CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  Prints the text "Hello, World!" to the terminal.
- name="Productive"
  Assigns the string "Productive" to a variable named '`name`'.
- touch file.txt
  Creates file.txt
- ls –a
  Lists all files and directories in the current directory, including hidden ones
- rm file.txt
  Deletes the file named `file.txt`.
- cp file1.txt file2.txt
  Copies the contents of file`1.txt` to `file2.txt`
- mv file.txt /path/to/directory/
  Moves `file.txt` to the specified directory (`/path/to/directory/`).
- grep "pattern" file.txt

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. : 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. : True

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions

   chmod is used

2. **cpy** is used to copy files and directories.
   cp is used.

3. **mkfile** is used to create a new file.
   Touch file

4. **catx** is used to concatenate files.
   cat is used.

5. **rn** is used to rename files.
   mv is used.

Part C

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

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

Ans)
```
#!/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.

Ans)
```
#!/bin/bash
echo "Enter a number:"
read number
echo "You entered: $number"
```

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

Ans)
```
#!/bin/bash
num1=5
num2=3
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
```

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

Ans)
```
#!/bin/bash
echo "Enter a number:"
read number
if [ $((number % 2)) -eq 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.
```
#!/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.
```
#!/bin/bash
i=1
while [ $i -le 5 ]; do
    echo $i
    i=$((i + 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".
```
#!/bin/bash
if [ -f "file.txt" ]; then
 echo "File exists"
else
  echo "File does not exist"
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.
```
#!/bin/bash
echo "Enter a number:"
read number
if [ $number -gt 10 ]; then
    echo "The number is greater than 10"
else
    echo "The number is not greater 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.
```
#!/bin/bash
for i in {1..5}; do
  for j in {1..5}; do
    printf "%4d" $((i * j))
  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.

```
#!/bin/bash
while true; do
   echo "Enter a number (negative number to exit):"
   read number

   if [ $number -lt 0 ]; then
      echo "Negative number entered. Exiting..."
      break
   fi

   square=$((number * number))
   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.

Average Waiting Time=30+4+6=310≈3.33
The average waiting time is approximately 3.33 units.

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=43+12+2+5=422=5.5
The average turnaround time is 5.5 units.

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=47+0+11+2=420=5
The average waiting time is 5 units.

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=410+13+4+10=437=9.25
The average turnaround time is 9.25 units.

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?

   The final value of x in both the parent and child processes is 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.