NAME: G. GANESH

REG NO: 192373008

EXERICSE-15

Design a C program to organise the file using a two level directory structure.

Aim:

To design a C program that simulates the **Two-Level Directory** structure to manage files.

Algorithm:

1. **Initialize Directory Structure:**
    - Create structures for users and their files.
    - Define an array of users, each with their file list.
2. **Menu Options:**
    - Provide user actions to:
        1. Add a user.
        2. Add a file to a user.
        3. Delete a file from a user.
        4. Search for a file under a user.
        5. Display all users and their files.
        6. Exit.
3. **File Operations:**
    - **AddUser:**
      Create a new user if not already present.
    - **AddFile:**
      Add a file under the selected user if it doesn't already exist.
    - **DeleteFile:**
      Remove the specified file from the user if it exists.
    - **SearchFile:**
      Search for a file under a specific user.
    - **DisplayUsersandFiles:**
      Show all users with their associated files.
4. **Exit Program:**
    - End when the user selects the exit option.

Code:

```
#include <stdio.h>
#include <string.h>
#define MAX_USERS 10
#define MAX_FILES 10
#define MAX_NAME_LENGTH 50
typedef struct {
    char name[MAX_NAME_LENGTH];
} File;
typedef struct {
    char username[MAX_NAME_LENGTH];
    File files[MAX_FILES];
    int file_count;
} User;
void add_user(User users[], int *user_count) {
    if (*user_count >= MAX_USERS) {
        printf("Maximum user limit reached.\n");
        return;
    }
    char username[MAX_NAME_LENGTH];
    printf("Enter username: ");
    scanf("%s", username);
    for (int i = 0; i < *user_count; i++) {
        if (strcmp(users[i].username, username) == 0) {
            printf("User '%s' already exists.\n", username);
            return;
        }
    }
    strcpy(users[*user_count].username, username);
    users[*user_count].file_count = 0;
```

```
    (*user_count)++;
    printf("User '%s' added successfully.\n", username);
}
void add_file(User users[], int user_count) {
    char username[MAX_NAME_LENGTH], file_name[MAX_NAME_LENGTH];
    printf("Enter username: ");
    scanf("%s", username);
    for (int i = 0; i < user_count; i++) {
        if (strcmp(users[i].username, username) == 0) {
            if (users[i].file_count >= MAX_FILES) {
                printf("File limit for user '%s' reached.\n", username);
                return;
            }
            printf("Enter file name: ");
            scanf("%s", file_name);
            for (int j = 0; j < users[i].file_count; j++) {
                if (strcmp(users[i].files[j].name, file_name) == 0) {
                    printf("File '%s' already exists for user '%s'.\n", file_name, username);
                    return;
                }
            }
            strcpy(users[i].files[users[i].file_count].name, file_name);
            users[i].file_count++;
            printf("File '%s' added successfully for user '%s'.\n", file_name, username);
            return;
        }
    }
    printf("User '%s' not found.\n", username);
}
void delete_file(User users[], int user_count) {
```

```
    char username[MAX_NAME_LENGTH], file_name[MAX_NAME_LENGTH];
    printf("Enter username: ");
    scanf("%s", username);
    for (int i = 0; i < user_count; i++) {
        if (strcmp(users[i].username, username) == 0) {
            printf("Enter file name to delete: ");
            scanf("%s", file_name);
            for (int j = 0; j < users[i].file_count; j++) {
                if (strcmp(users[i].files[j].name, file_name) == 0) {
                    for (int k = j; k < users[i].file_count - 1; k++) {
                        users[i].files[k] = users[i].files[k + 1];
                    }
                    users[i].file_count--;
                    printf("File '%s' deleted successfully for user '%s'.\n", file_name, username);
                    return;
                }
            }
            printf("File '%s' not found for user '%s'.\n", file_name, username);
            return;
        }
    }
    printf("User '%s' not found.\n", username);
}
void search_file(User users[], int user_count) {
    char username[MAX_NAME_LENGTH], file_name[MAX_NAME_LENGTH];
    printf("Enter username: ");
    scanf("%s", username);
    for (int i = 0; i < user_count; i++) {
        if (strcmp(users[i].username, username) == 0) {
            printf("Enter file name to search: ");
```

```
            scanf("%s", file_name);
            for (int j = 0; j < users[i].file_count; j++) {
                if (strcmp(users[i].files[j].name, file_name) == 0) {
                    printf("File '%s' found for user '%s'.\n", file_name, username);
                    return;
                }
            }
            printf("File '%s' not found for user '%s'.\n", file_name, username);
            return;
        }
    }
    printf("User '%s' not found.\n", username);
}
void display_users(User users[], int user_count) {
    if (user_count == 0) {
        printf("No users in the system.\n");
        return;
    }
    printf("Users and their files:\n");
    for (int i = 0; i < user_count; i++) {
        printf("User: %s\n", users[i].username);
        if (users[i].file_count == 0) {
            printf("  No files.\n");
        } else {
            for (int j = 0; j < users[i].file_count; j++) {
                printf("  File %d: %s\n", j + 1, users[i].files[j].name);
            }
        }
    }
}
```

```
int main() {
    User users[MAX_USERS];
    int user_count = 0, choice;
    do {
        printf("\nTwo-Level Directory Structure\n");
        printf("1. Add User\n");
        printf("2. Add File\n");
        printf("3. Delete File\n");
        printf("4. Search File\n");
        printf("5. Display Users and Files\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                add_user(users, &user_count);
                break;
            case 2:
                add_file(users, user_count);
                break;
            case 3:
                delete_file(users, user_count);
                break;
            case 4:
                search_file(users, user_count);
                break;
            case 5:
                display_users(users, user_count);
                break;
            case 6:
```

```
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice! Try again.\n");
    }
  } while (choice != 6);
  return 0;
}
```

Result:

The program successfully simulates a two-level directory structure, allowing user and file management operations.

Output: