SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri, Dharwad-580002, Karnataka State, India.

Email: cse.sdmcet@gmail.com

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MINOR WORK REPORT

COURSE CODE: 22UCSC501 COURSE TITLE: DATABASE MANAGEMENT SYSTEM

SEMESTER: V DIVISION: A

COURSE TEACHER: Dr. U. P. Kulkarni

[Academic Year- 2024-25]

Date of Submission: 22-10-2024

Submitted

By

Ms. Anusha Bennal USN: 2SD22CS014

Table of Contents

ASSIGNMENT – 1

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

Theory: File operations related SYSTEM CALLS supported by UNIX OS and C libraries are:

1. Opens a file: Creates or opens a file named "my_file.txt" for reading and writing.
2. Reads from the file: Reads data from the file into a buffer.
3. Writes to the file: Writes the read data back to the file.
4. Closes the file: Closes the file descriptor.
5. Checks file existence: Verifies if the file "my_file.txt" exists.
6. Gets file information: Retrieves information about the file, such as size and permissions.

Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>


#define FILENAME "example.txt"

#define BUFFER_SIZE 1024


void file_operations_system_calls() {

    int fd;
```

```
char buffer[BUFFER_SIZE];
ssize_t bytesRead, bytesWritten;

// Creating a file using system call
fd = open(FILENAME, O_CREAT | O_WRONLY | O_TRUNC, S_IRUSR | S_IWUSR);
if (fd == -1) {
    perror("Error creating file");
    exit(EXIT_FAILURE);
}
printf("File created successfully with file descriptor: %d\n", fd);

// Writing to the file
const char *text = "Hello, World! This is a test of system calls.\n";
bytesWritten = write(fd, text, strlen(text));
if (bytesWritten == -1) {
    perror("Error writing to file");
    close(fd);
    exit(EXIT_FAILURE);
}
printf("Written %zd bytes to the file.\n", bytesWritten);

// Closing the file
close(fd);
```

```
printf("File closed successfully.\n");

// Opening the file for reading
fd = open(FILENAME, O_RDONLY);
if (fd == -1) {
    perror("Error opening file");
    exit(EXIT_FAILURE);
}
printf("File opened successfully for reading with file descriptor: %d\n", fd);

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

// Closing the file
close(fd);
printf("File closed successfully.\n");
```

```
}

void file_operations_c_library() {
    FILE *file;
    char buffer[BUFFER_SIZE];

    // Creating and writing to the file using C library functions
    file = fopen(FILENAME, "w");
    if (!file) {
        perror("Error opening file with fopen");
        exit(EXIT_FAILURE);
    }
    printf("File created successfully using fopen.\n");

    const char *text = "Hello, World! This is a test of C library functions.\n";
    size_t bytesWritten = fwrite(text, sizeof(char), strlen(text), file);
    if (bytesWritten != strlen(text)) {
        perror("Error writing to file with fwrite");
        fclose(file);
        exit(EXIT_FAILURE);
    }
    printf("Written %zu bytes to the file using fwrite.\n", bytesWritten);
```

```
// Closing the file
fclose(file);
printf("File closed successfully using fclose.\n");

// Opening the file for reading using C library functions
file = fopen(FILENAME, "r");
if (!file) {
    perror("Error opening file with fopen");
    exit(EXIT_FAILURE);
}
printf("File opened successfully for reading using fopen.\n");

// Reading from the file
size_t bytesRead = fread(buffer, sizeof(char), BUFFER_SIZE - 1, file);
if (bytesRead == 0 && ferror(file)) {
    perror("Error reading from file with fread");
    fclose(file);
    exit(EXIT_FAILURE);
}
buffer[bytesRead] = '\0'; // Null-terminate the string
printf("Read %zu bytes from the file using fread: %s\n", bytesRead, buffer);

// Closing the file
```

```
    fclose(file);
    printf("File closed successfully using fclose.\n");
}

int main() {
    printf("Demonstrating file operations using system calls:\n");
    file_operations_system_calls();
    printf("\nDemonstrating file operations using C library functions:\n");
    file_operations_c_library();
    return 0;
}
```

<u>Output:</u>

```
Demonstrating file operations using C library functions:
File created successfully using fopen.
Written in the file using fwrite [Hello, World!].
File closed successfully using fclose.
File opened successfully for reading using fopen.
Read from the file using fread: Hello, World!
File closed successfully using fclose.
```

ASSIGNMENT – 2

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

Program:

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_SIZE 100

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

    printf("Array elements: ");

    for (int i = 0; i < size; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

void insert(int arr[], int *size, int element, int index) {

    if (*size >= MAX_SIZE) {

        printf("Array is full, cannot insert new element.\n");

        return;

    }

    if (index < 0 || index > *size) {

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

        return;

    }
    for (int i = *size; i > index; i--) {
        arr[i] = arr[i - 1]; // Shift elements to the right
    }
    arr[index] = element; // Insert the element
    (*size)++;
}
void delete(int arr[], int *size, int index) {
    if (index < 0 || index >= *size) {
        printf("Index out of bounds.\n");
        return;
    }
    for (int i = index; i < *size - 1; i++) {
        arr[i] = arr[i + 1]; // Shift elements to the left
    }
    (*size)--; // Decrease the size
}
int search(int arr[], int size, int element) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == element) {
            return i; // Return the index
        }
    }
    return -1; // Element not found
```

```
}
int main() {
    int arr[MAX_SIZE];
    int size = 0; // Current size of the array
    // Insert elements
    insert(arr, &size, 10, 0);
    insert(arr, &size, 20, 1);
    insert(arr, &size, 30, 2);
    insert(arr, &size, 25, 2); // Inserting 25 at index 2
    display(arr, size);
    // Delete an element
    delete(arr, &size, 1); // Deleting element at index 1
    display(arr, size);
    // Search for an element
    int element = 25;
    int index = search(arr, size, element);
    if (index != -1) {
        printf("Element %d found at index %d.\n", element, index);
    } else {
        printf("Element %d not found in the array.\n", element);
    }
      // Search for a non-existent element
    element = 20;
    index = search(arr, size, element);
    if (index != -1) {
        printf("Element %d found at index %d.\n", element, index);
    } else {
        printf("Element %d not found in the array.\n", element);
    }
    return 0;
}
```

Output:

```
Array elements: 10 20 25 30
Array elements: 10 25 30
Element 25 found at index 1.
Element 20 not found in the array.
```

ASSIGNMENT – 3

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

Program:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import org.apache.poi.ss.usermodel.*;

public class ExcelReader {
    public static void main(String[] args) {
        try {
            // Replace "your_excel_file.xlsx" with the actual file path
            FileInputStream file = new FileInputStream(new File("your_excel_file.xlsx"));

            // Create a Workbook object
            Workbook workbook = WorkbookFactory.create(file);

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

            // Iterate over rows and columns
            for (Row row : sheet) {
                for (Cell cell : row) {
                    // Get the cell type and value
                    CellType cellType = cell.getCellType();
                    String cellValue = "";

                    switch (cellType) {
                        case STRING:
                            cellValue = cell.getStringCellValue();
                            break;
                        case NUMERIC:
                            cellValue = String.valueOf(cell.getNumericCellValue());
                            break;
                        case BOOLEAN:
                            cellValue = String.valueOf(cell.getBooleanCellValue());
                            break;
                        case FORMULA:
                            cellValue = cell.getCellFormula();
                            break;
                        default:
                            cellValue = "Unknown";
                    }

                    // Print the cell value
                    System.out.print(cellValue + "\t");
                }
                System.out.println();
            }

            // Close the workbook
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

<u>OUTPUT:</u>

Assume the example.xlsx file has the following content:

Name	Age	City
Anusha	20	Hubli
Manasa	18	Gadag
Sharat	18	Bagalkot

After we run the program:

```
Name        Age    City
```

```
Anusha        20.0    Hubli
Manasa       18.0     Gadag
Sharat          18.0      Bagalkot
```