Name: Shriram Narendra Sabade
Module: Concepts of OS

ASSIGNMENT 2 – PART A

What will the following command do?

- **echo “Hello World”**
  - This command will print the string present in the double quotes i.e. Hello World. Even if quotes aren’t provided echo command prints the string provided with it.

- **name=”Productive”**
  - This command will assign a string literal i.e. Productive to the shell variable named name.

- **touch file.txt**
  - touch command will create an empty file. In the above example, touch command will create a file named file.txt.

- **ls -a**
  - ls command lists the contents of a current directory. With -a option we can also list hidden files and directories.

- **rm file.txt**
  - rm command is used to delete a file or directory (-r option). In the above example, rm command deletes the file named file.txt.

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

- **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/). For this command to work these directories must be present in advance.

- **chmod 755 script.sh**
  - chmod stands for change modifications. This command is used to assign read, write, and execute permissions to owner, group and other users respectively. 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 command is used to search for specific patterns or regular expressions in text files & display the matching lines. Above given command, searches for the string “pattern” from the file named file.txt.

- **kill PID**
  - This command will terminate the process whose PID is mentioned in the command. 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**
  - && (logical AND) operator is used here which enables the user to run multiple commands in single command.
  - The above command produces a series of results where output of previous command acts as input for a next command. At first, mkdir command creates a mydir directory in the current directory.
  - cd command is then used to change current directory to new created mydir directory. Touch file.txt creates an empty file named file.txt. Further, echo command will display the message “Hello World” on the terminal. This output of echo command is inserted into file.txt using (>) redirect operator.
  - Finally, contents of file.txt are displayed using cat command.

- **ls -l | grep ".txt"**
  - The above command uses piping to combine the output of both ls and grep command. ls -l is used to display the contents of current directory with details and grep “.txt” command is used to display all the files conating .txt pattern in their name.

- **cat file1.txt file2.txt | sort | uniq**
  - The above command uses piping to combine the output of cat sort and uniq commands. First command i.e. cat command is used to display the contents of file1.txt followed by contents of 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.
  - Finally, uniq command is use to display only distinct lines in the result.

- **ls -l | grep "^d"**
  - ls command lists the files and directories in long format. grep "^d" command filters the output to show only lines that start with "d" which in the ls -l output indicates directories.

- **grep -r "pattern" /path/to/directory/**
  - Here grep command is used to recursively search for given pattern “pattern” in the directory /path/to/directory, provided that such directory exists in first place. The output will display the lines containing the “pattern” pattern in it.

- **cat file1.txt file2.txt | sort | uniq –d**
  - 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.
  - uniq -d command is used to display only duplicate lines in the previous output.

- **chmod 644 file.txt**
  - The above command assigns read and write permissions to owner of the file file.txt and read permission to group users and other users respectively.

- **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**
  - This command displays the value of system environment variable that stores directories where executable programs are located.

Name: Shriram Narendra Sabade
Module: Concepts of OS

ASSIGNMENT 2 – 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**, -r (recursive option) is used for deleting directories, not files.

Identify the Incorrect Commands:

- chmodx is used to change file permissions.
  - chmod command is used to change file permissions.

- cpy is used to copy files and directories.
  - cp command is used to copy files and directories.

- mkfile is used to create a new file.
  - touch command is used to create a new file. mkdir command is used to create a new directory.

- catx is used to concatenate files.
  - cat command is used to concatenate files.

- rn is used to rename files.
  - mv command is used to rename files when 2 files names are passed as arguments.

Name: Shriram Narendra Sabade
Module: Concepts of OS

ASSIGNMENT 2 – PART C

- **Q1. Write a shell script that prints "Hello, World!" to the terminal.**

```
cdac@SHRIRAM:~$ cd feb25
cdac@SHRIRAM:~/feb25$ nano hello.sh
cdac@SHRIRAM:~/feb25$ cat hello.sh
echo "Hello, World!"
cdac@SHRIRAM:~/feb25$ bash hello.sh
Hello, World!
cdac@SHRIRAM:~/feb25$
```

- **Q2. Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.**

```
cdac@SHRIRAM:~/feb25$ nano name.sh
cdac@SHRIRAM:~/feb25$ cat name.sh
name="CDAC Mumbai"
echo $name
cdac@SHRIRAM:~/feb25$ bash name.sh
CDAC Mumbai
```

- **Q3. Write a shell script that takes a number as input from the user and prints it.**

```
cdac@SHRIRAM:~/feb25$ nano Q3.sh
cdac@SHRIRAM:~/feb25$ cat Q3.sh
echo "Enter a number"
read a
echo Your number is $a
cdac@SHRIRAM:~/feb25$ bash Q3.sh
Enter a number
659
Your number is 659
```

- **Q4. Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.**

```
cdac@SHRIRAM:~/feb25$ nano Q4.sh
cdac@SHRIRAM:~/feb25$ cat Q4.sh
echo "Enter a number"
read a
echo "Enter a number"
read b
sum=`expr $a + $b`
echo sum of $a and $b is $sum
cdac@SHRIRAM:~/feb25$ bash Q4.sh
Enter a number
5
Enter a number
3
sum of 5 and 3 is 8
```

- **Q5. Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".**

```
cdac@SHRIRAM:~/feb25$ nano Q5.sh
cdac@SHRIRAM:~/feb25$ cat Q5.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@SHRIRAM:~/feb25$ bash Q5.sh
Enter a number
5
5 is an odd number
cdac@SHRIRAM:~/feb25$ bash Q5.sh
Enter a number
6
6 is an even number
```

- **Q6. Write a shell script that uses a for loop to print numbers from 1 to 5.**

```
cdac@SHRIRAM:~/feb25$ nano Q6.sh
cdac@SHRIRAM:~/feb25$ cat Q6.sh
for i in 1 2 3 4 5
do
        echo $i
done
cdac@SHRIRAM:~/feb25$ bash Q6.sh
1
2
3
4
5
```

- **Q7. Write a shell script that uses a while loop to print numbers from 1 to 5.**

```
cdac@SHRIRAM:~/feb25$ nano Q7.sh
cdac@SHRIRAM:~/feb25$ cat Q7.sh
a=1
while [ $a -lt 6 ]
do
        echo $a
a=`expr $a + 1`
done
cdac@SHRIRAM:~/feb25$ bash Q7.sh
1
2
3
4
5
```

- **Q8. 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@SHRIRAM:~/feb25$ nano Q8.sh
cdac@SHRIRAM:~/feb25$ cat Q8.sh
if [ -e file.txt ]
then
        echo "File exists"
else
        echo "File doesn't exist"
fi

cdac@SHRIRAM:~/feb25$ bash Q8.sh
File doesn't exist
cdac@SHRIRAM:~/feb25$ touch file.txt
cdac@SHRIRAM:~/feb25$ bash Q8.sh
File exists
```

- **Q9. Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.**

```
cdac@SHRIRAM:~/feb25$ nano Q9.sh
cdac@SHRIRAM:~/feb25$ 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 "$a is smaller than 10"
        fi
fi
cdac@SHRIRAM:~/feb25$ bash Q9.sh
Enter a number
56
56 is greater than 10
cdac@SHRIRAM:~/feb25$ bash Q9.sh
Enter a number
10
10 is equal to 10
cdac@SHRIRAM:~/feb25$ bash Q9.sh
Enter a number
5
5 is smaller than 10
```

- **Q10. 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@SHRIRAM:~/feb25$ nano Q10.sh
cdac@SHRIRAM:~/feb25$ cat Q10.sh
for i in  {1..5}
do
        for j in {1..5}
        do
                result=`expr $i \* $j`
                echo -n "$result          "
        done
        echo
done
cdac@SHRIRAM:~/feb25$ bash Q10.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
```

- **Q11. 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@SHRIRAM:~/feb25$ nano Q11.sh
cdac@SHRIRAM:~/feb25$ cat Q11.sh
while [ true ]
do
        echo "Enter a number" ; read a
        if [ $a -lt 0 ]
        then
                break
        fi
done
echo "Program Terminated"
cdac@SHRIRAM:~/feb25$ bash Q11.sh
Enter a number
5
Enter a number
4
Enter a number
3
Enter a number
0
Enter a number
-1
Program Terminated
```

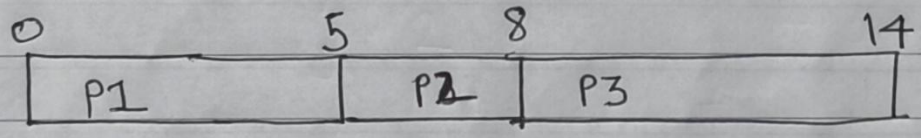
Name: Shriram Sabade
Module: Concepts of OS

Assignment – 2 Part – E

Q1] Algorithm used: FCFS

Process	Arrival Time	Burst time	Waiting Time
P1	0	5	0
P2	1	3	4
P3	2	6	6

Gantt chart:

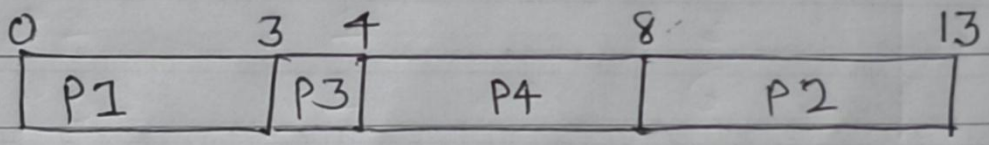


Avg. Waiting Time = (0+4+6)/3
= 10/3
= 3.333333
≈ 3.33

Q2] Algorithm Used : SJF (Non-Preemptive)

Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5

Gantt chart :



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

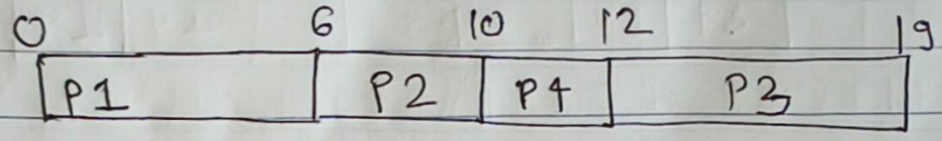
$= \frac{22}{4}$.

$= \underline{\underline{5.5}}$

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

Process	Arrival Time	Burst time	Priority	Waiting Time
P1	0	6	3	0
P2	1	4	1	5
P3	2	7	4	10
P4	3	2	2	7

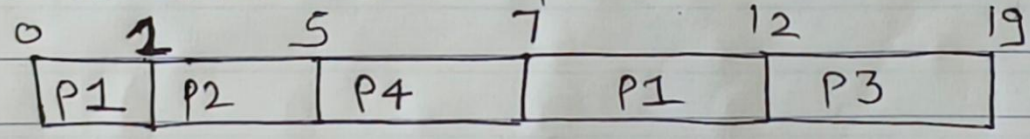
Gantt Chart:



Avg. Waiting Time = $\frac{22}{4}$

$= \underline{5.5}$

Gantt chart (Preemptive):



Waiting time
6
0
2
10

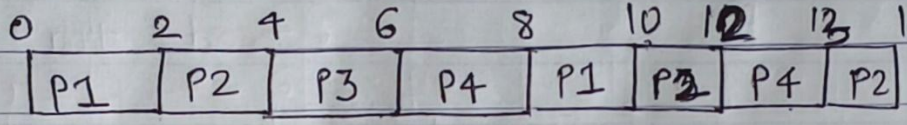
Avg. Waiting time = $\frac{18}{4}$

$= \underline{4.5}$

Q4] Algorithm Used : Round Robin
Quantum = 2 units

Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	4	6	10
P2	1	5	8	13
P3	2	2	2	4
P4	3	3	7	10

Gantt chart: (CPU ~~Idle~~ not kept idle)



Avg. Turnaround Time = (10+13+4+10)/4

= 37/4

= 9.25

- **Q5.**
  - When the fork() system call is used, it creates a child process that has its own copy of the parent's memory.
  - Before forking, the parent has a variable x = 5. After the fork, both the parent and child have separate copies of x, still equal to 5.
  - Each process then increments x by 1, so both the parent and child have x = 6, but in their own separate memory.
  - In parent process, x=6. In child process, x=6