CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  - echo is a shell command used to print text to the terminal.
  - "Hello, World!" is the string that will be printed.

```
cdac@PRATIK-DAREKAR:~$ echo "Hello Word!"
Hello Word!
cdac@PRATIK-DAREKAR:~$
```

- name="Productive"
  - This assigns the string "Productive" to a variable named name in a shell script or command line session.
  - The variable name can be used later in the script or session.

```
cdac@PRATIK-DAREKAR:~$ echo

cdac@PRATIK-DAREKAR:~$ name="Productive"
cdac@PRATIK-DAREKAR:~$ echo $name
Productive
cdac@PRATIK-DAREKAR:~$ name="Nisha"
cdac@PRATIK-DAREKAR:~$ echo $name
Nisha
cdac@PRATIK-DAREKAR:~$
```

- touch file.txt
  - The touch command is used to **create empty files** and **update timestamps** of existing files
  - File.txt is the name of the file that will be created or modified.

```
cdac@PRATIK-DAREKAR:~$ touch file.txt
cdac@PRATIK-DAREKAR:~$ ls
LinuxAssignment  file.txt
cdac@PRATIK-DAREKAR:~$ ls -1
LinuxAssignment
file.txt
cdac@PRATIK-DAREKAR:~$
```

- ls -a
  - The ls command is used to list files and directories.
  - The -a option stands for **"all"**, and it displays **hidden files**

```
cdac@PRATIK-DAREKAR:~$ ls -a
.   .bash_history  .bashrc  .landscape  .motd_shown  .sudo_as_admin_successful  file.txt
..  .bash_logout   .cache   .local      .profile     LinuxAssignment
cdac@PRATIK-DAREKAR:~$ ls
LinuxAssignment  file.txt
cdac@PRATIK-DAREKAR:~$ -a
-a: command not found
cdac@PRATIK-DAREKAR:~$
```

- rm file.txt
  - The rm remove command is used to **delete files and directories** in Linux.

```
cdac@PRATIK-DAREKAR:~$ touch file.txt
cdac@PRATIK-DAREKAR:~$ ls
LinuxAssignment  file.txt
cdac@PRATIK-DAREKAR:~$ rm file.txt
cdac@PRATIK-DAREKAR:~$ ls
LinuxAssignment
cdac@PRATIK-DAREKAR:~$
```

- cp file1.txt file2.txt
  - The cp (copy) command is used to **copy files and directories** in Linux.
  - In above example, the given command copies the contents of file1.txt,create a file named file2.txt and pastes the content in it.

```
This message is shown once a day. To disable it please create
/home/cdac/.hushlogin file.
cdac@PRATIK-DAREKAR:~$ echo "Hello, How are you"> file1.txt
cdac@PRATIK-DAREKAR:~$ ls
LinuxAssignment  file.txt  file1.txt
cdac@PRATIK-DAREKAR:~$ cp file1.txt file2.txt
cdac@PRATIK-DAREKAR:~$ ls -1
LinuxAssignment
file.txt
file1.txt
file2.txt
cdac@PRATIK-DAREKAR:~$ cat file2.txt
Hello, How are you
cdac@PRATIK-DAREKAR:~$
```

- mv file.txt /path/to/directory/
  - mv command is used rename or move a file.
  - In the above example, mv command moves the file (file.txt) into the specified directory (/path/to/directory/).

- chmod 755 script.sh
  - chmod stands Command to **change file permissions**.
  - 755 is Permission code.

User type	Permission	Explanation
Owner (User) -	7(rwx)	**Read (r)**, **Write (w)**, and **Execute (x)**
Group	5 (r-x)	**Read (r)** and **Execute (x)** (**No Write**)
Others (World)	5(r-x)	**Read (r)** and **Execute (x)** (**No Write**)

  - The above command gives read, write and execute permissions to the owner and read and execute permissions to group and other users respectively to script.sh file.

- grep "pattern" file.txt
  - grep → Searches for a pattern in a file.
  - "pattern" → The text or regex you want to find.
  - file.txt → The file where you want to search.

- kill PID
  - Terminates a process with the given **Process ID (PID)**.

- Since the above command doesn’t contain any process id, above command will result in an error

- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  - mkdir mydir is Create a directory named mydir.
  - cd mydir is Change into mydir.
  - touch file.txt is Create an empty file file.txt.
  - echo "Hello, World!" > file.txt is Write "Hello, World!" into file.txt.
  - cat file.txt is Display the contents of file.txt.
  - && (logical AND) operator is used here which enables the user to run multiple commands in single command.

- ls -1 | grep ".txt"
  - ls -1 is List files with detailed info.
  - grep ".txt" is Filter results to show only .txt files.

- cat file1.txt file2.txt | sort | uniq
  - cat file1.txt file2.txt is Display contents of both files.
  - sort is Sort the combined content.
  - uniq is Remove duplicate lines.

- ls -l | grep "^d"
  - ls -l is Lists files and directories with details.
  - grep "^d" is Filters only directories (d at the beginning)

- grep -r "pattern" /path/to/directory/
  - grep -r → Search inside all files in a directory (recursively).
  - "pattern" → The text to search for.
  - /path/to/directory/ → The directory to search in.

- cat file1.txt file2.txt | sort | uniq –d
  - uniq -d is Shows only **duplicate lines** in sorted input.
  - cat command displays the content of file1.txt followed by file2.txt. sort command is used to perform alphanumeric sort on the result of cat command. Contents of file1.txt and file2.txt are sorted separately in the result.

- chmod 644 file.txt
  - Owner: **Read & Write** (rw-).
  - Group & Others: **Read-only** (r--).
  - Used for **public readable files** where only the owner can edit.

- cp -r source_directory destination_directory
  - The above command is used to copy the source_directory to destination directory.
  - This is done by using -r option so that all files in source_directory are copied recursively.

- find /path/to/search -name "*.txt"
  - find command is used for searching the files and directories.

    - Given command searches /path/to/search directory and its subdirectories for any file ending with .txt pattern

- chmod u+x file.txt
    - This command is used to grant execute permissions for file.txt file to the user(owner) of the file.

- echo $PATH
    - Shows directories where executable programs are located

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**
  it is used to change the directory**.**
- **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. – **False**
  (recursive option) is used for deleting directories, not files.

Identify the Incorrect Commands:

- **chmodx** is used to change file permissions.
  - The correct command to change file permissions is **chmod** (short for "change mode").

- **cpy** is used to copy files and directories.
  - Use copy for files or xcopy / robocopy for directories.

- **mkfile** is used to create a new file.
  - The correct command to create a new file in Linux is **touch filename** or **echo "" > filename**.

- **catx** is used to concatenate files.
  - The correct command is **cat**, which is used to concatenate and display the contents of files.

- **rn** is used to rename files.
  - The correct commands are **mv** (Linux/macOS) and **ren** or **rename** (Windows).

Part C

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

```
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ nano hello.sh
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ cat hello.sh
echo "hello nisha"
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ bash hello.sh
hello nisha
cdac@PRATIK-DAREKAR:~/LinuxAssignment$
```

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

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

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

```
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ nano num.sh
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ cat num.sh
echo "Enter a number"
read a
echo your number is $a
cdac@PRATIK-DAREKAR:~/LinuxAssignment$ bash num.sh
Enter a number
4857
your number is 4857
cdac@PRATIK-DAREKAR:~/LinuxAssignment$
```

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

```
cdac@PRATIK-DAREKAR:~$ nano Add.sh
cdac@PRATIK-DAREKAR:~$ cat Add.sh
echo "Enter a number"
read a
echo "Enter b number"
read b
sum=`expr $a + $b`
echo sum of $a and $b is $sum
cdac@PRATIK-DAREKAR:~$ bash Add.sh
Enter a number
5
Enter b number
3
sum of 5 and 3 is 8
cdac@PRATIK-DAREKAR:~$
```

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

```
cdac@PRATIK-DAREKAR:~$ nano evenodd.sh
cdac@PRATIK-DAREKAR:~$ cat evenodd.sh
echo "Enter a number"
read a
if [ `expr $a % 2` -eq 0 ]
then
        echo "$a is an even number"
else
        echo "$a is an odd number"
fi

cdac@PRATIK-DAREKAR:~$ bash evenodd.sh
Enter a number
2
2 is an even number
cdac@PRATIK-DAREKAR:~$ 5
5: command not found
cdac@PRATIK-DAREKAR:~$ bash evenodd.sh
Enter a number
5
5 is an odd number
cdac@PRATIK-DAREKAR:~$
```

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

```
cdac@PRATIK-DAREKAR:~$ nano prime.sh
cdac@PRATIK-DAREKAR:~$ cat prime.sh
for i in 1 2 3 4 5 6 7 8 9
do
        echo $i
done

cdac@PRATIK-DAREKAR:~$ bash prime.sh
1
2
3
4
5
6
7
8
9
cdac@PRATIK-DAREKAR:~$
```

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

```
cdac@PRATIK-DAREKAR:~$ nano while.sh
cdac@PRATIK-DAREKAR:~$ cat while.sh
a=1
while [ $a -lt 6 ]
do
        echo $a
a=`expr $a + 1`
done

cdac@PRATIK-DAREKAR:~$ bash while.sh
1
2
3
4
5
cdac@PRATIK-DAREKAR:~$
```

Q 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@PRATIK-DAREKAR:~$ nano 8.sh
cdac@PRATIK-DAREKAR:~$ cat 8.sh
if [ -e file.txt ]
then
        echo "File exists"
else
        echo "File doesn't exist"
fi
cdac@PRATIK-DAREKAR:~$ bash 8.sh
File exists
cdac@PRATIK-DAREKAR:~$
```

Q 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@PRATIK-DAREKAR:~$ nano Q9.sh
cdac@PRATIK-DAREKAR:~$ cat Q9.sh
echo "Enter a number" ;
read a
if [ $a -gt 10 ]
then
        echo "$a is greater than 10"
else
        if [ $a -eq 10 ]
        then
                echo "$a is equal to 10"
        else
                echo "$a is smaller than 10"
        fi
fi
cdac@PRATIK-DAREKAR:~$ bash Q9.sh
Enter a number
58
58 is greater than 10
cdac@PRATIK-DAREKAR:~$ bash Q9.sh
Enter a number
81
81 is greater than 10
cdac@PRATIK-DAREKAR:~$ bash Q9.sh
Enter a number
10
10 is equal to 10
cdac@PRATIK-DAREKAR:~$
```

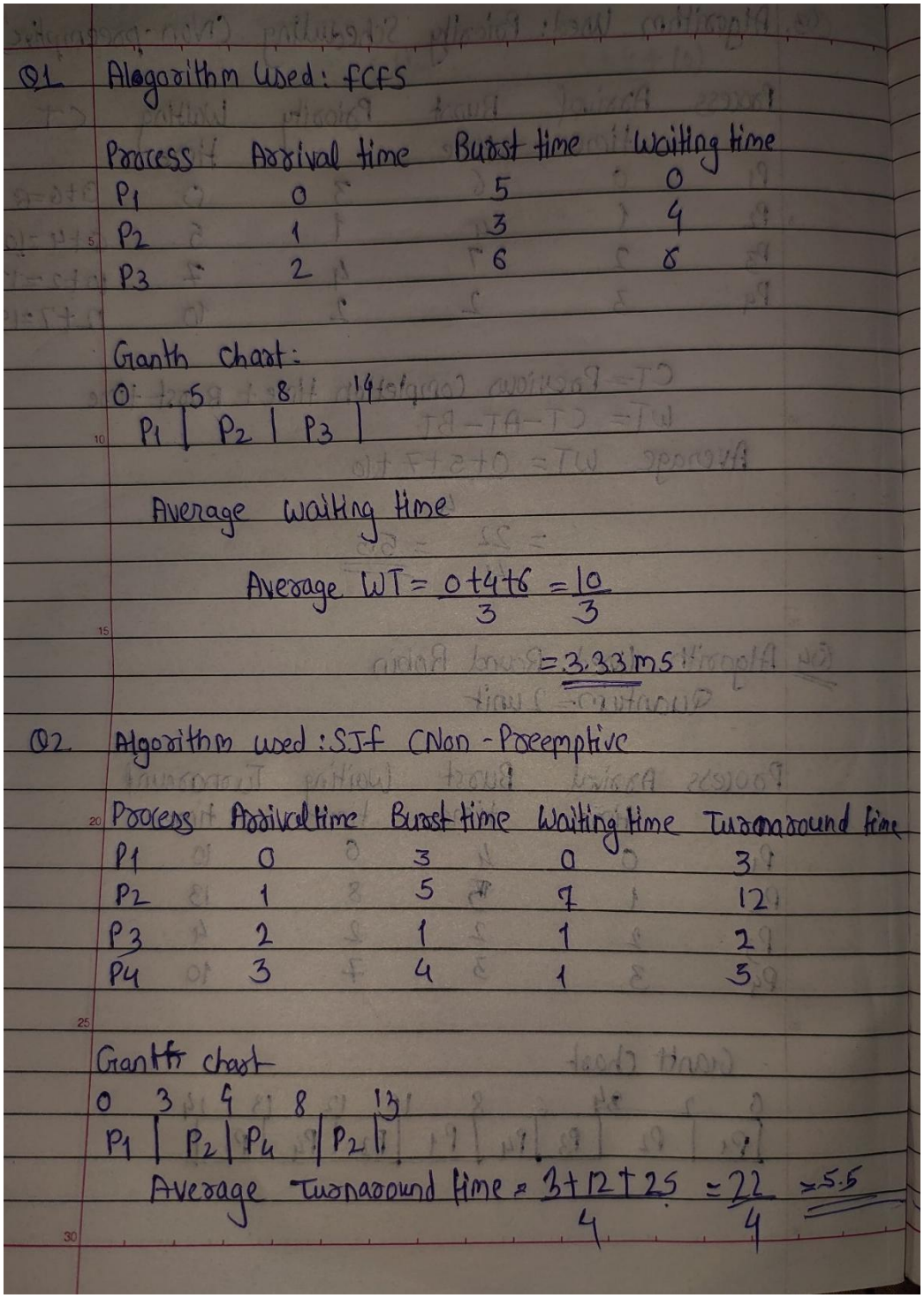
Q 10. Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number

```
cdac@PRATIK-DAREKAR:~$ nano multi.sh
cdac@PRATIK-DAREKAR:~$ cat multi.sh
for i in {1..5}
do
        for j in {1..5}
        do
                result=`expr $i \* $j`
                echo -n "$result   "
        done
        echo
done
cdac@PRATIK-DAREKAR:~$ bash multi.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
cdac@PRATIK-DAREKAR:~$
```

Q 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@PRATIK-DAREKAR:~$ nano 11.sh
cdac@PRATIK-DAREKAR:~$ cat 11.sh
while [ true ]
do
        echo "Enter a number";
        read a
        if [ $a -lt 0 ]
        then
                break
        fi
done
echo "Program Terminated"
cdac@PRATIK-DAREKAR:~$ bash 11.sh
Enter a number
5
Enter a number
4
Enter a number
3
Enter a number
2
Enter a number
1
Enter a number
0
Enter a number
-1
Program Terminated
cdac@PRATIK-DAREKAR:~$
```

Part – E



Q1 Algorithm used: FCFS

Process	Arrival time	Burst time	Waiting time
P_1	0	5	0
P_2	1	3	4
P_3	2	6	8

Gantt chart:

0 5 8 14

| P_1 | P_2 | P_3 |

Average waiting time

$$\text{Average WT} = \frac{0+4+6}{3} = \frac{10}{3}$$

$$= 3.33 \text{ ms}$$

Q2 Algorithm used: SJF (Non-Preemptive)

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

Gantt chart

0 3 4 8 13

| P_1 | P_2 | P_4 | P_2 |

Average Turnaround time $= \frac{3+12+25}{4} = \frac{22}{4} = 5.5$

Q3. Algorithm Used: Priority Scheduling (Non-preemptive)

Process	Arrival time	Burst time	Priority	Waiting time	CT
P_1	0	6	3	0	0+6=6
P_2	1	4	1	5	6+4=10
P_3	2	7	4	7	10+2=12
P_4	3	2	2	10	12+7=19

CT = Previous Completion time + Burst time

WT = CT − AT − BT

Average WT = $\frac{0+5+7+10}{4}$

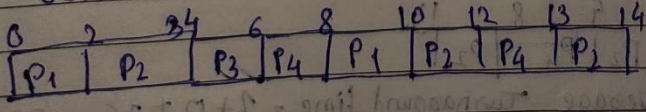
$= \frac{22}{4} = 5.5$

Q4 Algorith Used: Round Robin

Quantum = 2 unit

Process	Arrival time	Burst time	Waiting time	Turnaround 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

Gantt Chart



Avg. turnaround time = $\frac{(10+13+4+10)}{4}$

$= \frac{34}{4}$

$= 9.25$