CDAC MUMBAI

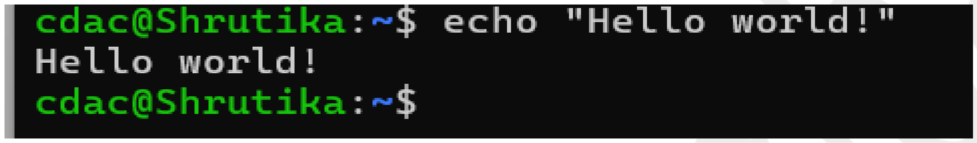
Concepts of Operating System

Assignment 2

Part A

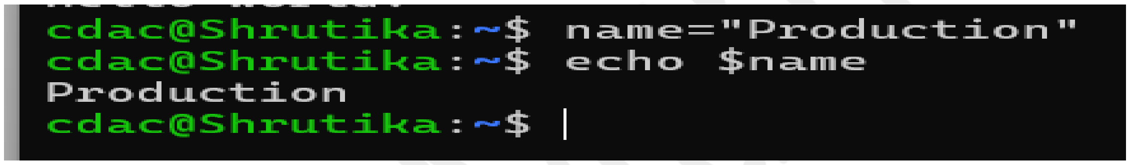
What will the following commands do?

- echo "Hello, World!"



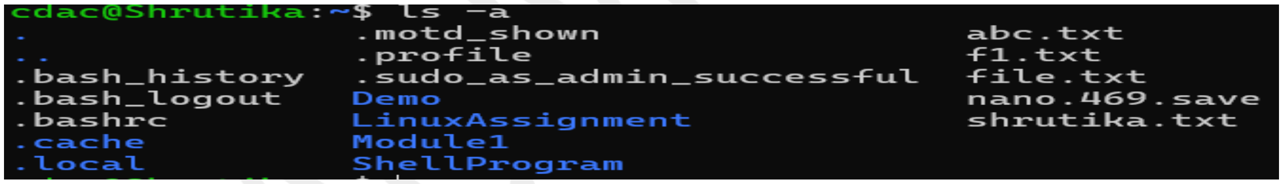
```
cdac@Shrutika:~$ echo "Hello world!"
Hello world!
cdac@Shrutika:~$
```

- name="Productive"



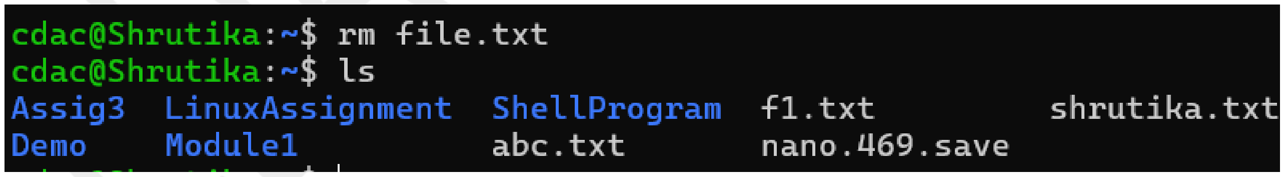
```
cdac@Shrutika:~$ name="Production"
cdac@Shrutika:~$ echo $name
Production
cdac@Shrutika:~$ |
```

- ls -a



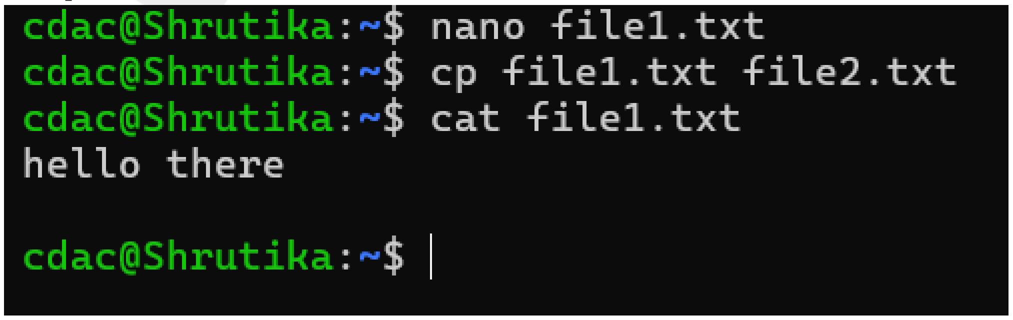
```
cdac@Shrutika:~$ ls -a
.                 .motd_shown                   abc.txt
..                .profile                      f1.txt
.bash_history     .sudo_as_admin_successful     file.txt
.bash_logout      Demo                          nano.469.save
.bashrc           LinuxAssignment               shrutika.txt
.cache            Module1
.local            ShellProgram
```

- rm file.txt



```
cdac@Shrutika:~$ rm file.txt
cdac@Shrutika:~$ ls
Assig3  LinuxAssignment  ShellProgram  f1.txt         shrutika.txt
Demo    Module1          abc.txt       nano.469.save
```

- cp file1.txt file2.txt

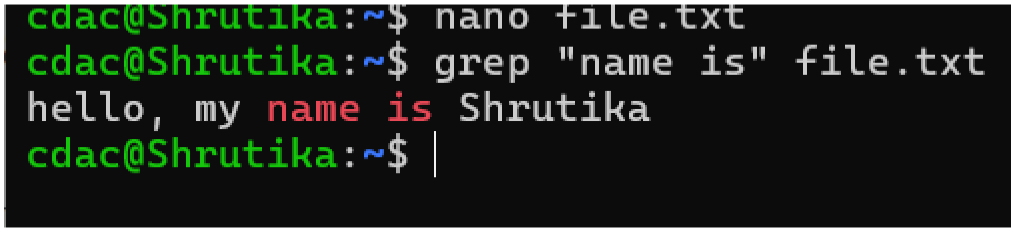


```
cdac@Shrutika:~$ nano file1.txt
cdac@Shrutika:~$ cp file1.txt file2.txt
cdac@Shrutika:~$ cat file1.txt
hello there

cdac@Shrutika:~$ |
```

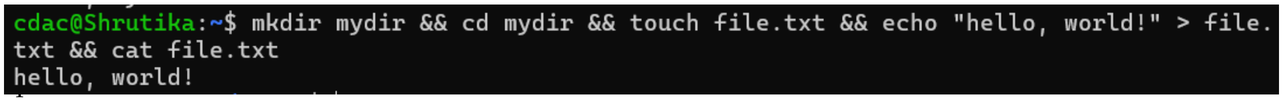
- mv file.txt /path/to/directory/

- grep "pattern" file.txt



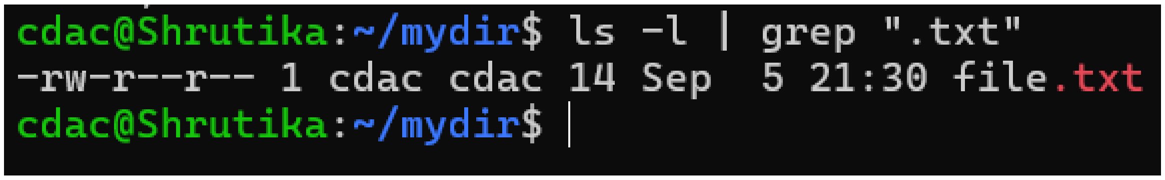
```
cdac@Shrutika:~$ nano file.txt
cdac@Shrutika:~$ grep "name is" file.txt
hello, my name is Shrutika
cdac@Shrutika:~$ 
```

-
-
-
- kill PID
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt



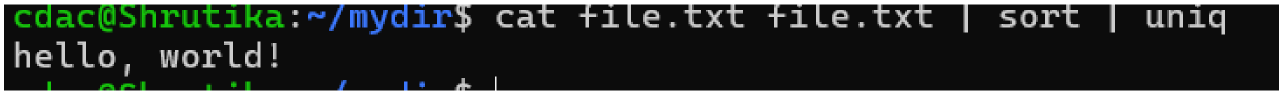
```
cdac@Shrutika:~$ mkdir mydir && cd mydir && touch file.txt && echo "hello, world!" > file.
txt && cat file.txt
hello, world!
```

- ls -l | grep ".txt"



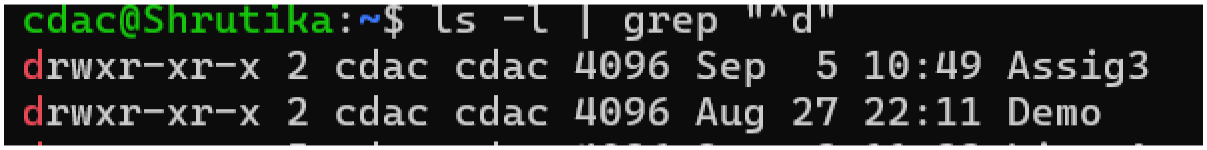
```
cdac@Shrutika:~/mydir$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac 14 Sep  5 21:30 file.txt
cdac@Shrutika:~/mydir$ 
```

- cat file1.txt file2.txt | sort | uniq



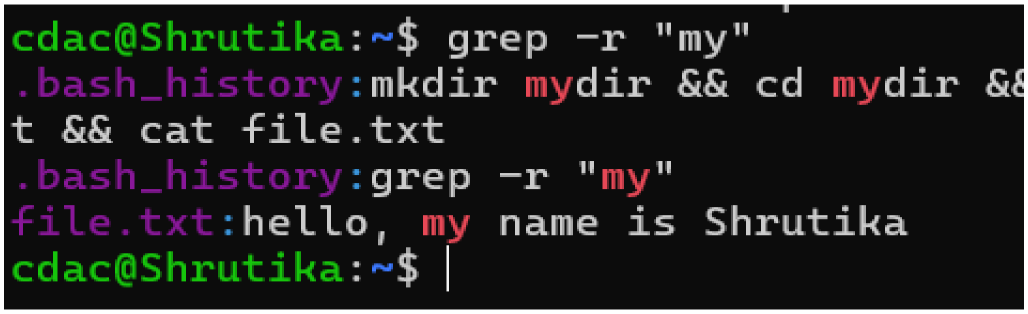
```
cdac@Shrutika:~/mydir$ cat file.txt file.txt | sort | uniq
hello, world!
```

- ls -l | grep "^d"



```
cdac@Shrutika:~$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Sep  5 10:49 Assig3
drwxr-xr-x 2 cdac cdac 4096 Aug 27 22:11 Demo
```

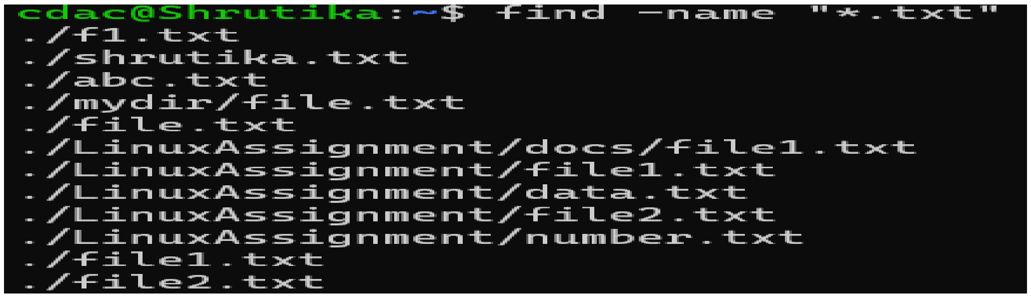
- grep -r "pattern" /path/to/directory/



```
cdac@Shrutika:~$ grep -r "my"
.bash_history:mkdir mydir && cd mydir &&
t && cat file.txt
.bash_history:grep -r "my"
file.txt:hello, my name is Shrutika
cdac@Shrutika:~$ 
```

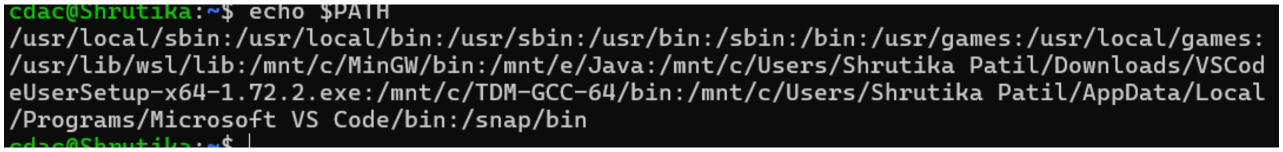
-
-
-
-
-
-
-
-
- cat file1.txt file2.txt | sort | uniq –d
- chmod 644 file.txt
- cp -r source_directory destination_directory

- find /path/to/search -name "*.txt"



```
cdac@Shrutika:~$ find -name "*.txt"
./f1.txt
./shrutika.txt
./abc.txt
./mydir/file.txt
./file.txt
./LinuxAssignment/docs/file1.txt
./LinuxAssignment/file1.txt
./LinuxAssignment/data.txt
./LinuxAssignment/file2.txt
./LinuxAssignment/number.txt
./file1.txt
./file2.txt
```

- echo $PATH



```
cdac@Shrutika:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:
/usr/lib/wsl/lib:/mnt/c/MinGW/bin:/mnt/e/Java:/mnt/c/Users/Shrutika Patil/Downloads/VSCod
eUserSetup-x64-1.72.2.exe:/mnt/c/TDM-GCC-64/bin:/mnt/c/Users/Shrutika Patil/AppData/Local
/Programs/Microsoft VS Code/bin:/snap/bin
cdac@Shrutika:~$
```

<u>Part B</u>

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

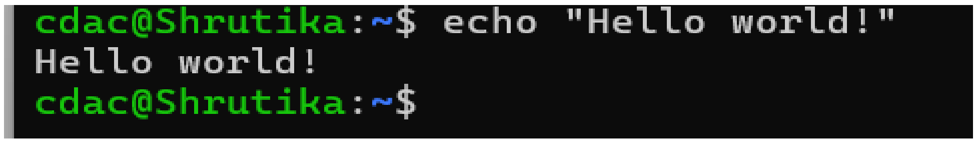
Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.
   chmod is used to change file permissions.
2. **cpy** is used to copy files and directories.
   cp is ued to copy files and directories
3. **mkfile** is used to create a new file.
   touch is used to create a new file.

4. **catx** is used to concatenate files.
   cat is used to concatenate files
5. **rn** is used to rename files.
   mv is used to rename files**.**

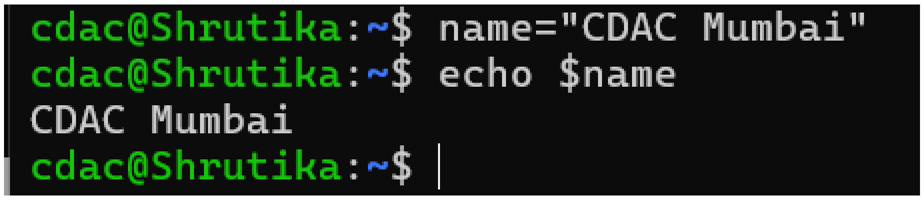
Part C

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



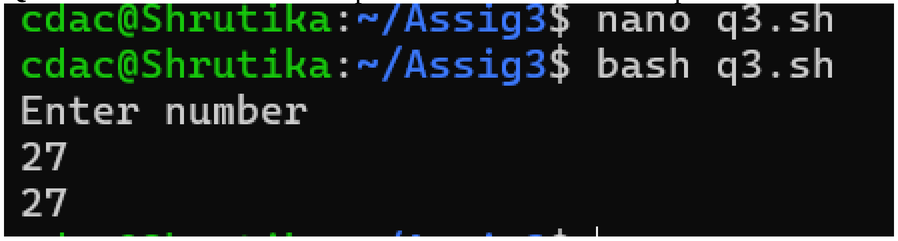
```
cdac@Shrutika:~$ echo "Hello world!"
Hello world!
cdac@Shrutika:~$
```

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



```
cdac@Shrutika:~$ name="CDAC Mumbai"
cdac@Shrutika:~$ echo $name
CDAC Mumbai
cdac@Shrutika:~$
```

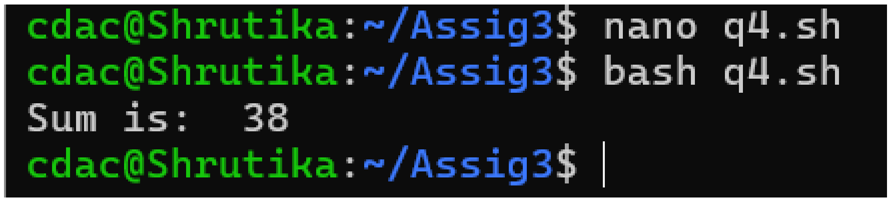
Question 3: Write a shell script that takes a number as input from the user and prints it.



```
cdac@Shrutika:~/Assig3$ nano q3.sh
cdac@Shrutika:~/Assig3$ bash q3.sh
Enter number
27
27
```

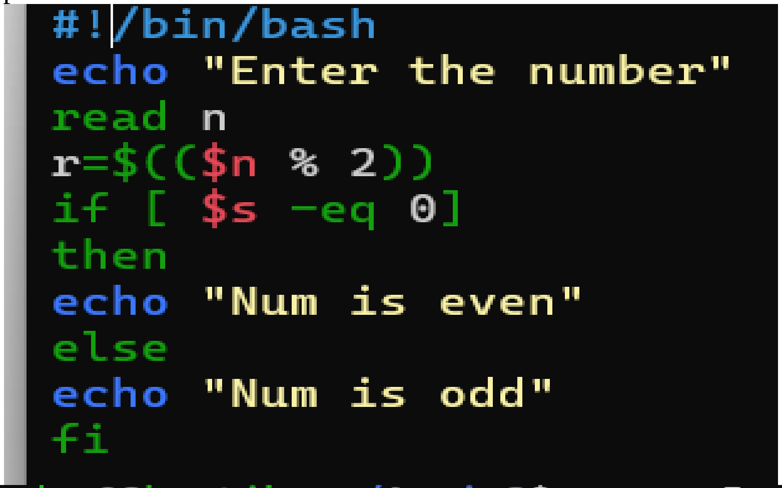
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

```
#!/bin/bash
a=11
b=27
sum=$(($a + $b))
echo "Sum is: " $sum
```

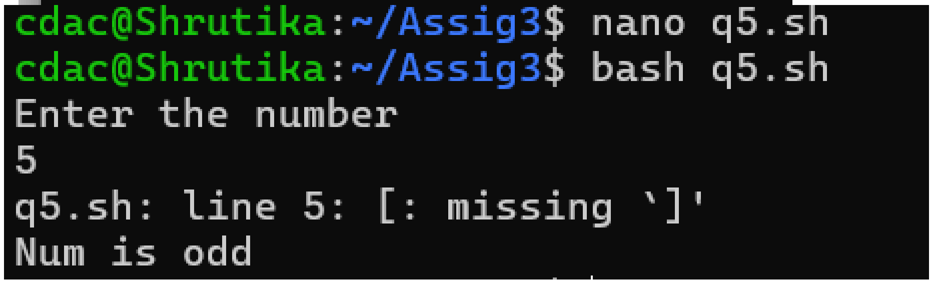


```
cdac@Shrutika:~/Assig3$ nano q4.sh
cdac@Shrutika:~/Assig3$ bash q4.sh
Sum is:  38
cdac@Shrutika:~/Assig3$
```

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

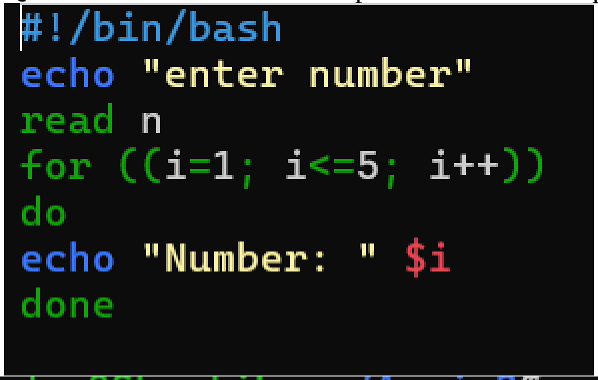


```
#!/bin/bash
echo "Enter the number"
read n
r=$(($n % 2))
if [ $s -eq 0]
then
echo "Num is even"
else
echo "Num is odd"
fi
```

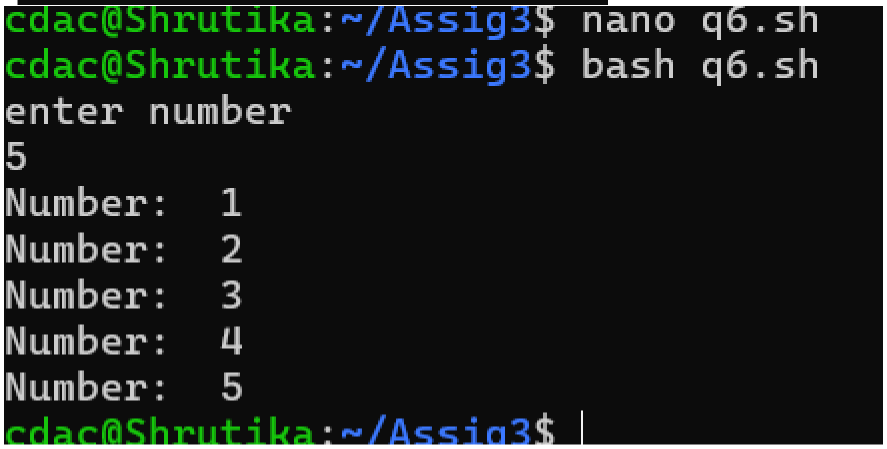


```
cdac@Shrutika:~/Assig3$ nano q5.sh
cdac@Shrutika:~/Assig3$ bash q5.sh
Enter the number
5
q5.sh: line 5: [: missing `]'
Num is odd
```

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



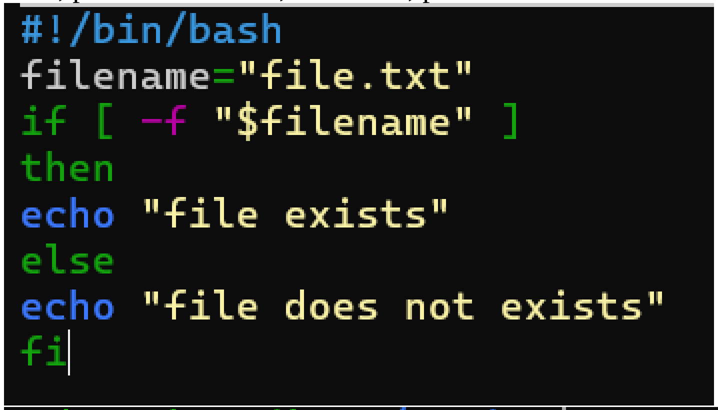
```
#!/bin/bash
echo "enter number"
read n
for ((i=1; i<=5; i++))
do
echo "Number: " $i
done
```



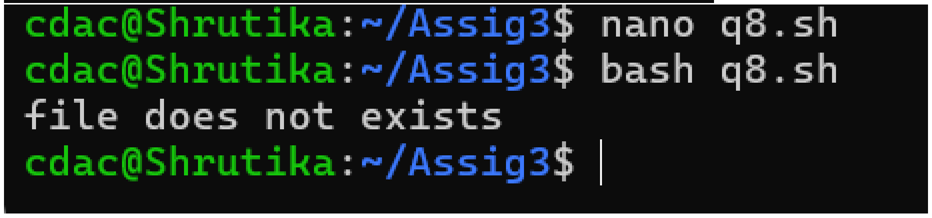
```
cdac@Shrutika:~/Assig3$ nano q6.sh
cdac@Shrutika:~/Assig3$ bash q6.sh
enter number
5
Number:  1
Number:  2
Number:  3
Number:  4
Number:  5
cdac@Shrutika:~/Assig3$
```

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".

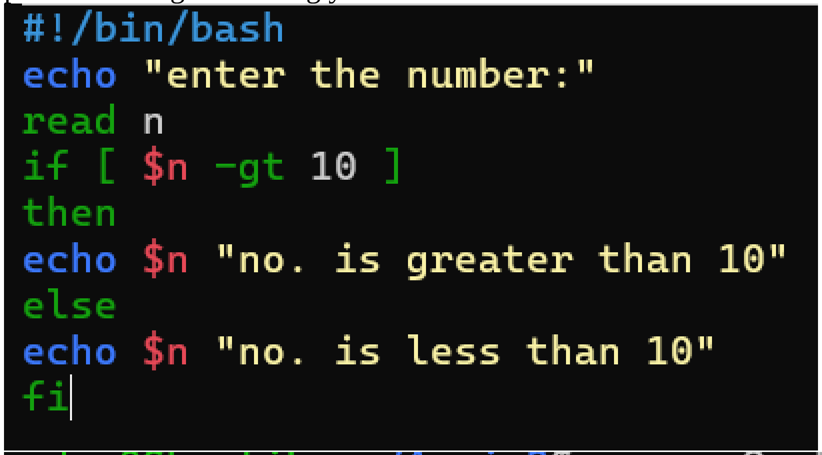


```
#!/bin/bash
filename="file.txt"
if [ -f "$filename" ]
then
echo "file exists"
else
echo "file does not exists"
fi
```

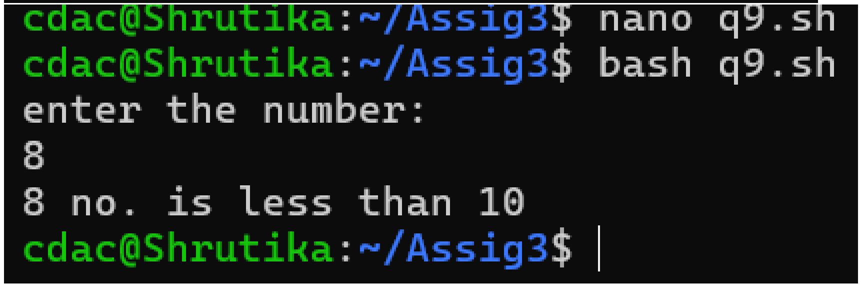


```
cdac@Shrutika:~/Assig3$ nano q8.sh
cdac@Shrutika:~/Assig3$ bash q8.sh
file does not exists
cdac@Shrutika:~/Assig3$
```

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.



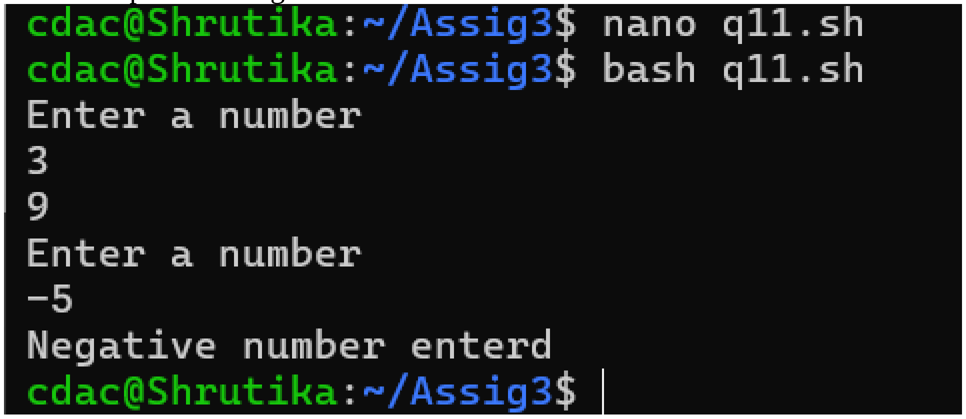
```
#!/bin/bash
echo "enter the number:"
read n
if [ $n -gt 10 ]
then
echo $n "no. is greater than 10"
else
echo $n "no. is less than 10"
fi
```



```
cdac@Shrutika:~/Assig3$ nano q9.sh
cdac@Shrutika:~/Assig3$ bash q9.sh
enter the number:
8
8 no. is less than 10
cdac@Shrutika:~/Assig3$
```

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 entersa negative number. For each positive number entered, print its square. Use the **break** statement to exit theloop when a negative number is entered.



```
cdac@Shrutika:~/Assig3$ nano q11.sh
cdac@Shrutika:~/Assig3$ bash q11.sh
Enter a number
3
9
Enter a number
-5
Negative number enterd
cdac@Shrutika:~/Assig3$
```

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
   Ans. Operating system is software that manages computer hardware and software resources and provides services for computer programs. Primary functions include managing hardware resources (CPU, memory, I/O devices), providing a user interface, executing and managing tasks, and handling system security and file management.

2. Explain the difference between process and thread.
   Ans. Processes are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution. PCB (Process Control Block) holds the context of process. A process can create other processes which are known as Child Processes. The process can have the following states new, ready, running, waiting, terminated, and suspended.

   Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process. A thread has three states: Running, Ready, and Blocked.

3. What is virtual memory, and how does it work?
   Ans. A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.
   The main visible advantage of this scheme is that programs can be larger than physical memory

4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
   Ans. Multiprogramming: Multiple programs are loaded into memory and executed by the CPU one by one.
   Multitasking: Multiple tasks or processes are executed simultaneously by rapidly switching between them.
   Multiprocessing: Multiple CPUs or cores process multiple tasks simultaneously.

5. What is a file system, and what are its components?
   Ans. A file system is a method an operating system uses to store, organize, and manage files and directories on a storage device.
   Components include the boot sector, file allocation table (FAT), directories, and files.

6. What is a deadlock, and how can it be prevented?
   Ans. A deadlock is a situation where a set of processes is stuck in a state where each process is waiting for a resource held by another, leading to an indefinite halt. It can be prevented by ensuring at least one of the following conditions does not hold: mutual exclusion, hold and wait, no preemption, and circular wait.

7. Explain the difference between a kernel and a shell.
   Ans. The kernel is the core part of the OS that interacts with hardware and manages resources. The shell is a user interface that provides access to the services of the OS, often through command-line input.

8. What is CPU scheduling, and why is it important?
   Ans. The kernel is the core part of the OS that interacts with hardware and manages resources. The shell is a user interface that provides access to the services of the OS, often through command-line input**.**

9. How does a system call work?
   Ans. A system call is a way for a program to request services from an operating system's kernel. This process allows programs to access privileged functionalities that the operating system provides.

10. What is the purpose of device drivers in an operating system?
    Ans. Device Driver in computing refers to a special kind of software program or a specific type of software application that controls a specific hardware device that enables different hardware devices to communicate with the computer's Operating System.

11. Explain the role of the page table in virtual memory management.
    Ans. The page table maps virtual addresses to physical addresses. It keeps track of where the OS has stored data in physical memory (or on disk) corresponding to the virtual memory used by processes.

12. What is thrashing, and how can it be avoided?
    Ans. Thrashing occurs when a system spends more time swapping pages in and out of memory than executing processes. It can be avoided by using better page replacement algorithms.

13. Describe the concept of a semaphore and its use in synchronization.
    Ans. Semaphores are used to manage concurrent processes by controlling access to shared resources. Semaphores can signal and wait, ensuring that critical sections are accessed by only one process at a time.

14. How does an operating system handle process synchronization?
    Ans. OSs use synchronization mechanisms like mutexes, semaphores, and monitors to coordinate processes, ensuring that shared data is accessed in a controlled and safe manner, preventing race conditions and data inconsistency.

15. What is the purpose of an interrupt in operating systems?
    Ans. An interrupt is a signal that causes an operating system (OS) to stop a current process or service and take a specific action. Interrupts can be generated by a device attached to the computer or by a program running on the computer

16. Explain the concept of a file descriptor.

Ans. A file descriptor is an abstract identifier used by a process to access a file or other input/output resource, such as a socket. It's a handle through which a process interacts with these resources

17. How does a system recover from a system crash?
    Ans. After a system crash, the OS may use techniques like checkpointing, logs, and backup copies to restore the system to a stable state. Recovery processes include restarting services, restoring data from backups, and repairing corrupted files.

18. Describe the difference between a monolithic kernel and a microkernel.
    Ans. The kernel manages the operations of the computer, In microkernel, the user services and kernel services are implemented in different address spaces. The user services are kept in the user address space, and kernel services are kept under the kernel address space.
    Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services are implemented in the same address space.

19. What is the difference between internal and external fragmentation?
    Ans. Internal fragmentation: Wasted space within allocated memory blocks due to allocation units being larger than the requested memory.
    External fragmentation: Free memory is scattered, making it difficult to allocate contiguous blocks despite sufficient total free space.

20. How does an operating system manage I/O operations?
    Ans.. The OS manages the I/O operations by using interrupts, buffering data, device drivers.

21. Explain the difference between preemptive and non-preemptive scheduling.
    Ans. Preemptive scheduling: The OS can interrupt a running process to assign the CPU to another process.
    Non-preemptive scheduling: A running process must finish its CPU burst before the CPU can be assigned to another process.

22. What is round-robin scheduling, and how does it work?
    Ans. Round-robin scheduling is a preemptive scheduling algorithm where each process is assigned a fixed time slice (quantum). Processes are cycled through in a queue, ensuring each gets a fair share of CPU time.

23. Describe the priority scheduling algorithm. How is priority assigned to processes?
    Ans. In priority scheduling, processes are assigned priorities, and the CPU is allocated to the process with the highest priority.
    Priorities can be assigned based on factors like process type, resource needs, or user-defined criteria.

24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
    Ans. SJN (or SJF, Shortest Job First) selects the process with the smallest execution time to run next. It's used to minimize waiting time and optimize overall CPU usage, but it can lead to starvation for longer processes.

25. Explain the concept of multilevel queue scheduling.
    Ans. Multilevel queue scheduling divides the ready queue into several separate queues, each with its scheduling algorithm

26. What is a process control block (PCB), and what information does it contain?
    Ans. The PCB is a data structure in the OS that stores all the information about a process, including its process ID, state, CPU registers, scheduling information, memory management information, and I/O status.

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?
    Ans. Processes can coordinate and interact with one another using a method called inter-process communication (IPC) .

29. What is process synchronization, and why is it important?
    Ans. Process synchronization ensures that concurrent processes operate correctly when accessing shared resources, preventing race conditions and ensuring data integrity.

30. Explain the concept of a zombie process and how it is created.
    Ans. A zombie process occurs when a child process has terminated, but its exit status has not been read by its parent process. It remains in the process table until the parent reads the status or terminates.

31. Describe the difference between internal fragmentation and external fragmentation.
    Ans. Internal fragmentation: Wasted space within allocated memory blocks due to allocation units being larger than the requested memory.
    External fragmentation: Free memory is scattered, making it difficult to allocate contiguous blocks despite sufficient total free space.

32. What is demand paging, and how does it improve memory management efficiency?
    Ans. Demand paging loads pages into memory only when they are needed, reducing the memory usage and improving efficiency by avoiding loading unnecessary pages.

33. Explain the role of the page table in virtual memory management.
    Ans. The page table maps virtual addresses to physical addresses. It keeps track of where the OS has stored data in physical memory corresponding to the virtual memory used by processes.

34. How does a memory management unit (MMU) work?
    Ans. MMU stands for Memory management unit also known as PMMU (paged memory management unit), Every computer system has a memory management unit , it is a hardware component whose main purpose is to convert virtual addresses created by the CPU into physical addresses in the computer's memory. In simple words, it is responsible for memory management.

35. What is thrashing, and how can it be avoided in virtual memory systems?
    Ans. Thrashing occurs when a system spends more time swapping pages in and out of memory than executing processes, usually due to insufficient memory. It can be avoided by adjusting the degree of multiprogramming or using better page replacement algorithms.

36. What is a system call, and how does it facilitate communication between user programs and the operating system?
    Ans. A system call provides a controlled entry point to the services offered by the OS. When a user program requests a service from the OS, it triggers a system call, which switches the process to kernel mode and executes the corresponding kernel function.

37. Describe the difference between a monolithic kernel and a microkernel.
    Ans. The kernel manages the operations of the computer, In microkernel, the user services and kernel services are implemented in different address spaces. The user services are kept in the user address space, and kernel services are kept under the kernel address space.
    In a Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services are implemented in the same address space.

38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.

Ans. A race condition occurs when the outcome of a process depends on the timing of other processes, leading to unpredictable results. It can be prevented using synchronization mechanisms like locks, semaphores.

40. Describe the role of device drivers in an operating system.
Ans. Device drivers are specialized programs that allow the OS to communicate with hardware devices.

41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
Ans. A zombie process occurs when a child process has terminated, but its exit status has not been read by its parent process. It remains in the process table until the parent reads the status or terminates.

A zombie process can be prevented by ensuring the parent process calls wait() or waitpid() to read the child's exit status.
Ans. An orphan process occurs when its parent process terminates before the child process. The OS handles orphan processes by reassigning them to the init process (PID 1), which then waits for their termination.

42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
Ans. A parent process creates a child process, which inherits most of the parent's attributes, including environment variables, file descriptors, and memory segments. The parent can control or communicate with the child process.

43. What is the relationship between a parent process and a child process in the context of process management?
Ans. In many operating systems, the fork system call is an essential operation. The fork system call allows the creation of a new process. When a process calls the fork(), it duplicates itself, resulting in two processes running at the same time. The new process that is created is called a child process. It is a copy of the parent process. The fork system call is required for process creation and enables many important features such as parallel processing, multitasking, and the creation of complex process hierarchies.

44. How does the fork() system call work in creating a new process in Unix-like operating systems?
Ans. A parent process can use the wait() or waitpid() system calls to pause execution until one or more child processes have terminated, allowing the parent to retrieve the child's exit status.

45. Describe how a parent process can wait for a child process to finish execution.
Ans. A parent process can use the wait() or waitpid() system calls to pause execution until one or more child processes have terminated, allowing the parent to retrieve the child's exit status.

46. What is the significance of the exit status of a child process in the wait() system call?
Ans. The exit status returned by wait() indicates whether the child process terminated normally or abnormally.

47. How can a parent process terminate a child process in Unix-like operating systems?
Ans. A parent process can terminate a child process using the kill() system call.

48. Explain the difference between a process group and a session in Unix-like operating systems.
Ans. A process group is a collection of related processes which can all be signalled at once. A session is a collection of process groups, which are either attached to a single terminal device (known as the controlling terminal) or not attached to any terminal.

49. Describe how the exec() family of functions is used to replace the current process image with a

new one.
Ans. The exec() family of functions replaces the current process image with a new program image, effectively running a new program within the same process. The process ID remains the same, but the code, data, and stack are replaced

50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
Ans. waitpid() allows a parent to wait for a specific child process to terminate, identified by its PID. Unlike wait(), which waits for any child process, waitpid() can be more selective and can also wait for non-child processes if used with certain flags.

51. How does process termination occur in Unix-like operating systems?
Ans. Process termination occurs when a process finishes executing or is killed by a signal.

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?
Ans. The long-term scheduler controls the admission of processes into the system, determining which processes should be brought into the ready queue. It influences the degree of multiprogramming by controlling the number of processes loaded into memory.

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?
Ans. The short-term scheduler (CPU scheduler) makes frequent, quick decisions about which process should run next, usually based on a scheduling algorithm. The long-term scheduler runs less frequently, deciding which processes should enter the ready queue. The medium-term scheduler swaps processes in and out of memory to balance load and manage memory.

54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.
Ans. The medium-term scheduler may be invoked when the system is low on memory, and processes need to be swapped out to free up space. It helps by reducing the degree of multiprogram ming temporarily, ensuring the most important processes have enough resources to run 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.