Assignment 2

CDAC MUMBAI

Part A

What will the following commands do?

- echo "Hello, World!"
  - Displays the text Hello, World! on the terminal.
- name="Productive"
  - Assigned “Productive” to the variable
- touch file.txt
  - Creates an empty file named file.txt
- ls -a
  - Lists all files and directories in the current directory.
- rm file.txt
  - Deletes the file named file.txt
- cp file1.txt file2.txt
  - Copies the contents of file1.txt to file2.txt. If file2.txt
- mv file.txt /path/to/directory/
  - Moves the file file.txt to some directory
- chmod 755 script.sh
  - Changes the permissions of script.sh to 755
- grep "pattern" file.txt
  - Search for lines containing the string pattern in file.txt and prints those lines to the terminal.
- kill PID
  - Sends a signal to terminate the process with the specified process ID
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt ls -l | grep ".txt"
  - `mkdir mydir`: Creates the directory `mydir`.
  - `cd mydir`: Changes to the newly created directory.
  - `touch file.txt`: Creates an empty file named `file.txt`.
  - `echo "Hello, World!" > file.txt`: Writes "Hello, World!" to `file.txt`.
  - `cat file.txt`: Displays the contents of `file.txt` ("Hello, World!").
  - `ls -l | grep ".txt"`: Lists detailed information about files in the directory and filters for `.txt` files.
- cat file1.txt file2.txt | sort | uniq
  - `cat file1.txt file2.txt`:
    - Concatenates the contents of `file1.txt` and `file2.txt` and sends the combined output to the standard output (usually the terminal).
  - `| sort`:
    - Pipes (`|`) the combined output from `cat` into the `sort` command. `sort` arranges the lines of the input in alphabetical order (or numerical order if the input consists of numbers).
  - `| uniq`:
    - Pipes the sorted output into `uniq`. The `uniq` command filters out adjacent duplicate lines, so only unique lines remain. It relies on the fact that the input is already sorted to correctly identify duplicates.
- ls -l | grep "^d"
  - Lists files in the current directory in long format and filters the results to show only directories.
  - | grep is used to pipe out the output and print only those who start with g
- grep -r "pattern" /path/to/directory/
  - Recursively searches for the string pattern in all files within the specified directory (/path/to/directory/) and its subdirectories
- cat file1.txt file2.txt | sort | uniq –d
  - Concatenates the contents of file1.txt and file2.txt
- chmod 644 file.txt
  - Changes the permissions of file.txt to 644.

- cp -r source_directory destination_directory
  - Recursively copies the contents of source_directory to destination_directory. This includes all subdirectories and files.
- find /path/to/search -name "*.txt"
  - Searches for files with the .txt extension within the specified directory
- chmod u+x file.txt
  - Adds execute permission for the user (owner) of file.txt
- echo $PATH
  - Displays the current value of the PATH environment variable,

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

4. pwd stands for "print working directory" and 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.⇒ 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.

```
aditimehre@Aditis-MacBook-Air ShellHW % ls
hello_world
aditimehre@Aditis-MacBook-Air ShellHW % nano hello_world
aditimehre@Aditis-MacBook-Air ShellHW % ./hello_world.sh
zsh: no such file or directory: ./hello_world.sh
aditimehre@Aditis-MacBook-Air ShellHW % ./hello_world
zsh: permission denied: ./hello_world
aditimehre@Aditis-MacBook-Air ShellHW % chmod +x hello_world
aditimehre@Aditis-MacBook-Air ShellHW % ./hello_world
Hello, World!
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
  UW PICO 5.09

#!/bin/bash
echo "Hello, World!"
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % nano hello_world
aditimehre@Aditis-MacBook-Air ShellHW % ./hello_world
CDAC-Mumbai
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash
 name="CDAC-Mumbai"
echo $name
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % ./hello_world
enter the number
24
You entered the number 24
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash
echo "enter the number"
read number
echo "You entered the number $number"
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter the first number
3
Enter the second number
5
The sum is : 8
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash

echo "Enter the first number"
read num1

echo "Enter the second number"
read num2

sum=$((num1+num2))

echo "The sum is : $sum"
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter the first number
4
Even
aditimehre@Aditis-MacBook-Air ShellHW % nano add_numbers.sh
aditimehre@Aditis-MacBook-Air ShellHW % nano add_numbers.sh
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter the first number
6
Even
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter the first number
0
Even
```

```
#!/bin/bash

echo "Enter the first number"
read num1

if((num1%2==0));
 then
echo "Even"
else
echo"odd"
fi
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
1
2
3
4
5
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash

for i in {1..5}
do
echo $i
done
```

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

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
0
1
2
3
4
5
6
7
8
9
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash
a=0
while [ $a -lt 10 ]
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".

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
./add_numbers.sh: line 2: -e: command not found
File not present
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash
if(-e "file.txt"); then
echo "File present"
else
echo "File not present"
fi
```

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.

```
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter a number
78
Number is greater than 10
aditimehre@Aditis-MacBook-Air ShellHW %
```

```
#!/bin/bash
echo "Enter a number"
read num1

if [ $num1 -gt 10 ]; then
echo "Number is greater than 10 "
else
echo "Number is less than 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.

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.

```
Enter a number (negative number to quit):
-1
Negative number entered. Exiting.
aditimehre@Aditis-MacBook-Air ShellHW % ./add_numbers.sh
Enter a number (negative number to quit):
6
The square of 6 is 36
Enter a number (negative number to quit):
```

```
#!/bin/bash

# Start an infinite loop
while true; do
    # Prompt the user to enter a number
    echo "Enter a number (negative number to q
    read number

    # Check if the number is negative
    if [ "$number" -lt 0 ]; then
        echo "Negative number entered. Exiting
        break  # Exit the loop
    fi

    # Calculate the square of the number
    square=$((number * number))

    # Print the square
    echo "The square of $number is $square"
done
```

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.

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.

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.

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?

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.