**Experiment:15-Design a C program to organise the file using a two level directory structure.**

Aim:

The aim of this program is to organize files using a two-level directory structure in C. A two-level directory structure is a hierarchical system where a root directory contains subdirectories, and each subdirectory contains files. This program will:

1. Create a root directory.
2. Create subdirectories within the root directory.
3. Create files inside the subdirectories.
4. List the files in the subdirectories.

Procedure:

1. Create Root Directory: The program will create a root directory.
2. Create Subdirectories: Create subdirectories under the root directory.
3. Create Files in Subdirectories: Files will be created inside these subdirectories.
4. List Files in Subdirectories: The program will list all files present in the subdirectories.
5. Error Handling: Proper error handling will be performed to ensure that each step, such as directory creation and file creation, is successful.

Steps Involved:

1. Create Root Directory: Use mkdir() to create the root directory.
2. Create Subdirectories: Similarly, use mkdir() to create subdirectories inside the root directory.
3. Create Files in Subdirectories: Use fopen() to create files in these subdirectories.
4. List Files in Subdirectories: Use opendir() and readdir() to list the contents of subdirectories.

C Program to Organize Files Using a Two-Level Directory Structure:

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#define ROOT\_DIRECTORY "root\_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 a 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 a 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) { // Regular file

printf("%s\n", entry->d\_name);

}

}

closedir(dir);

}

int main() {

// Create the root directory

create\_directory(ROOT\_DIRECTORY);

// Subdirectory names

const char \*subdirs[] = {"subdir1", "subdir2", "subdir3"};

// Create subdirectories within the root directory

for (int i = 0; i < 3; i++) {

char subdir\_path[256];

snprintf(subdir\_path, sizeof(subdir\_path), "%s/%s", ROOT\_DIRECTORY, subdirs[i]);

create\_directory(subdir\_path);

// Create files inside each subdirectory

for (int j = 1; j <= 3; j++) {

char file\_name[256];

snprintf(file\_name, sizeof(file\_name), "file%d.txt", j);

create\_file(subdir\_path, file\_name);

}

// List files inside each subdirectory

list\_files(subdir\_path);

}

return 0;

}

**Output:**

****