CDAC MUMBAI

Concepts of Operating System

Assignment 2

Khushi Nikhare_KH

Part A

What will the following commands do?

- echo "Hello, World!" : It prints “Hello, World!”
- name="Productive" : It creates the variable “name” and assigns the “Productive” value to it.
- touch file.txt : It creates an empty file named file.txt
- ls -a : display the list(ls) of files in directory along with hidden files since -a(all) is included
- rm file.txt : remove(rm) the file named file.txt
- cp file1.txt file2.txt : copy(cp) the file1.txt(reference file) to file2.txt(target file)
- mv file.txt /path/to/directory/ : move(mv) the location of file.txt to specified path(/path/to/directory/)
- chmod 755 script.sh : change permission or mode(chmod) of script.sh file based on numbers (4:read / 3:write / 1:execute); in this case, 755: read, write, execute permission to owner and read and execute permission to groups and others.
- grep "pattern" file.txt : global regular expression print(grep) searches for text words/phrases (“pattern”) in file.txt
- kill PID : It kills the process with specifies Id(PID)
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt && ls -l | grep ".txt" : It creates(mkdir) the directory names mydir then change directory(cd) to recently made directory then creates an empty file named file.txt then prints “Hello, World!” to screen then redirects its output to file.txt then display the content of file.txt then display the list of files and directories in the system(la -l) and pipeline its output and displays only file that ends with “.txt”
- cat file1.txt file2.txt | sort | uniq : display the content of file1.txt and file2.txt but sort it and display only the unique(uniq) words in it.
- ls -l | grep "^d" : list all the directories after searching directories from the list of files and directories in it.
- grep -r "pattern" /path/to/directory/ : It searches the files in specifies path with text pattern “pattern” in it.
- cat file1.txt file2.txt | sort | uniq –d : display the repeated/duplicate(-d) content of file1.txt and file2.txt.
- chmod 644 file.txt : Change permissions of file.txt allowing owner to read and write, and only read permission to groups and others.
- cp -r source_directory destination_directory : copy the content of specified directory recursively(-r).
- find /path/to/search -name "*.txt" : find all the files whose name ends with “*.txt” in the specified directory path.

- chmod u+x file.txt : change permission by adding execute permission to user while letting groups and owner permission untouched of file.txt
- echo $PATH : print the path of specified file.

Part B

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 : cd is used for changing directory and cp is used to copy files and directories.
4. pwd stands for "print working directory" and displays the current directory. : FALSE : pwd stands for “present working directory” though it displays the current directory.
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. : chmod
2. cpy is used to copy files and directories. : cp
3. mkfile is used to create a new file. : mkdir is used to make directories and touch is use to create files
4. catx is used to concatenate files. : cat
5. rn is used to rename files. : mv command is used to rename and move files as well.

Part C

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

Cmd used:

echo “Hello, World!”

```
cdac@Khushi:~$ echo "Hello, World!"
Hello, World!
```

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

Cmd used:

name=”CDAC Mumbai”
echo $name

```
cdac@Khushi:~$ name="CDAC Mumbai"
cdac@Khushi:~$ echo $name
CDAC Mumbai
```

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

Cmd used:

touch input.txt
nano input.txt
bash input.txt

Shell script:

echo “Enter a number:”
read num
echo $num

```
cdac@Khushi:~$ touch input.txt
cdac@Khushi:~$ nano input.txt
cdac@Khushi:~$ cat input.txt
echo "Enter a number:"
read num
echo $num
cdac@Khushi:~$ bash input.txt
Enter a number:
4
4
```

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

Shell script:

```
sum=0
echo “Enter First Number:”
read num1
echo “Enter Second Number:”
read num2
sum=`expr $num1 + $num2`
echo Sum of $num1 + $num2 is equal to $sum
```

```
cdac@Khushi:~$ nano add.txt
cdac@Khushi:~$ cat add.txt
sum=0
echo "Enter First Number:"
read num1
echo "Enter Second Number:"
read num2
sum=`expr $num1 + $num2`
echo Sum of $num1 + $num2 is equal to $sum
cdac@Khushi:~$ bash add.txt
Enter First Number:
7
Enter Second Number:
8
Sum of 7 + 8 is equal to 15
```

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

Shell script:

```
echo "Enter a Number:"
read num
if [ `expr $num % 2` -eq 0 ]
then
        echo Even
else
        echo Odd
fi
```

```
cdac@Khushi:~$ nano evenOdd.txt
cdac@Khushi:~$ cat evenOdd.txt
echo "Enter a Number:"
read num
if [ `expr $num % 2` -eq 0 ]
then
        echo Even
else
        echo Odd
fi
cdac@Khushi:~$ bash evenOdd.txt
Enter a Number:
6
Even
cdac@Khushi:~$ bash evenOdd.txt
Enter a Number:
7
Odd
```

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

Shell script:

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

```
cdac@Khushi:~$ nano forloop.txt
cdac@Khushi:~$ bash forloop.txt
1
2
3
4
5
cdac@Khushi:~$ cat forloop.txt
a=1
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.

Shell script:

```
a=1
while [ `expr $a` -lt 6 ]
do
        echo $a
        a=`expr $a + 1`
done
```

```
cdac@Khushi:~$ nano whileLoop.txt
cdac@Khushi:~$ bash whileLoop.txt
1
2
3
4
5
cdac@Khushi:~$ cat whileLoop.txt
a=1
while [ `expr $a` -lt 6 ]
do
        echo $a
        a=`expr $a + 1`
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".

Shell script:

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

```
cdac@Khushi:~$ nano fileChk.txt
cdac@Khushi:~$ bash fileChk.txt
File does not exist
cdac@Khushi:~$ touch file.txt
cdac@Khushi:~$ ls
add.txt      file.txt     fileChk.txty  input.txt
evenOdd.txt  fileChk.txt  forloop.txt   whileLoop.txt
cdac@Khushi:~$ bash fileChk.txt
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.

Shell script:

```
echo "Enter a number:"
read num
if [ `expr $num` -gt 10 ]
```

```
then
        echo "$num is greater then 10"
else
        echo "$num is less then 10"
fi
```

```
cdac@Khushi:~$ nano greater.txt
cdac@Khushi:~$ bash greater.txt
Enter a number:
7
7 is less then 10
cdac@Khushi:~$ bash greater.txt
Enter a number:
13
13 is greater then 10
cdac@Khushi:~$ cat greater.txt
echo "Enter a number:"
read num
if [ `expr $num` -gt 10 ]
then
        echo "$num is greater then 10"
else
        echo "$num is less then 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.

Shell script:

```
for a in 1 2 3 4 5
do
    for b in 1 2 3 4 5 6 7 8 9 10
    do
        res=`expr $a \* $b`
        printf "%-4d" $res
    done
    echo
done
```

```
cdac@Khushi:~$ nano table.txt
cdac@Khushi:~$ bash table.txt
1   2   3   4   5   6   7   8   9   10
2   4   6   8   10  12  14  16  18  20
3   6   9   12  15  18  21  24  27  30
4   8   12  16  20  24  28  32  36  40
5   10  15  20  25  30  35  40  45  50
cdac@Khushi:~$ cat table.txt
for a in 1 2 3 4 5
do
        for b in 1 2 3 4 5 6 7 8 9 10
        do
                res=`expr $a \* $b`
                printf "%-4d" $res
        done
        echo
done
```

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.

Shell script:

```
while [ true ]
do
    echo "Enter a number:"
    read num
    if [ `expr $num` -lt 0 ]
    then
        break
    fi
    echo `expr $num \\* $num`
done
```

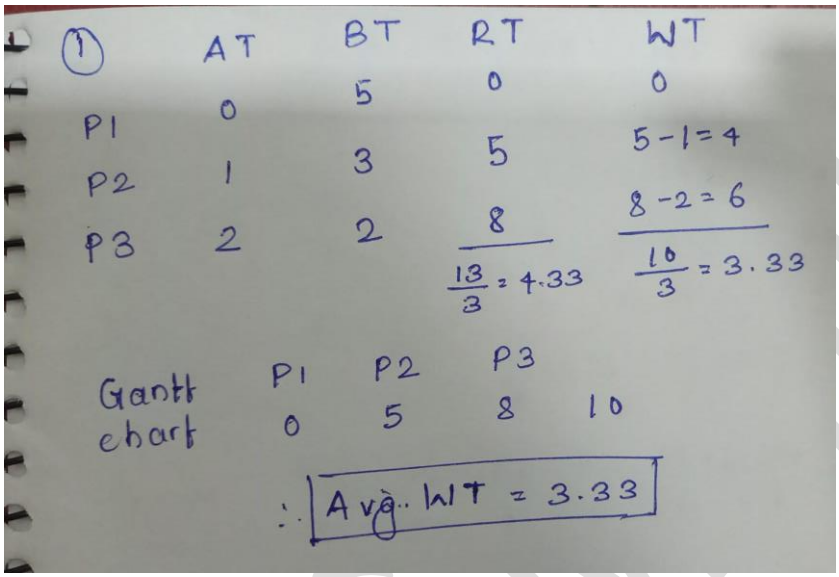
```
cdac@Khushi:~$ nano negative.txt
cdac@Khushi:~$ bash negative.txt
Enter a number:
6
36
Enter a number:
8
64
Enter a number:
-1
cdac@Khushi:~$ cat negative.txt
while [ true ]
do
        echo "Enter a number:"
        read num
        if [ `expr $num` -lt 0 ]
        then
                break
        fi
        echo `expr $num \\* $num`
done
```

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

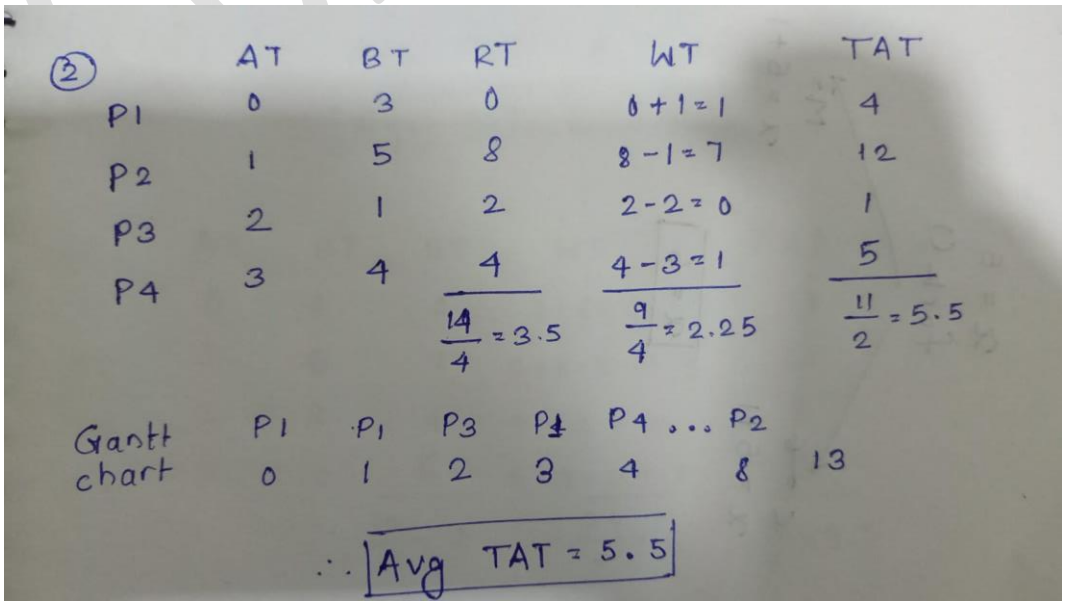
Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.



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

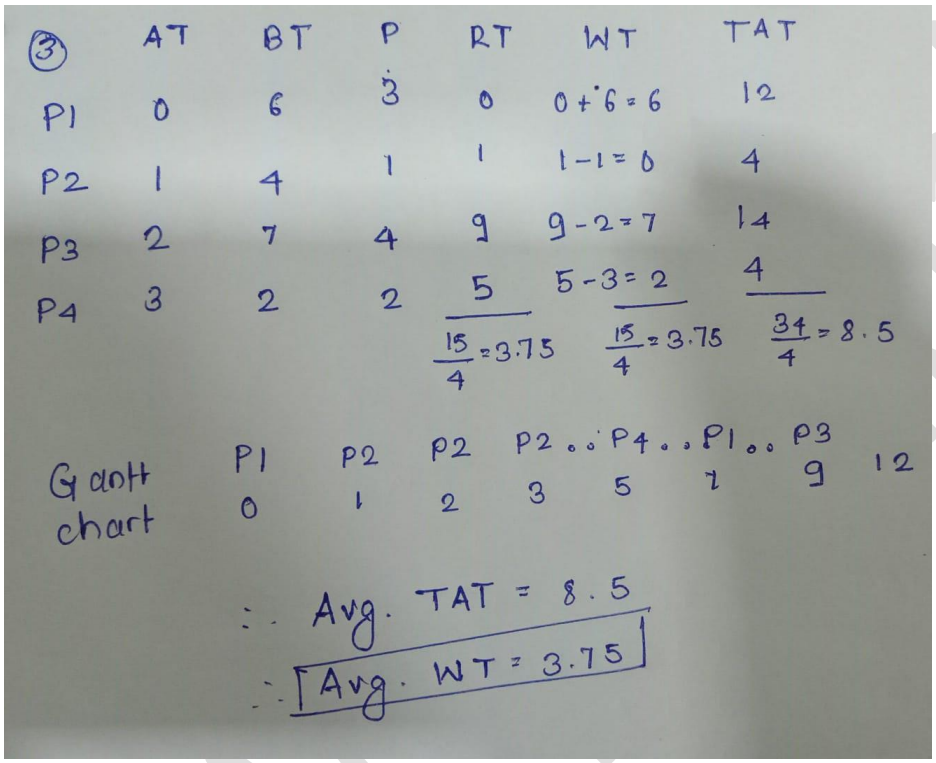
Calculate the average turnaround time using Shortest Job First (SJF) scheduling.



3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

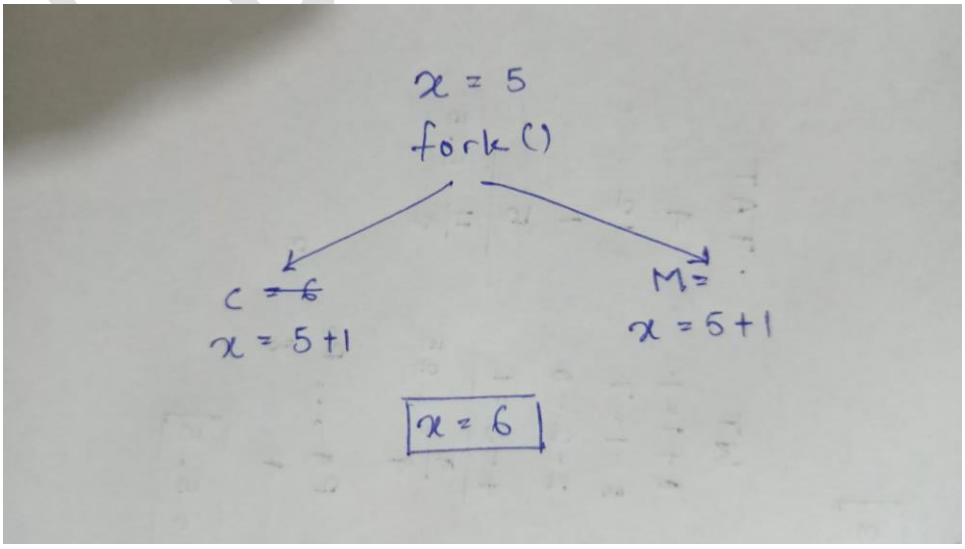
Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.



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?



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.