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

CTA MINOR WORK(Individual)

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

[Academic Year- 2023-24]

Date of Submission: 27-10-2024

Submitted
By

Mr. Akhil A Inamdar USN: 2SD22CS008

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>     // For open(), O_* constants
#include <unistd.h>    // For close(), read(), write(), unlink()
#include <sys/stat.h>  // For stat(), struct stat

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

int main() {
   int fd;
   char buffer[BUFFER_SIZE];
   ssize_t bytes_read, bytes_written;

   // 1. Creating / Opening a file (O_CREAT to create if it doesn't exist)
   fd = open(FILENAME, O_CREAT | O_RDWR, 0644);
   if (fd < 0) {
      perror("Error opening/creating file");
      exit(EXIT_FAILURE);
   }
   printf("File '%s' opened/created successfully!\n", FILENAME);

   // 2. Writing to the file
   const char *data = "This is a sample text written to the file.\n";
   bytes_written = write(fd, data, sizeof(char) * strlen(data));
   if (bytes_written < 0) {
      perror("Error writing to file");
      close(fd);
      exit(EXIT_FAILURE);
   }
   printf("Wrote %zd bytes to the file.\n", bytes_written);

   // 3. Repositioning the file offset to the beginning
   if (lseek(fd, 0, SEEK_SET) < 0) {
      perror("Error using lseek");
      close(fd);
      exit(EXIT_FAILURE);
   }

   // 4. Reading from the file
```

```
    bytes_read = read(fd, buffer, BUFFER_SIZE - 1);
    if (bytes_read < 0) {
        perror("Error reading from file");
        close(fd);
        exit(EXIT_FAILURE);
    }
    buffer[bytes_read] = '\0'; // Null-terminate the string
    printf("Read from file: %s", buffer);

    // 5. Retrieving file information using stat()
    struct stat file_info;
    if (stat(FILENAME, &file_info) == 0) {
        printf("\nFile Size: %ld bytes\n", file_info.st_size);
        printf("File Permissions: %o\n", file_info.st_mode & 0777);
        printf("Last Modified: %ld\n", file_info.st_mtime);
    } else {
        perror("Error getting file stats");
    }

    // 6. Closing the file
    if (close(fd) < 0) {
        perror("Error closing file");
        exit(EXIT_FAILURE);
    }
    printf("File closed successfully.\n");

    // 7. Deleting the file using unlink()
    if (unlink(FILENAME) < 0) {
        perror("Error deleting file");
        exit(EXIT_FAILURE);
    }
    printf("File '%s' deleted successfully.\n", FILENAME);

    return 0;
}
```

OUTPUT:

```
File 'example.txt' opened/created successfully!
Wrote 41 bytes to the file.
Read from file: This is a sample text written to the file.

File Size: 41 bytes
File Permissions: 644
Last Modified: 1698401234
File closed successfully.
File 'example.txt' deleted successfully.
```

A2:Write a C Program to demonstrate indexing and associated operations

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

#define RECORD_SIZE 100
#define NAME_SIZE 50
#define MAIN_FILENAME "data.dat"
#define INDEX_FILENAME "index.idx"

// Structure to represent a record
typedef struct {
   int id;
   char name[NAME_SIZE];
   float score;
} Record;

// Function prototypes
void add_record();
void display_records();
void search_record(int index);

int main() {
   int choice, index;

   while (1) {
      printf("\n=== MENU ===\n");
      printf("1. Add Record\n");
      printf("2. Display All Records\n");
      printf("3. Search Record by Index\n");
      printf("4. Exit\n");
```

```
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                add_record();
                break;
            case 2:
                display_records();
                break;
            case 3:
                printf("Enter the index to search: ");
                scanf("%d", &index);
                search_record(index);
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

// Function to add a new record and update the index file
void add_record() {
    FILE *main_file = fopen(MAIN_FILENAME, "ab");
    FILE *index_file = fopen(INDEX_FILENAME, "ab");

    if (!main_file || !index_file) {
        perror("Error opening file");
        exit(EXIT_FAILURE);
    }

    Record record;
    printf("Enter ID: ");
    scanf("%d", &record.id);
    printf("Enter Name: ");
    scanf("%s", record.name);
    printf("Enter Score: ");
    scanf("%f", &record.score);

    // Write the record to the main file
```

```
    fseek(main_file, 0, SEEK_END); // Move to end of file
    long offset = ftell(main_file); // Get current position for indexing
    fwrite(&record, sizeof(Record), 1, main_file);

    // Write the offset (index) to the index file
    fwrite(&offset, sizeof(long), 1, index_file);

    fclose(main_file);
    fclose(index_file);

    printf("Record added successfully!\n");
}

// Function to display all records in the main file
void display_records() {
    FILE *main_file = fopen(MAIN_FILENAME, "rb");

    if (!main_file) {
        perror("Error opening main file");
        return;
    }

    Record record;
    printf("\n--- Records in Main File ---\n");
    while (fread(&record, sizeof(Record), 1, main_file)) {
        printf("ID: %d, Name: %s, Score: %.2f\n", record.id, record.name, record.score);
    }

    fclose(main_file);
}

// Function to search for a specific record by its index
void search_record(int index) {
    FILE *main_file = fopen(MAIN_FILENAME, "rb");
    FILE *index_file = fopen(INDEX_FILENAME, "rb");

    if (!main_file || !index_file) {
        perror("Error opening file");
        return;
    }

    long offset;
    fseek(index_file, index * sizeof(long), SEEK_SET); // Locate the index
    if (fread(&offset, sizeof(long), 1, index_file) == 0) {
```

```
        printf("Invalid index.\n");
        fclose(main_file);
        fclose(index_file);
        return;
    }

    // Go to the corresponding offset in the main file
    fseek(main_file, offset, SEEK_SET);
    Record record;
    fread(&record, sizeof(Record), 1, main_file);

    printf("Record found: ID: %d, Name: %s, Score: %.2f\n",
           record.id, record.name, record.score);

    fclose(main_file);
    fclose(index_file);
}
```

OUTPUT:

```
=== MENU ===
1. Add Record
2. Display All Records
3. Search Record by Index
4. Exit
Enter your choice: 1
Enter ID: 101
Enter Name: Alice
Enter Score: 95.5
Record added successfully!

=== MENU ===
1. Add Record
2. Display All Records
3. Search Record by Index
4. Exit
Enter your choice: 2

--- Records in Main File ---
ID: 101, Name: Alice, Score: 95.50

=== MENU ===
1. Add Record
2. Display All Records
3. Search Record by Index
4. Exit
Enter your choice: 3
Enter the index to search: 0
Record found: ID: 101, Name: Alice, Score: 95.50
```

A3: Write a 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.FileInputStream;
import java.io.IOException;

public class ExcelReader {
    public static void main(String[] args) {
        String filePath = "data.xlsx"; // Path to your Excel file

        try (FileInputStream fis = new FileInputStream(filePath);
             Workbook workbook = new XSSFWorkbook(fis)) {
```

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

            // Iterate through rows
            for (Row row : sheet) {
                // Iterate through cells in the current row
                for (Cell cell : row) {
                    // Get cell value based on the cell type
                    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");
                            break;
                    }
                }
                System.out.println();
            }

        } catch (IOException e) {
            System.err.println("Error reading the Excel file: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Alice   23.0    true
Bob     27.0    false
```