   - touch is used to create a new file.

4. **catx is used to concatenate files.**
   - cat is used to concatenate files.

5. **rn is used to rename files.**
   - mv is used to rename files.

Part C

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

```
cdac@shankar:~/LinuxAssignment$ nano helloworld.sh
cdac@shankar:~/LinuxAssignment$ cat helloworld.sh
echo "Hello, World!"
cdac@shankar:~/LinuxAssignment$ bash helloworld.sh
Hello, World!
```

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

```
cdac@shankar:~/LinuxAssignment$ nano name.sh
cdac@shankar:~/LinuxAssignment$ cat name.sh
name="CDAC Mumbai"
echo $name
cdac@shankar:~/LinuxAssignment$ bash name.sh
CDAC Mumbai
```

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

```
cdac@shankar:~/LinuxAssignment$ nano number.sh
cdac@shankar:~/LinuxAssignment$ cat number.sh
echo Enter a number:
read number
echo your number is $number
cdac@shankar:~/LinuxAssignment$ bash number.sh
Enter a number:
12
your number is 12
```

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

```
cdac@shankar:~/LinuxAssignment$ nano add.sh
cdac@shankar:~/LinuxAssignment$ cat add.sh
echo enter a number:
read num1
echo enter a number:
read num2

echo sum of $num1 and $num2 is $((num1+num2))
cdac@shankar:~/LinuxAssignment$ bash add.sh
enter a number:
5
enter a number:
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".

```
cdac@shankar:~/LinuxAssignment$ nano oddeven.sh
cdac@shankar:~/LinuxAssignment$ cat oddeven.sh
echo Enter a number:
read num

if [ $((num % 2)) -eq 0 ]
then
    echo Even
else
    echo Odd
fi
cdac@shankar:~/LinuxAssignment$ bash oddeven.sh
Enter a number:
5
Odd
```

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

```
cdac@shankar:~/LinuxAssignment$ nano for.sh
cdac@shankar:~/LinuxAssignment$ cat for.sh
for i in 1 2 3 4 5
do
  echo $i
done
cdac@shankar:~/LinuxAssignment$ bash for.sh
1
2
3
4
5
```

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

```
cdac@shankar:~/LinuxAssignment$ nano while.sh
cdac@shankar:~/LinuxAssignment$ cat while.sh
i=1
while [ $i -le 5 ]
do
  echo $i
  i=$((i + 1))
done
cdac@shankar:~/LinuxAssignment$ bash while.sh
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".

```
cdac@shankar:~/LinuxAssignment$ nano existsfile.sh
cdac@shankar:~/LinuxAssignment$ cat existsfile.sh
if [ -f "file.txt" ]
then
  echo "File exists"
else
  echo "File does not exist"
fi
cdac@shankar:~/LinuxAssignment$ bash existsfile.sh
File does not exist
cdac@shankar:~/LinuxAssignment$ touch file.txt
cdac@shankar:~/LinuxAssignment$ bash existsfile.sh
File exists
```

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@shankar:~/LinuxAssignment$ nano greater.sh
cdac@shankar:~/LinuxAssignment$ cat greater.sh
echo Enter a number
read number
if [ $number -gt 10 ]
then
  echo $number is greater than 10.
else
  echo $number is not greater than 10.
fi
cdac@shankar:~/LinuxAssignment$ bash greater.sh
Enter a number
15
15 is greater than 10.
cdac@shankar:~/LinuxAssignment$ bash greater.sh
Enter a number
8
8 is not 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@shankar:~/LinuxAssignment$ nano table.sh
cdac@shankar:~/LinuxAssignment$ cat table.sh
for i in {1..5}
do
    for j in {1..5}
    do
        res=`expr $i \* $j`
        echo -n "$res  "
        if [ $res -lt 10 ]
        then
            echo -n " "
        fi
    done
    echo
done

cdac@shankar:~/LinuxAssignment$ bash table.sh
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
```

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@shankar:~/LinuxAssignment$ nano negative.sh
cdac@shankar:~/LinuxAssignment$ cat negative.sh
while true
do
        echo Enter a number
        read number
        if [ $number -lt 0 ]
        then
                break
        fi
        square=$((number * number))
        echo The square of $number is $square.
done
echo Program terminated
cdac@shankar:~/LinuxAssignment$ bash negative.sh
Enter a number
5
The square of 5 is 25.
Enter a number
25
The square of 25 is 625.
Enter a number
6
The square of 6 is 36.
Enter a number
-1
Program terminated
```

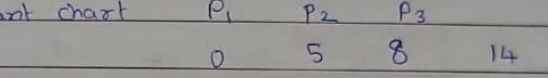
Name: Shankar Sitaram Parab

Part E

Q1 Consider the following processes with arrival times and burst times. Calculate the average waiting time using First come, First served (FCFS) sheduling.

Process	Arrival time	Burst time	waiting time
P_1	0	5	0
P_2	1	3	4
P_3	2	6	6

Gant chart



Avg waiting time = (0 + 4 + 6) / 3

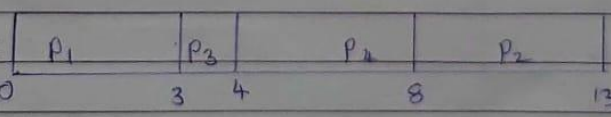
= 10 / 3

= 3.33

Q2 Consider the following Process with arrival time and burst times: Calculate the average turnaround time using Shortest Job First (SJF) sheduling

Process	Arrival time	Burst time	waiting time	Turnaround time
P_1	0	3	0	3
P_2	1	5	7	12
P_3	2	1	1	2
P_4	3	4	1	5

Gant chart:



Avg turnaround time = $\frac{3 + 12 + 2 + 5}{4}$

= 22 / 4

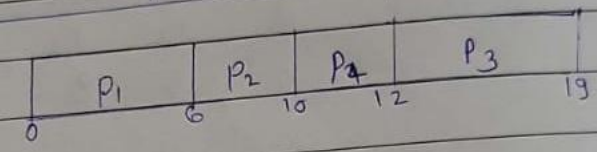
= 5.5

Name: Shankar Sitaram Parab

Q3 Consider the following process with arrival times, burst ti
and priorities (Lower number indicate higher priority)
Calculate the average waiting time using Priority sheduling

Process	Arrival time	Brust time	Priority	waiting time
P_1	0	6	3	0
P_2	1	4	1	5
P_3	2	7	4	10
P_4	3	2	2	7

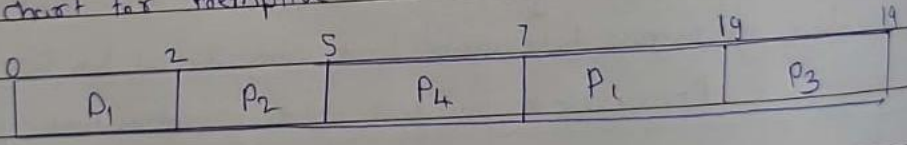
Gant chart



Avg waiting time = $\frac{22}{4}$

= 5.5

Gant chart for Premptive



Waiting time
6
0
2
10

Avg waiting time = $\frac{18}{4}$

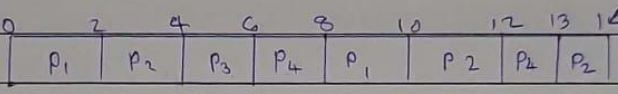
= 4.5

Name: Shankar Sitaram Parab

Q4 Consider the following process with arrival times and burst time and the time quantum for Round Robin sheduling is 2units:

Process	Arrival time	Burst time	Waiting time	Turn over time
P_1	0	4	6	10
P_2	1	5	8	13
P_3	2	2	2	4
P_4	3	3	7	10

Gant chart



Avg Turnaround time = (10 + 13 + 4 + 10) / 4

= 37 / 4

= 9.25

Q5. 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?

- Parent process starts with x = 5.
  fork() creates a child process with x = 5.
  Both parent and child increment x by 1.

  Final values:
  Parent process: x = 6
  Child process: x = 6

  So final values of x in the parent and child processes after the fork() call is **6**