SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri, Dharwad-580002, Karnataka State, India.

Email: cse.sdmcet@gmail.com

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Report on DBMS – Minor Assignment

COURSE CODE: 22UCSC501
COURSE TITLE: Database Management System
SEMESTER: 5 DIVISION: A
COURSE TEACHER: Dr. U P Kulkarni

[Academic Year- 2024-25]

Date of Submission: 22-10-2024

Submitted
By

Mr. AbdulBasith A Mulla
USN: 2SD22CS001

Table of Contents

Minor Work:

A1: Write a C program to study all file operations related SYSTEM CALLS supported by UNIX OS and C libraries for file operations.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define FILENAME "dbms.txt"
#define BUFFER_SIZE 100

int main() {
    int fd;  // File descriptor
    char text[] = "Hello, this is a test file.\n";  // Data to write to the file
    char buffer[BUFFER_SIZE];  // Buffer to hold read data

    // 1. Create and open a file for writing
    fd = open(FILENAME, O_CREAT | O_WRONLY | O_TRUNC);
    if (fd == -1) {
        perror("Error opening file for writing");
        return EXIT_FAILURE;
    }
    printf("File '%s' created successfully.\n", FILENAME);

    // 2. Write to the file
    if (write(fd, text, strlen(text)) == -1) {
        perror("Error writing to file");
        close(fd);
        return EXIT_FAILURE;
    }
    printf("Data written to file successfully.\n");

    // 3. Close the file
    if (close(fd) == -1) {
        perror("Error closing file after writing");
        return EXIT_FAILURE;
    }
    printf("File closed successfully after writing.\n");
    // 4. Open the file for reading
    fd = open(FILENAME, O_RDONLY);
    if (fd == -1) {
        perror("Error opening file for reading");
        return EXIT_FAILURE;
    }
    printf("File '%s' opened for reading.\n", FILENAME);

    // 5. Read from the file
    ssize_t bytesRead = read(fd, buffer, sizeof(buffer) - 1);
    if (bytesRead == -1) {
        perror("Error reading from file");
        close(fd);
        return EXIT_FAILURE;
    }
    buffer[bytesRead] = '\0'; // Null-terminate the buffer
    printf("Data read from file: %s", buffer);

    // 6. Close the file after reading
    if (close(fd) == -1) {
        perror("Error closing file after reading");
        return EXIT_FAILURE;
    }
    printf("File closed successfully after reading.\n");

    // 7. Delete the file
    if (remove(FILENAME) == 0) {
        printf("File '%s' deleted successfully.\n", FILENAME);
    } else {
        perror("Error deleting file");
    }

    return EXIT_SUCCESS;
}
```

Output:

```
PS C:\Users\ABDULBASITH-HOME> cd "c:\Users\ABDULBASITH-HOME\Documents\" ; if ($?) {
        gcc dbmsA1.c -o dbmsA1 } ; if ($?) { .\dbmsA1 }
File 'dbms.txt' created successfully.
Data written to file successfully.
File closed successfully after writing.
File 'dbms.txt' opened for reading.
Data read from file: Hello, this is a test file.
File closed successfully after reading.
File 'dbms.txt' deleted successfully.
PS C:\Users\ABDULBASITH-HOME\Documents>
```

A2: Write a C program to demonstrate indexing and associated operations.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_RECORDS 100
#define NAME_LENGTH 50
#define FILENAME "students.csv"

typedef struct {
    int id;
    char name[NAME_LENGTH];
} Student;

typedef struct {
    int id;
    long position; // Position in the data file
} IndexEntry;

void writeRecords(FILE *dataFile) {
    fprintf(dataFile, "ID,Name\n"); // Header row
    Student students[MAX_RECORDS] = {
        {1, "Aman"},
        {2, "Shrikar"},
        {3, "Ravi"},
        {4, "Harish"},
        {5, "Ramesh"}
    };

    for (int i = 0; i < 5; i++) {
        fprintf(dataFile, "%d,%s\n", students[i].id, students[i].name);
    }
}


void createIndex(FILE *dataFile, IndexEntry *index, int *indexSize) {
    fseek(dataFile, 0, SEEK_SET);
    char line[NAME_LENGTH + 10]; // Buffer for reading lines
    *indexSize = 0;

    while (fgets(line, sizeof(line), dataFile)) {
        // Get the position of the current line
        index[*indexSize].position = ftell(dataFile) - strlen(line);
        // Parse the ID from the line
        sscanf(line, "%d,", &index[*indexSize].id);
        (*indexSize)++;
    }
}

void searchRecord(FILE *dataFile, IndexEntry *index, int indexSize, int searchId) {
    for (int i = 0; i < indexSize; i++) {
        if (index[i].id == searchId) {
            char line[NAME_LENGTH + 10];
            fseek(dataFile, index[i].position, SEEK_SET);
            fgets(line, sizeof(line), dataFile);
            printf("Record Found: %s", line);
            return;
        }
    }
    printf("Record with ID %d not found.\n", searchId);
}

int main() {
    FILE *dataFile = fopen(FILENAME, "w+");
    if (dataFile == NULL) {
        perror("Unable to open file");
        return 1;
    }

    // Step 1: Write records to the data file
    writeRecords(dataFile);

    // Step 2: Create an index for the records
    IndexEntry index[MAX_RECORDS];
    int indexSize;
    createIndex(dataFile, index, &indexSize);

    // Step 3: Search for records using the index
    int searchId;
    printf("Enter ID to search for: ");
    scanf("%d", &searchId);
    searchRecord(dataFile, index, indexSize, searchId);

    // Clean up
    fclose(dataFile);
    return 0;
}
```

A3: Write a Java program to access the given excel file with known file format.

```
package dbms123;
import java.io.File;
import java.io.FileInputStream;
import java.util.Iterator;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;

public class ReadExcel {
    public static void main(String[] args) {
        try {
            FileInputStream file = new FileInputStream(new File("input.xlsx"));
            XSSFWorkbook workbook = new XSSFWorkbook(file);
            XSSFSheet sheet = workbook.getSheetAt(0);
            Iterator<Row> rowIterator = sheet.iterator();

            while (rowIterator.hasNext()) {
                Row row = rowIterator.next();
                Iterator<Cell> cellIterator = row.cellIterator();

                while (cellIterator.hasNext()) {
                    Cell cell = cellIterator.next();

                    switch (cell.getCellType()) {
                        case NUMERIC:
                            System.out.print(cell.getNumericCellValue() + "\t");
                            break;
                        case STRING:
                            System.out.print(cell.getStringCellValue() + "\t");
                            break;
                        default:
                            System.out.print("Unknown type\t");
                            break;
                    }
                }
                System.out.println("");
            }
            file.close();
            workbook.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```