praveenkumar.R
192324012

Experient:14-Construct a C program to organise the file using a single level directory.

Aim:

The aim of this program is to demonstrate how to organize files using a **single-level directory structure** in C. A single-level directory is a simple file structure in which all files are stored directly in one directory, with no subdirectories. This program will create a directory, create some files within it, and then list the files contained in that directory.

Procedure:

1. **Create a Directory**: The program will create a directory using mkdir() system call.
2. **Create Files in the Directory**: We will create files inside the directory using file operations like fopen().
3. **List Files in the Directory**: The program will use opendir() and readdir() system calls to list the contents of the directory.
4. **Error Handling**: Proper error handling will be implemented to ensure the creation of the directory and files, and listing of files occurs successfully.

Steps Involved:

1. **Create a Directory**: Use the mkdir() function to create a directory where files will be stored.
2. **Create Files**: Use fopen() to create files inside the directory.
3. **List Files**: Use opendir() and readdir() to list the contents of the directory.
4. **Clean Up**: Close files and directories properly.

C Program to Organize Files Using a Single-Level Directory:

c

Copy code

```
#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>


#define DIRECTORY_NAME "single_level_directory"


// Function to create a directory
```

```
void create_directory(const char *dir_name) {
    if (mkdir(dir_name, 0755) == -1) {
        perror("Error creating directory");
        exit(EXIT_FAILURE);
    } else {
        printf("Directory '%s' created successfully.\n", dir_name);
    }
}

// Function to create a file inside the directory
void create_file(const char *dir_name, const char *file_name) {
    char path[256];
    FILE *file;

    // Construct the full path for the file
    snprintf(path, sizeof(path), "%s/%s", dir_name, file_name);

    // Create and open the file for writing
    file = fopen(path, "w");
    if (file == NULL) {
        perror("Error creating file");
        exit(EXIT_FAILURE);
    }

    fprintf(file, "This is file: %s\n", file_name);
    fclose(file);
    printf("File '%s' created successfully in directory '%s'.\n", file_name, dir_name);
}

// Function to list the files in the directory
void list_files(const char *dir_name) {
```

```
    DIR *dir;
    struct dirent *entry;

    // Open the directory
    dir = opendir(dir_name);
    if (dir == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    printf("\nListing files in directory '%s':\n", dir_name);

    // Read and list the files in the directory
    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_REG) {
            printf("%s\n", entry->d_name);
        }
    }

    closedir(dir);
}

int main() {
    // Create a directory
    create_directory(DIRECTORY_NAME);

    // Create files in the directory
    create_file(DIRECTORY_NAME, "file1.txt");
    create_file(DIRECTORY_NAME, "file2.txt");
    create_file(DIRECTORY_NAME, "file3.txt");
```

```
    // List the files in the directory
    list_files(DIRECTORY_NAME);

    return 0;
}
```

Output: