CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
- name="Productive"
- touch file.txt
- ls -a
- rm file.txt
- cp file1.txt file2.txt
- mv file.txt /path/to/directory/
- chmod 755 script.sh
- grep "pattern" file.txt
- kill PID
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
- ls -l | grep ".txt"
- cat file1.txt file2.txt | sort | uniq
- ls -l | grep "^d"
- grep -r "pattern" /path/to/directory/
- cat file1.txt file2.txt | sort | uniq –d
- chmod 644 file.txt
- cp -r source_directory destination_directory
- find /path/to/search -name "*.txt"
- chmod u+x file.txt
- echo $PATH

Part B

1)Answer:- to print the content

```
pwcdac@DESKTOP-E4S120V:~$ pwd
/home/cdac
cdac@DESKTOP-E4S120V:~$ echo "Hello,World!"
Hello,World!
cdac@DESKTOP-E4S120V:~$
```

2)assign the value “productive” to name

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
the name is productive
```

3) To create file and name file is touch

```
cdac@DESKTOP-E4S120V:~$ touch file.txt
cdac@DESKTOP-E4S120V:~$ ls
65]       duplicate.txt   file      file1.txt  number1.txt  output.txt  shellp1.sh  shp3.sh
data.txt  duplicate1.txt  file.txt  input.txt  numbers.txt  p4.sh       shp2.sh     sorted.txt
cdac@DESKTOP-E4S120V:~$
```

4)it gives hidden files and show the parent(..) and child(.) directory.

```
cdac@DESKTOP-E4S120V:~$ touch file.txt
cdac@DESKTOP-E4S120V:~$ ls
65]       duplicate.txt   file      file1.txt  number1.txt  output.txt  shellp1.sh  shp3.sh
data.txt  duplicate1.txt  file.txt  input.txt  numbers.txt  p4.sh       shp2.sh     sorted.txt
cdac@DESKTOP-E4S120V:~$
```

5)to remove file we use command rm

```
cdac@DESKTOP-E4S120V:~$ ls
65]       duplicate.txt   file      file1.txt  number1.txt  output.txt  shellp1.sh  shp3.sh
data.txt  duplicate1.txt  file.txt  input.txt  numbers.txt  p4.sh       shp2.sh     sorted.txt
cdac@DESKTOP-E4S120V:~$ rm file1.txt
cdac@DESKTOP-E4S120V:~$ ls
65]       duplicate.txt   file      input.txt    numbers.txt  p4.sh       shp2.sh  sorted.txt
data.txt  duplicate1.txt  file.txt  number1.txt  output.txt   shellp1.sh  shp3.sh
```

6)copy the file from file2 to file1

```
cdac@DESKTOP-E4S120V:~/mydir$ cp file.txt input.txt
cdac@DESKTOP-E4S120V:~/mydir$ ls
file.txt  input.txt
cdac@DESKTOP-E4S120V:~/mydir$
```

7)move file to the path of dirctory

```
cdac@DESKTOP-E4S120V:~$ mv data.txt file
cdac@DESKTOP-E4S120V:~$ ls
65]           duplicate1.txt  file.txt   number1.txt  output.txt  shellp1.sh  shp3.sh
duplicate.txt  file           input.txt  numbers.txt  p4.sh       shp2.sh     sorted.txt
```

8)the file can be executed by the owner, and it also sets appropriate read and execute permissions for others.

```
cdac@DESKTOP-E4S120V:~$ chmod 755 script.sh
cdac@DESKTOP-E4S120V:~$ ls -l
total 52
-rw-r--r-- 1 cdac cdac    0 Aug 31 14:04 65]
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:26 duplicate.txt
-rw-r--r-- 1 cdac cdac   76 Aug 31 04:38 duplicate1.tx
drwxr-xr-x 2 cdac cdac 4096 Aug 31 21:58 file
-rw-r--r-- 1 cdac cdac  382 Aug 31 19:46 file.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:10 input.txt
-rw-r--r-- 1 cdac cdac    0 Aug 31 03:54 number1.txt
-rw-r--r-- 1 cdac cdac   66 Aug 31 03:48 numbers.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:13 output.txt
-rw-r--r-- 1 cdac cdac  324 Aug 31 17:22 p4.sh
-rwxr-xr-x 1 cdac cdac    3 Aug 31 22:48 script.sh
```

9) find pattern word in mentioned file

```
cdac@DESKTOP-E4S120V:~$ nano file.txt
cdac@DESKTOP-E4S120V:~$ grep "pattern" file.txt
Birds are pattern
Sun shines pattern
Wind pattern quietly
Flowers pattern beautifully
pattern drift down
Leaves rustle pattern
Fire pattern warmly
cdac@DESKTOP-E4S120V:~$
```

10)it kill the process ID

11)print hello world!

```
cdac@DESKTOP-E4S120V:~$ mkdir mydir && cd mydir && touch file.txt && echo "Hello
,World!" > file.txt && cat file.txt
Hello,World!
```

12)it give all detailed file of ".txt"

```
cdac@DESKTOP-E4S120V:~$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:26 duplicate.txt
-rw-r--r-- 1 cdac cdac   76 Aug 31 04:38 duplicate1.txt
-rw-r--r-- 1 cdac cdac  384 Aug 31 22:52 file.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:10 input.txt
-rw-r--r-- 1 cdac cdac    0 Aug 31 03:54 number1.txt
-rw-r--r-- 1 cdac cdac   66 Aug 31 03:48 numbers.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:13 output.txt
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:23 sorted.txt
cdac@DESKTOP-E4S120V:~$
```

13)it give common content between file

```
cdac@DESKTOP-E4S120V:~$ cat file.txt number1.txt | sort | uniq
Birds are pattern
Clouds float above
Fire pattern warmly
Flowers pattern beautifully
Grass grows green
Leaves rustle pattern
Moon glows peacefully
Mountains rise high
Paths wind ahead
Rain falls gently
Rivers flow gently
Shadows stretch long
Sky is blue
Stars twinkle softly
Sun shines pattern
Time moves on
Trees stand tall
Waves crash loudly
Wind pattern quietly
pattern drift down
cdac@DESKTOP-E4S120V:~$
```

14)it give all detailed directories

```
cdac@DESKTOP-E4S120V:~$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Aug 31 21:58 file
cdac@DESKTOP-E4S120V:~$
```

15)it find file which has “pattern” content using path of file

```
cdac@DESKTOP-E4S120V:~$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:26 duplicate.txt
-rw-r--r-- 1 cdac cdac   76 Aug 31 04:38 duplicate1.txt
-rw-r--r-- 1 cdac cdac  384 Aug 31 22:52 file.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:10 input.txt
-rw-r--r-- 1 cdac cdac    0 Aug 31 03:54 number1.txt
-rw-r--r-- 1 cdac cdac   66 Aug 31 03:48 numbers.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:13 output.txt
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:23 sorted.txt
cdac@DESKTOP-E4S120V:~$
```

16)Displays only the lines that appear more than once (i.e., duplicates) in the sorted data.

```
cdac@DESKTOP-E4S120V:~$ cat file.txt number1.txt | sort | uniq -d
Birds are pattern
Clouds float above
Fire pattern warmly
Flowers pattern beautifully
Grass grows green
Leaves rustle pattern
```

17)it give permission to owner(read,execute),group(read),other(read)

```
cdac@DESKTOP-E4S120V:~$ chmod 644 file.txt
cdac@DESKTOP-E4S120V:~$ ls -l
total 56
-rw-r--r-- 1 cdac cdac    0 Aug 31 14:04 65]
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:26 duplicate.txt
-rw-r--r-- 1 cdac cdac   76 Aug 31 04:38 duplicate1.txt
drwxr-xr-x 2 cdac cdac 4096 Aug 31 21:58 file
-rw-r--r-- 1 cdac cdac  384 Aug 31 22:52 file.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:10 input.txt
-rw-r--r-- 1 cdac cdac  125 Aug 31 23:13 number1.txt
-rw-r--r-- 1 cdac cdac   66 Aug 31 03:48 numbers.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:13 output.txt
-rw-r--r-- 1 cdac cdac  324 Aug 31 17:22 p4.sh
-rwxr-xr-x 1 cdac cdac    3 Aug 31 22:48 script.sh
-rw-r--r-- 1 cdac cdac    9 Aug 31 13:48 shellp1.sh
-rw-r--r-- 1 cdac cdac   54 Aug 31 13:56 shp2.sh
-rw-r--r-- 1 cdac cdac  164 Aug 31 14:03 shp3.sh
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:23 sorted.txt
cdac@DESKTOP-E4S120V:~$
```

18)a)the entire directory, including all its contents, from a source location to a destination location

b) -r : This option stands for "recursive." It tells `cp` to copy directories recursively, meaning that it will copy the directory and all its contents, including subdirectories and files, into the destination directory

```
cdac@DESKTOP-E4S120V:~$ cp -r file.txt file
cdac@DESKTOP-E4S120V:~$ ls
65]            duplicate1.txt  file.txt   numbe
duplicate.txt  file            input.txt  numbe
cdac@DESKTOP-E4S120V:~$ cd file
cdac@DESKTOP-E4S120V:~/file$ ls
data.txt  file.txt
cdac@DESKTOP-E4S120V:~/file$
```

19) is used to search for files with a specific name pattern within a directory and its subdirectories.

```
cdac@DESKTOP-E4S120V:~$ find /home/cdac/file -name "*.txt"
/home/cdac/file/data.txt
/home/cdac/file/file.txt
```

20)change permission owner(read).

```
cdac@DESKTOP-E4S120V:~$ chmod u+x file.txt
cdac@DESKTOP-E4S120V:~$ ls -l
total 56
-rw-r--r-- 1 cdac cdac    0 Aug 31 14:04 65]
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:26 duplicate.txt
-rw-r--r-- 1 cdac cdac   76 Aug 31 04:38 duplicate1.txt
drwxr-xr-x 2 cdac cdac 4096 Aug 31 23:22 file
-rwxr--r-- 1 cdac cdac  384 Aug 31 22:52 file.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:10 input.txt
-rw-r--r-- 1 cdac cdac  125 Aug 31 23:13 number1.txt
-rw-r--r-- 1 cdac cdac   66 Aug 31 03:48 numbers.txt
-rw-r--r-- 1 cdac cdac  172 Aug 31 04:13 output.txt
-rw-r--r-- 1 cdac cdac  324 Aug 31 17:22 p4.sh
-rwxr-xr-x 1 cdac cdac    3 Aug 31 22:48 script.sh
-rw-r--r-- 1 cdac cdac    9 Aug 31 13:48 shellp1.sh
-rw-r--r-- 1 cdac cdac   54 Aug 31 13:56 shp2.sh
-rw-r--r-- 1 cdac cdac  164 Aug 31 14:03 shp3.sh
-rw-r--r-- 1 cdac cdac   40 Aug 31 04:23 sorted.txt
cdac@DESKTOP-E4S120V:~$
```

21)

```
cdac@DESKTOP-E4S120V:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr
/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/
javapath:/mnt/c/Python312/Scripts/:/mnt/c/Python312/:/mnt/c/WINDOWS/system32
:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/Windows
PowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files/nodej
s/:/mnt/c/ProgramData/chocolatey/bin:/mnt/c/Program Files/Git/cmd:/mnt/c/Pro
gram Files/MySQL/MySQL Server 8.0/bin:/mnt/c/Program Files/MySQL/MySQL Shell
 8.0/bin/:/mnt/c/Users/Admin/AppData/Local/Microsoft/WindowsApps:/mnt/c/User
s/Admin/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Users/Admin/AppD
ata/Roaming/npm:/mnt/c/Program Files/JetBrains/IntelliJ IDEA Community Editi
on 2024.1.3/bin:/snap/bin
```

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.-**false**

Identify the Incorrect Commands:

All are 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. -touch
4. **catx** is used to concatenate files. -cat
5. **rn** is used to rename files. -mv

Part C

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

```
cdac@DESKTOP-E4S120V:~/mydir$ nano shp3.sh
cdac@DESKTOP-E4S120V:~/mydir$ bash shp3.sh
Hello, World!
cdac@DESKTOP-E4S120V:~/mydir$
```

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

```
cdac@DESKTOP-E4S120V:~/mydir$ nano shp3.sh
cdac@DESKTOP-E4S120V:~/mydir$ bash shp3.sh
Enter Name
abc
Hello ,abc
cdac@DESKTOP-E4S120V:~/mydir$
```

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

```
cdac@DESKTOP-E4S120V:~/mydir$ nano shp3.sh
cdac@DESKTOP-E4S120V:~/mydir$ bash shp3.sh
Enter number
25
Hello ,25
cdac@DESKTOP-E4S120V:~/mydir$
```

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

```
#!bin/bash
echo enter number1
read num1
echo enter numbe2
read num2
sum=$((num1+num2))
echo sum is $sum
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
enter number1
20
enter numbe2
10
sum is 30
```

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

```
#!bin/bash
echo enter number
read num1
if [ $((num1%2)) -eq 0 ]
then
echo Even number
else
echo Odd Number
fi
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
enter number
20
Even number
```

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

```
#!bin/bash
a = 0
for a in 1 2 3 4 5 6
do
echo $a
done
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
shp3.sh: line 2: a: command not found
1
2
3
4
5
6
```

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

```
#!bin/bash
a=0
while [ $a -le 10 ]
do
echo $a
a=$((a + 1))
done
```

e

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
0
1
2
3
4
5
6
7
8
9
10
```

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".

```
#!bin/bash

if [ -f "file.txt" ]
then
echo "File exit"
else
echo"File not exist"
fi
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
File exit
```

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.

```
#!bin/bash

echo Enter number
read num

if [ $num -gt 10 ]
then
echo "the number is greater 10"
else
echo"the numbe is not grater 10."
fi
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
Enter number
20
the number is greater 10
cdac@DESKTOP-E4S120V:~$
```

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

a=1
b=1
for a in 1 2 3 4 5
do
for b in 1 2 3 4 5
do
mul=$(( a * b ))
printf $mul" "
done
echo
done
```

```
cdac@DESKTOP-E4S120V:~$ bash shp3.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.

```
#!bin/bash

while true
do
echo Enter number
read num
if [ $num -lt 0 ]
then
echo "Negative number not allows"
break
else
sqa=$((num * num))
echo the square is $sqa
fi
done
```

```
cdac@DESKTOP-E4S120V:~$ nano shp3.sh
cdac@DESKTOP-E4S120V:~$ bash shp3.sh
Enter number
4
the square is 16
Enter number
-5
Negative number not allows
cdac@DESKTOP-E4S120V:~$
```

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?

6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?
10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?
16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.
26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.

40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

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.
Average Waiting Time:- 10/3=3.3333..

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.
Average Turnaround Time:-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.
Average Waiting Time:-12/4=3

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.
Average Turnaround Time:-39/4=9.75

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?

Ans:-after the fork() system call, both the parent and child processes will have x equal to 6,

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.