**Question 1**

Write a simplified version of the Linux utility ls with invocation structure

#include <stdio.h>

#include <sys/stat.h>

#include <time.h>

#include <unistd.h>

void print\_details(const char \*file) {

struct stat st;

if (lstat(file, &st) == -1) {

perror("lstat");

return;

}

printf("b. Inode number: %ld\n", (long)st.st\_ino);

char time\_buffer[80];

struct tm \*tm = localtime(&st.st\_ctime);

strftime(time\_buffer, sizeof(time\_buffer), "%c", tm);

printf("c. Last change: %s\n", time\_buffer);

tm = localtime(&st.st\_mtime);

strftime(time\_buffer, sizeof(time\_buffer), "%c", tm);

printf("d. Last modification: %s\n", time\_buffer);

printf("e. Number of hard links: %ld\n", (long)st.st\_nlink);

}

int main(int argc, char \*argv[]) {

if (argc != 2) {

fprintf(stderr, "Usage: %s <file>\n", argv[0]);

return 1;

}

print\_details(argv[1]);

return 0;

}

**Question 2**

#!/bin/bash

if [ $# -ne 2 ]; then

echo "Usage: test.sh c-sourcefile test-file"

exit 1

fi

c\_sourcefile=$1

test\_file=$2

if [ ! -f "$c\_sourcefile" ]; then

echo "Error: C source file $c\_sourcefile not found."

exit 1

fi

if [ ! -f "$test\_file" ]; then

echo "Error: Test file $test\_file not found."

exit 1

fi

gcc -o temp "$c\_sourcefile" || { echo "Error: Compilation failed."; exit 1; }

if ./temp > actual.txt 2>&1; then

if diff -q actual.txt "$test\_file" >/dev/null; then

echo "Test passed."

else

echo "Test failed. Expected output does not match actual output."

fi

else

echo "Test failed. Compilation or execution failed."

**Question 3**

Write a script renamelower.sh with invocation structure renamelower.sh path That changes all filenames in the directory path to lowercase

#!/bin/bash

# Check if the path is provided

if [ -z $1 ]; then

echo "Usage: $0 path"

exit 1

fi

# Change to the provided path

cd $1

# Loop through all files and directories

for file in \*; do

# Rename the file to lowercase

mv "$file" "$(echo $file | tr '[:upper:]' '[:lower:]')"

done

# Change back to the original directory

cd -

**Question 4**

Write a shell script rcat.sh (reverse cat ) with invocation syntax

rcat.sh filename

It takes a file name as input and output the content of filename on the standard output with the content

reversed.

Eg if a file phill.txt (thats not a typo) contains the lines

invoking rcat.sh phill.txt will output

#!/bin/bash

# Check if correct number of arguments is provided

if [ "$#" -ne 1 ]; then

echo "Usage: $0 <filename>"

exit 1

fi

# Check if the file exists

if [ ! -f "$1" ]; then

echo "Error: File '$1' not found."

exit 1

fi

# Reverse and output the content of the file

tac "$1" | rev

**Question 5**

Write a shell script that will take an input file and remove identical lines (or duplicate lines

from the file)

#!/bin/bash

# Check if correct number of arguments is provided

if [ "$#" -ne 1 ]; then

echo "Usage: $0 <input-file>"

exit 1

fi

# Check if the input file exists

if [ ! -f "$1" ]; then

echo "Error: File '$1' not found."

exit 1

fi

# Remove duplicate lines from the input file and output the result

awk '!seen[$0]++' "$1"

**Question 6**

Write a MATLAB program that accepts the scores of a student and determine the grade of the student. Utilize the Acity grading system.

% Grading System

2

3% Prompt the user to enter the student's score

4score = input('Enter the student\'s score: ');

5

6% Determine the grade based on the grading system

7if score >= 80 && score <= 100

8 grade = 'A';

9elseif score >= 70 && score < 80

10 grade = 'B';

11elseif score >= 60 && score < 70

12 grade = 'C';

13elseif score >= 50 && score < 60

14 grade = 'D';

15else

16 grade = 'FAIL';

17end

18

19% Display the grade

20fprintf('The student\'s grade is: %s\n', grade);

**Question 7**

Write a C program that accepts a number and determine whether it is an odd or even number.

#include <stdio.h>

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

// if the number is divisible by 2 without remainder, it's even

if (num % 2 == 0) {

printf("%d is even.\n", num);

} else {

printf("%d is odd.\n", num);

}

return 0;

}

1. **Briefly discuss the four main parts of the Linux system:**

Kernel: The core component that interacts directly with the hardware, manages resources, and provides essential services.

System Libraries: Libraries of pre-compiled functions used by applications to interact with the kernel and hardware.

System Utilities: Essential system management tools and utilities for tasks like file management, process management, and network configuration.

Shell: The command-line interface that allows users to interact with the operating system by entering commands.

1. **List the 4 main categories of Linux distributions with 10 examples each:**

Debian-based:

{Debian, Ubuntu, Linux Mint, elementary OS, Pop!\_OS, Kali Linux, MX Linux, Deepin, Zorin OS, antiX.}

Red Hat-based:

{Red Hat Enterprise Linux (RHEL), CentOS, Fedora, Oracle Linux, Rocky Linux, AlmaLinux, ClearOS, Scientific Linux, Springdale Linux, Vine Linux.}

Arch-based:

{Arch Linux, Manjaro, EndeavourOS, ArcoLinux, Parabola GNU/Linux-libre, Garuda Linux, Artix Linux, Anarchy Linux, BlackArch, ArchBang. }

Slackware-based:

{Slackware, Salix OS, Slax, Zenwalk Linux, Absolute.}

1. **What is a:**

Shell: A program that interprets commands and acts as an intermediary between the user and the operating system, enabling users to execute commands.

b. Shell Programming: Writing scripts or programs using shell commands to automate tasks or perform operations on the system.

1. **Why Shell Programming:**

Shell programming allows users to automate tasks, customize system behavior, and perform complex operations efficiently through **scripting**.

1. **Mention 4 types of Shell:**

Bourne Shell (sh)

Bourne Again Shell (bash)

C Shell (csh)

Korn Shell (ksh)

1. **Differentiate between Absolute and Relative directory references:**

Absolute directory reference: Specifies the exact location of a directory from the root directory (/).

Relative directory reference: Specifies the location of a directory relative to the current working directory.

1. **Explain these commands with examples:**
2. cd: Change directory. Example: cd /home/user/
3. pwd: Print working directory. Example: pwd
4. ls: List directory contents. Example: ls
5. ls –r: List directory contents recursively. Example: ls -r
6. ls –f: List directory contents with full path. Example: ls -f
7. ls –e: option on ls to ensure the same format for all files regardless of their age
8. cp: Copy files or directories. Example: cp file1 file2
9. Rm: Remove files or directories. Example: rm file
10. mv: Move or rename files or directories. Example: mv file1 file2
11. mkdir: Create a directory. Example: mkdir directory
12. sort: Sort lines of text files. Example: sort file.txt
13. ps: Display information about active processes. Example: ps
14. **Differentiate between hard and symbolic links:**

Hard links: Direct references to the physical location of a file on disk. Changes in the original file reflect in the hard link, and vice versa.

Symbolic links: Indirect references to the original file's location. They can point to files or directories across different file systems, and changes in the original file do not affect the symbolic link.

1. **What is:**

a. Input redirection: Redirecting input from a file or command to a program. Example: cat < file.txt

b. Output redirection: Redirecting output from a program to a file or another command. Example: ls > output.txt

**/bin/sh** is the default system shell in many Unix-like operating systems. It stands for the Bourne shell, which was one of the earliest Unix shells. On many systems, **/bin/sh** is symlinked to another shell like Bash or Dash.

Here's the general syntax:  
sh script\_name.sh

**12. Write a shell script that accepts the name of your beloved and displays it to the whole world and state whether you love him or her.**

#!/bin/bash

echo "Enter the name of your beloved:"

read beloved\_name

echo "You entered: $beloved\_name"

echo "I love $beloved\_name."

**State 10 different commands in Linux and how they are used with examples:**

ls: List directory contents. Example: ls -l

pwd: Print working directory. Example: pwd

cd: Change directory. Example: cd /home/user

mkdir: Create a directory. Example: mkdir new\_directory

rm: Remove files or directories. Example: rm file.txt

cp: Copy files or directories. Example: cp file1.txt file2.txt

mv: Move or rename files or directories. Example: mv file1.txt

new\_directory cat: Display file contents. Example: cat

file.txt grep: Search for patterns in files. Example: grep pattern file.txt echo: Print arguments to the standard output.

Example: echo "Hello, world!"

**14.What are the two ways of manipulating files and directories in Linux:**

-Command-line utilities: Using commands like cp, mv, rm, mkdir, etc.

- Graphical file managers: Using GUI-based tools like Nautilus, Dolphin, Thunar, etc.

1**5. Differentiate between System Calls and Library Functions:**

a) System Calls:

Interface between user-space applications and the operating system kernel. Invokes kernel services directly.

b) Library Functions: Precompiled functions provided by libraries like libc. Implement higher-level operations and often wrap system calls for easier use.

**16.What are the disadvantages of:**

1. System Calls:

-System calls involve context switching, which incurs overhead.

- System calls may vary between different operating systems.

Security: Direct access to kernel services can pose security risks if not handled properly.

1. Library Functions: Dependency: Library functions rely on underlying system calls, making them susceptible to changes in system behavior.

**18.Differentiate between Program, Process, & Thread:**

1. Program: Collection of instructions and data stored in a file. Passive entity.
2. Process: Running instance of a program with its own memory space, resources, and execution context. Active entity.
3. Thread: Execution unit within a process that shares the same memory space and resources as other threads within the same process. Allows for concurrent execution within a process.

**19.State the range of the PID representation:**

The range of PID (Process ID) representation typically depends on the system. In most modern Unix-like systems, PIDs are 16-bit or 32-bit integers, which means they can range from 0 to 65,535 or 0 to 4,294,967,295, respectively.

**20.2 Differentiate between Child and Parent Process:**

Parent Process: A process that creates another process is termed as the parent process.

Child Process: A process that is created by another process is termed as the child process.

**21.What is forking:**

Forking is the process by which a new process is created by a currently running process. The new process is called the child process, and it is an identical copy of the parent process.

**22.Differentiate between Idle Process and Init Process:**

1. Idle Process: An idle process is a system process that doesn't have any task assigned to it. It exists to ensure that there is always at least one process running.
2. Init Process: Init (short for initialization) is the first process started during booting of the computer system. It is the parent process of all other processes and has process ID 1.

**23.What is Interprocess Communication (IPC):**

Interprocess Communication (IPC) is a mechanism that allows processes to communicate and synchronize with each other. It enables data exchange, coordination, and cooperation between separate processes.

**24.What is a system:**

A system refers to a collection of interrelated components working together to achieve a common goal.

**25What makes system development difficult and how can it be mitigated:**

System development can be challenging due to factors such as complex requirements, changing user needs, technological advancements, tight budgets, and strict deadlines.

To mitigate these challenges, proper planning, clear communication, collaboration between stakeholders, continuous testing and feedback, and adherence to best practices can be employed.