**SDM College of Engineering and Technology**

Dhavalagiri, Dharwad-580 002. Karnataka State. India.

Email: [principal@sdmcet.ac.in](mailto:principal@sdmcet.ac.in), [cse.sdmcet@gmail.com](mailto:cse.sdmcet@gmail.com)

Ph: 0836-2447465/ 2448327 Fax: 0836-2464638 Website: sdmcet.ac.in

**Department**

**of**

**COMPUTER SCIENCE AND ENGINEERING**

**Report on**

**Minor Work**

**[22UCSC501- DATABASE MANAGEMENT SYSTEMS]**

**Course Teacher: Prof. Rashmi Patil**

****

**2024- 2025**

Submitted by

**Nanda Gaddad**

**2SD23CS405**

**4th Semester B division**

**Table of Contents**

[Activity-1: 3](#_Toc180781440)

[Problem Statement: 3](#_Toc180781441)

[Program: 3](#_Toc180781442)

[Activity-2: 6](#_Toc180781443)

[Problem Statement: 6](#_Toc180781444)

[Program: 6](#_Toc180781445)

[Activity-3: 11](#_Toc180781446)

[Problem Statement: 11](#_Toc180781447)

[Program: 11](#_Toc180781448)

# [Activity-1](#gjdgxs):

## Problem Statement:

Write a C program to study all file operations related SYSTEM CALLS supported by UNIX OS and C libraries for file operations.

### Program:

C program demonstrating file operations open(), write(), read(), close(), lseek(), unlink() using system calls in UNIX OS and C libraries for file operations:

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h> // For open()

#include <unistd.h> // For close(), read(), write(), lseek()

#include <string.h> // For strlen()

int main() {

int file;

ssize\_t bytes;

char buffer[100];

// Create and open a file using open() system call

file = open("sample.txt", O\_CREAT | O\_WRONLY | O\_TRUNC, 0644);

if (file < 0) {

perror("Error opening file");

exit(EXIT\_FAILURE);

}

printf("File 'sample.txt' created and opened for writing.\n");

// Write to file using write() system call

char \*data = "This is a test for file operations using system calls.\n";

bytes = write(file, data, strlen(data));

if (bytes < 0) {

perror("Error writing to file");

close(file);

exit(EXIT\_FAILURE);

}

printf("Data written to file: %ld bytes.\n", bytes);

// Close the file using close() system call

if (close(file) < 0) {

perror("Error closing file after writing");

exit(EXIT\_FAILURE);

}

// Open the file for reading

file = open("sample.txt", O\_RDONLY);

if (file < 0) {

perror("Error opening file for reading");

exit(EXIT\_FAILURE);

}

printf("File 'sample.txt' opened for reading.\n");

// Read from the file using read() system call

bytes = read(file, buffer, sizeof(buffer) - 1);

if (bytes < 0) {

perror("Error reading from file");

close(file);

exit(EXIT\_FAILURE);

}

buffer[bytes] = '\0'; // Null-terminate the buffer

printf("Data read from file:\n%s", buffer);

// Repositioning the file offset using lseek() system call

if (lseek(file, 0, SEEK\_SET) < 0) {

perror("Error repositioning file offset");

close(file);

exit(EXIT\_FAILURE);

}

printf("File offset repositioned to the beginning of the file.\n");

// Close the file

if (close(file) < 0) {

perror("Error closing file after reading");

exit(EXIT\_FAILURE);

}

// Delete the file using unlink() system call

if (unlink("sample.txt") < 0) {

perror("Error deleting file");

exit(EXIT\_FAILURE);

}

printf("File 'sample.txt' deleted successfully.\n");

return 0;

}

**System Calls Used:**

1. open(): Opens or creates a file.
   * Flags:
     + O\_CREAT: Create the file if it doesn't exist.
     + O\_WRONLY: Open for writing only.
     + O\_TRUNC: Truncate the file to 0 length if it already exists.
2. write(): Writes data to a file.
3. read(): Reads data from a file.
4. close(): Closes the opened file descriptor.
5. lseek(): Changes the file offset (seek position).
6. unlink(): Deletes a file from the filesystem.

# [Activity-2](#gjdgxs):

## Problem Statement:

Write a C program to demonstrate indexing and associated operations.

### Program:

The C program that demonstrates the concept of indexing and associated operations on an array. This program includes initializing an array, accessing elements using indexing, updating values, and searching for an element.

#include <stdio.h>

#define SIZE 10

// Function to display the array elements

void displayArray(int arr[], int size) {

    printf("Array elements: ");

    for (int i = 0; i < size; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

// Function to update an element at a given index

void updateElement(int arr[], int index, int newValue) {

    if (index >= 0 && index < SIZE) {

        arr[index] = newValue;

        printf("Updated element at index %d to %d.\n", index, newValue);

    } else {

        printf("Index out of bounds!\n");

    }

}

// Function to search for an element and return its index

int searchElement(int arr[], int size, int value) {

    for (int i = 0; i < size; i++) {

        if (arr[i] == value) {

            return i;  // Return the index if the element is found

        }

    }

    return -1;  // Return -1 if the element is not found

}

int main() {

    int arr[SIZE];

    // Taking user input to populate the array

    printf("Enter %d elements for the array:\n", SIZE);

    for (int i = 0; i < SIZE; i++) {

        printf("Element %d: ", i + 1);

        scanf("%d", &arr[i]);

    }

    int choice;

    while (1) {

        printf("Enter your choice: ");

        printf("1. Update an element\t");

        printf("2. Search for an element\t");

        printf("3. display array elements\t");

        printf("4. Exit\n");

        scanf("%d", &choice);

        switch (choice) {

            case 1: {

                // Update operation

                int index, newValue;

                printf("Enter the index you want to update (0-%d): ", SIZE - 1);

                scanf("%d", &index);

                printf("Enter the new value for index %d: ", index);

                scanf("%d", &newValue);

                updateElement(arr, index, newValue);

                break;

            }

            case 2: {

                // Search operation

                int searchValue;

                printf("Enter a value to search in the array: ");

                scanf("%d", &searchValue);

                int foundIndex = searchElement(arr, SIZE, searchValue);

                if (foundIndex != -1) {

                    printf("Element %d found at index %d.\n", searchValue, foundIndex);

                } else {

                    printf("Element %d not found in the array.\n", searchValue);

                }

                break;

            }

            case 3:{

                //displaying the array elements

                displayArray(arr, SIZE);

                break;

            }

            case 4:

                printf("Exiting program.\n");

                return 0;  // Exit the program

            default:

                printf("Invalid choice! Please try again.\n");

        }

    }

    return 0;

}

**Sample input and outputs:**

Enter 10 elements for the array:

Element 1: 12

Element 2: 34

Element 3: 65

Element 4: 90

Element 5: 27

Element 6: 64

Element 7: 72

Element 8: 66

Element 9: 89

Element 10: 93

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

3

Array elements: 12 34 65 90 27 64 72 66 89 93

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

1

Enter the index you want to update (0-9): 5

Enter the new value for index 5: 65

Updated element at index 5 to 65.

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

3

Array elements: 12 34 65 90 27 65 72 66 89 93

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

2

Enter a value to search in the array: 8

Element 8 not found in the array.

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

2

Enter a value to search in the array: 27

Element 27 found at index 4.

Enter your choice: 1. Update an element 2. Search for an element 3. display array elements 4. Exit

4

Exiting program.

# [Activity-3:](#gjdgxs)

## Problem Statement:

Write a Java program to access the given excel file with known file format.

### Program:

A sample java code that reads data from an Excel file with a .xlsx format. The program reads an Excel file from a specified file path(e.g., C:/Users/LENOVO/Downloads/excel-file.xlsx). It uses Workbook and Sheet classes to access the workbook and sheets, each cell is checked for data type (String, Numeric, or Boolean) and printed out. Assuming the structure of the Excel file is known, such as which sheet and cells contain the data which we need.

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

public class ExcelReader {

public static void main(String[] args) {

String filePath = " C:/Users/LENOVO/Downloads/excel-file.xlsx ";

try (FileInputStream file = new FileInputStream(new File(filePath));

Workbook = new XSSFWorkbook(file)) {

// Access the first sheet

Sheet sheet = workbook.getSheetAt(0);

// Loop through all rows in the sheet

for (Row row : sheet) {

// Loop through all cells in the row

for (Cell cell : row) {

switch (cell.getCellType()) {

case STRING:

System.out.print(cell.getStringCellValue() + "\t");

break;

case NUMERIC:

System.out.print(cell.getNumericCellValue() + "\t");

break;

case BOOLEAN:

System.out.print(cell.getBooleanCellValue() + "\t");

break;

default:

System.out.print("Unknown\t");

}

}

System.out.println();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}