CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A complete

What will the following commands do?

- echo "Hello, World!" echo command is use to print content. Here, it is used to print Hello World.
- name="Productive" Assign the value "Productive" to the variable "name"
- touch file.txt Touch command is used to create an empty file in current directory. It creates a file named file.txt which is empty.
- ls -a this command is use to show all files and directories including hidden in the current directory
- rm file.txt rm command is use to remove that file
- cp file1.txt file2.txt cp command is use for copy. In this, the content of file1.txt will be copy into the content of file2.txt
- mv file.txt /path/to/directory/ mv command is used to move the file into directory
- chmod 755 script.sh set the permisiion of the file "script.sh" to read,write,and execute for the owner, and read and execute for the group and others.
- grep "pattern" file.txt grep command is use to find that word from the following given content.
- kill PID kill command is use to terminate process and PID is Process Id which is the unique identifier assigned to each running process in the system.
- 11 mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
- ls -l | grep ".txt" list files in long format . It shows read write, execute files.
- cat file1.txt file2.txt | sort | uniq concatenates the contents of "file1.txt" and "file2.txt", sorts the lines, and then removes duplicates.
- ls -l | grep "^d" This command lists files in long format and filters out only the lines representing directories.
- grep -r "pattern" /path/to/directory/ This command searches recursively for the specified pattern in all files under the directory
- cat file1.txt file2.txt | sort | uniq –d This command concatenates the contents of "file1.txt" and "file2.txt", sorts the lines, and then prints only the duplicated lines.
- chmod 644 file.txt This command sets the permissions of "file.txt" to read and write for the owner, and read-only for the group and others.
- cp -r source_directory destination_directory This command recursively copies the contents of "source_directory" to "destination_directory".
- find /path/to/search -name "*.txt" : This command searches for files with a ".txt" extension under the directory "/path/to/search".
- chmod u+x file.txt This command adds execute permission for the owner of "file.txt".
- echo $PATH This command prints the value of the environment variable PATH, which contains a list of directories where executable files are located.

solution 11:- This command sequence creates a directory named "mydir", changes into that directory, creates a new file named "file.txt", writes "Hello, World!" into "file.txt", and then displays the contents of "file.txt".

Part B complete

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 command is used to change the current directory work
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. Incorrect, chmod rwx........ use to change the permision of file
2. **cpy** is used to copy files and directories. incorrect.... cp is command used to copy files and directories
3. **mkfile** is used to create a new file. Incorrect..... touch command is use to create a new file
4. **catx** is used to concatenate files. incorrect... cat command is use to concatenate and display the contents of the file.
5. **rn** is used to rename files. incorrect, mv is use to rename file.

Part C

complete

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

```
#!/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.

```
name="CDAC Mumbai"
echo "$name"
```

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

```
#!/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.

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

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

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

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".

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.

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.

Ans 9

```
#!/bin/bash

echo "Enter num"
read num

if [ $num -gt 10 ]
then
echo "Num greater than 10
else
echo "Number not greaer than 10"
fi
```

ANs 10

```
#!/bin/bash

# Using nested for loops to print a multiplication table for numbers from 1 to 5
for ((i=1; i<=5; i++)); do
    for ((j=1; j<=5; j++)); do
        echo -n "$((i * j)) "
    done
    echo
done
```

Ans 4

```
echo "Enter the first number: "
read num1

echo "Enter the second number: "
read num2

# Perform addition
result=$((num1 + num2))

# Print the result
echo "The sum is: $result"
```

Ans 5

```
echo "Enter a number: "
read num

if (( num % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi
```

Ans6

```
#!/bin/bash

for ((i=1; i<=5; i++)); do
    echo $i
done
```

Ans 7

```
#!/bin/bash

a=1

while [ $a -le 5 ];

do

echo $a

((a++))

done
```

Ans 8

```
#!/bin/bash

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

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.

```
Ans 11

#!/bin/bash

# Using a while loop to read numbers from the user until a negative number is entered
while true; do
   echo "Enter a number (negative number to exit): "
   read num

   # Check if the number is negative
   if [ $num -lt 0 ]; then
      break
   fi

   # Print the square of the positive number
   echo "The square of $num is $((num * num))"
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

Solution in notebook

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

Solution in notebook

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.

Solution in notebook

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.

Solution in notebook

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?

Solution in notebook

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.