CDAC MUMBAI

Concepts of Operating System

Assignment 2

# Part A

What will the following commands do?

* echo "Hello, World!" --**This will print the output.**
* name="Productive" **--Assign the value to the variable name.**
* touch file.txt **---Used to create empty file.**
* ls -a **-- Display all files Include hidden files and directories in the listing.**
* rm file.txt **--Used to remove a file.**
* cp file1.txt file2.txt **---Used to copy the contents of file1.txt to file2.txt**
* mv file.txt /path/to/directory/  **--Move the file in the destination directory**
* chmod 755 script.sh **-- gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.**
* grep "pattern" file.txt -**-Search this word and Display the line which contains the word “pattern”.**
* kill PID **--terminates 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"
* cat file1.txt file2.txt | sort | uniq
* ls -l | grep "^d"
* grep -r "pattern" /path/to/directory/ **--Search the pattern word in given path.**
* cat file1.txt file2.txt | sort | uniq –d
* chmod 644 file.txt **-- read and write permission for owner, read-only permission for group and others)**
* cp -r source\_directory destination\_directory **–Copy the content and sub content into destination directory.**
* find /path/to/search -name "\*.txt" **–Display the name of the file with extension .txt.**
* chmod u+x file.txt  **–Used to give the execute(x) permission to the owner.**
* echo $PATH **–Print the value of PATH 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. --**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**

**Explanation: The command rm -rf file.txt deletes a file named file.txt forcefully without asking for confirmation. The -r option is typically used for directories to remove them recursively, but it does not affect the behavior for a regular file. The -f option stands for "force" and tells rm to ignore nonexistent files and never prompt.**

Identify the Incorrect Commands:

1. chmodx is used to change file permissions. –**Incorrect Command**

**Correct command is chmod +x this command adds the execute (x) permission to a file.**

1. cpy is used to copy files and directories. **.**

**Correct command is cp source\_file destination\_file**

1. mkfile is used to create a new file. -**Correct Command**

**mkfile creates one or more files that are suitable for use as NFS-mounted swap areas.**

1. catx is used to concatenate files. **–Incorrect Command**

**Correct command is cat file1 file2 >file3 --🡪this will concatenate file1,file2 contents in file3**

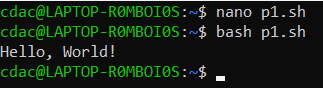
1. rn is used to rename files. –**Incorrect Command**

**Correct command is mv**

# Part C

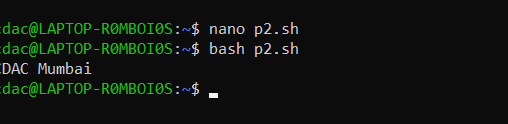
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



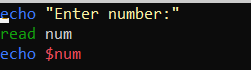
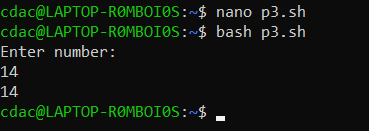


Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

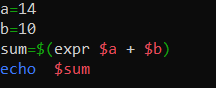


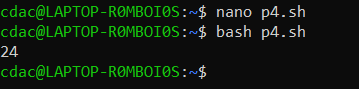


Question 3: Write a shell script that takes a number as input from the user and prints it.

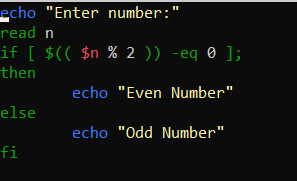
 

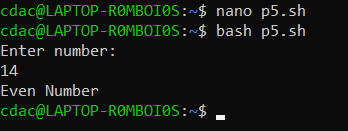
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



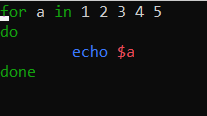


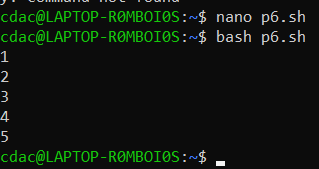
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



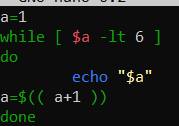


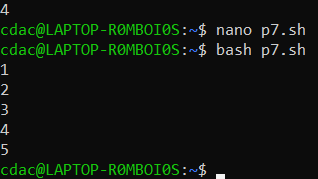
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



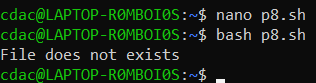
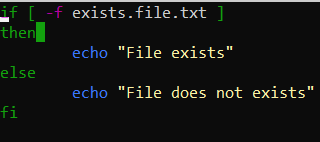


Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

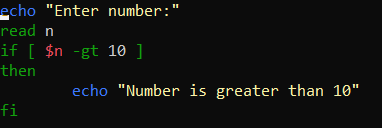


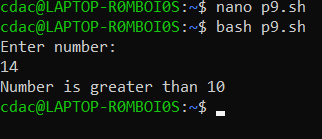


Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



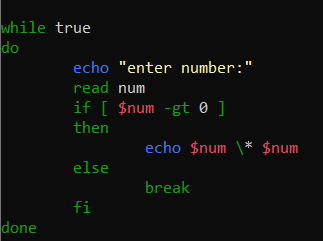
Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

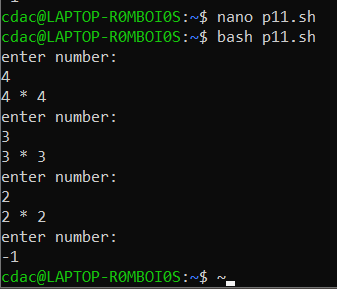




Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

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.





# 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.

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Waiting Time |
| P1 | 0 | 5 | 0 |
| P2 | 1 | 3 | 4 |
| P3 | 2 | 6 | 6 |

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

Gantt Chart

0 5 8 14

**Average Waiting Time=3.3**

1. 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Waiting Time | Completion Time | Turn Around Time |
| P1 | 0 | 3 | 0 | 3 | 3 |
| P2 | 1 | 5 | 7 | 13 | 12 |
| P3 | 2 | 1 | 1 | 4 | 2 |
| P4 | 3 | 4 | 1 | 8 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P3 | P4 | P2 |

Gantt Chart

0 3 4 8 13

**Average Turn Around Time = 5.5**

1. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Priority | Waiting Time |
| P1 | 0 | 6 | 3 | 6 |
| P6 | 1 | 4 | 1 | 0 |
| P3 | 2 | 7 | 4 | 10 |
| P4 | 3 | 2 | 2 | 2 |

Gantt Chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P4 | P1 | P3 |

0 1 5 7 12 19

**Average Waiting Time = 4.5**

1. 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Arrival Time | Burst Time | Waiting Time | Completion Time | Turn Around Time |
| P1 | 0 | 4 | 6 | 10 | 10 |
| P2 | 1 | 5 | 9 | 15 | 14 |
| P3 | 2 | 2 | 2 | 6 | 4 |
| P4 | 3 | 3 | 7 | 13 | 10 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P1 | P2 | P4 Pr | P2 Pr |

Gantt Chart

0 2 4 6 8 10 12 14 16

**Average Turn Around Time = 9.5**

1. 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?

**x=5**

**After fork()**

**For Parent x=6 and Child x=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.