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

Minor Work

COURSE CODE: 22UCSC501 COURSE TITLE: DBMS
SEMESTER: V DIVISION: A
COURSE TEACHER: Dr. U P Kulkarni

[Academic Year- 2023-24]

Date of Submission: dd-11-2024

Submitted
By

Mr. Abhishek K S USN: 2SD22CS004

Table of Contents

Environment setup details

To complete this assignment, I employed two distinct Integrated Development Environments (IDEs), each optimized for specific programming languages:

- C Program: C Programs: Crafted using Visual Studio Code. Leveraging its lightweight nature, extensive extension library, and robust C/C++ support, this IDE provided an efficient and customizable platform for debugging and developing C programs.
- Java Program: Java Program: Crafted using Eclipse IDE. Leveraging Eclipse's robust Java development tools, including built-in support for Java libraries and frameworks, this IDE facilitated efficient code refactoring, debugging, and project management, ultimately contributing to the development of a well-structured and efficient Java program.

A1:

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

Solution:

This C code implements a simple file I/O program that allows you to:

1. **Write to a file**: You can input text, which is then written to a file named "example.txt".'
2. **Read from a file**: The program reads the contents of the file and displays them on the console.

3. **Exit the program**: This option terminates the program.

Code:

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

int main() {
   FILE *file;
   size_t bytesRead, bytesWritten;
   char buffer[100];
   int choice;

   // Open a file for reading and writing, create if it doesn't exist
   file = fopen("example.txt", "w+");
   if (file == NULL) {
      perror("Error opening file");
      return 1;
   }

   while (1) {
      printf("\nChoose an option:\n");
      printf("1. Write to file\n");
      printf("2. Read from file\n");
      printf("3. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter text to write: ");
            scanf(" %[^\n]", buffer);
            bytesWritten = fwrite(buffer, sizeof(char), strlen(buffer), file);
            if (bytesWritten < strlen(buffer)) {
               perror("Error writing to file");
               fclose(file);
               return 1;
            }
            fflush(file); // Ensure data is written to the file
```

```
            break;

        case 2:
            if (fseek(file, 0, SEEK_SET) != 0) {
                perror("Error seeking in file");
                fclose(file);
                return 1;
            }
            bytesRead = fread(buffer, sizeof(char), sizeof(buffer) - 1, file);
            if (bytesRead == 0 && ferror(file)) {
                perror("Error reading from file");
                fclose(file);
                return 1;
            }
            buffer[bytesRead] = '\0';
            printf("Read from file: %s\n", buffer);
            break;

        case 3:
            fclose(file);
            return 0;

        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

Output :

```
Choose an option:
1. Write to file
2. Read from file
3. Exit
Enter your choice: 1
Enter text to write: Hi my name is abhishek
```

Choose an option:
1. Write to file
2. Read from file
3. Exit
Enter your choice: 2
Read from file: Hi my name is abhishek

Choose an option:
1. Write to file
2. Read from file
3. Exit
Enter your choice: 3

A2:

Write a C program to demonstrate file indexing and associated operations.

Solution:

This C program sets up a basic file indexing system to handle records with unique IDs and names. It offers four primary features:

1. Add Record: Appends a new record to records.dat. The user inputs the record's ID and name, which are then saved to the file.

2. Create Index: Produces an index file, index.dat, linking each record's ID to its position in records.dat. This facilitates quick lookups.

3. Search Record by ID: Enables the user to find a specific record by its ID. The program reads the index file to locate the file position of the desired ID and retrieves the corresponding record from records.dat.

4. Display All RecordsShows all records stored in records.dat by reading each one in sequence.

Code:

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

#define MAX_RECORDS 100

struct Record {
    int id;
    char name[50];
    int age;
};

struct IndexEntry {
    int id;
    long offset;
};


int main() {
    struct Record records[MAX_RECORDS];
    struct IndexEntry index[MAX_RECORDS];
    int numRecords = 0;
    FILE *fp;
    int choice;

    while (1) {
        printf("\n1. Add Record\n");
        printf("2. Create Index\n");
        printf("3. Search Record by ID\n");
        printf("4. Display All Records\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                if (numRecords >= MAX_RECORDS) {
                    printf("Error: Maximum number of records reached.\n");
```

```
            break;
        }

        printf("Enter ID: ");
        scanf("%d", &records[numRecords].id);
        printf("Enter Name: ");
        scanf("%s", records[numRecords].name);
        printf("Enter Age: ");
        scanf("%d", &records[numRecords].age);

        numRecords++;
        break;

    case 2:
        fp = fopen("records.dat", "wb");
        if (fp == NULL) {
            perror("Error opening file");
            return 1;
        }

        for (int i = 0; i < numRecords; i++) {
            index[i].id = records[i].id;
            index[i].offset = ftell(fp);
            fwrite(&records[i], sizeof(struct Record), 1, fp);
        }

        fclose(fp);
        printf("Index created successfully.\n");
        break;

    case 3:
        if (numRecords == 0) {
            printf("Error: No records to search.\n");
            break;
        }

        int searchID, found = 0;
        printf("Enter ID to search: ");
        scanf("%d", &searchID);
```

```
            for (int i = 0; i < numRecords; i++) {
                if (index[i].id == searchID) {
                    fp = fopen("records.dat", "rb");
                    if (fp == NULL) {
                        perror("Error opening file");
                        return 1;
                    }

                    fseek(fp, index[i].offset, SEEK_SET);
                    fread(&records[i], sizeof(struct Record), 1, fp);
                    fclose(fp);

                    printf("Record found: ID = %d, Name = %s, Age = %d\n", records[i].id,
records[i].name, records[i].age);
                    found = 1;
                    break;
                }
            }

            if (!found) {
                printf("Record not found.\n");
            }

            break;

        case 4:
            if (numRecords == 0) {
                printf("Error: No records to display.\n");
                break;
            }

            printf("ID\tName\tAge\n");
            printf("----------------------\n");
            for (int i = 0; i < numRecords; i++) {
                printf("%d\t%s\t%d\n", records[i].id, records[i].name, records[i].age);
            }
            break;

        case 5:
            return 0;
```

```
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Output:

```
1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 1
Enter ID: 1
Enter Name: abhi
Enter Age: 20

1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 1
Enter ID: 2
Enter Name: akhil
Enter Age: 24

1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 4
```

```
ID      Name    Age
--------------------
1       abhi        20
2       akhil       24

1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 2
Index created successfully.

1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 3
Enter ID to search: 2
Record found: ID = 2, Name = akhil, Age = 24

1. Add Record
2. Create Index
3. Search Record by ID
4. Display All Records
5. Exit
Enter your choice: 5
```

A3:

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

Solution:

This code snippet for reading an Excel file using Apache POI looks great! It effectively reads data from the first sheet of an Excel file and prints out the cell values based on their types:

1. Handle Date Cells: If your Excel file contains date values, you might want to handle them separately.

2. Close Resources: Although you are using try-with-resources, which is good, ensure that all resources are properly closed.

3. Error Handling: Consider adding more specific error handling to provide better feedback in case of different types of exceptions.

Code:

```
//Write Java program to access the given excel file with known file format
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 ReadExcelFile {
   public static void main(String[] args) {
      String excelFilePath = "example.xlsx";
      try (FileInputStream fis = new FileInputStream(new File(excelFilePath));
          Workbook workbook = new XSSFWorkbook(fis)) {

         // Get the first sheet
         Sheet sheet = workbook.getSheetAt(0);

         // Iterate through each row
         for (Row row : sheet) {
            // Iterate through each cell 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 value\t");
                    break;
                }
            }
            System.out.println();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Output:

LAB SESSION	DATE	STATUS
LAB 1	18/05/2004	COMPLETED
LAB 2	26/05/2024	COMPLETED
LAB 3	03/06/2024	IN PROGRESS