B. Siva Shirish-192324016

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

AIM:

To design a C program that organizes files using a two-level directory structure.

ALGORITHM:

1. Initialize the directory structure with a maximum number of directories and files.
2. Define functions to create directories and files, delete files, display files in a directory, and search for files within a directory.
3. Implement user interaction through a menu to allow creating directories, adding/removing files, displaying files, and searching for files.

PROCEDURE:

1. Define the Directory structure with an array of files.
2. Define functions for managing directories and files:
    - create_directory() to create a new directory.
    - create_file_in_directory() to add files to an existing directory.
    - delete_file_from_directory() to remove files from a directory.
    - display_files_in_directory() to display the list of files in a directory.
    - search_file_in_directory() to search for a file in a directory.
3. Use a loop to present a menu to the user for interaction.
4. Allow the user to create directories, add files, delete files, display files, and search files.

CODE:

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

#define MAX_USERS 10
#define MAX_FILES 10
#define MAX_NAME_LENGTH 50

typedef struct {
   char fileName[MAX_NAME_LENGTH];
   int isOccupied; // 1 if file exists, 0 otherwise
} File;
```

```
typedef struct {
   char userName[MAX_NAME_LENGTH];
   File files[MAX_FILES];
   int fileCount; // Number of files in the user's directory
} UserDirectory;

UserDirectory userDirectories[MAX_USERS];
int userCount = 0;

// Function to find a user by name
int findUser(char userName[]) {
   for (int i = 0; i < userCount; i++) {
      if (strcmp(userDirectories[i].userName, userName) == 0) {
         return i;
      }
   }
   return -1;
}

// Add a new user
void addUser() {
   if (userCount >= MAX_USERS) {
      printf("Maximum user limit reached. Cannot add more users.\n");
      return;
   }

   char userName[MAX_NAME_LENGTH];
   printf("Enter the user name: ");
   scanf("%s", userName);

   if (findUser(userName) != -1) {
      printf("User '%s' already exists.\n", userName);
      return;
   }

   strcpy(userDirectories[userCount].userName, userName);
   userDirectories[userCount].fileCount = 0;
   for (int i = 0; i < MAX_FILES; i++) {
      userDirectories[userCount].files[i].isOccupied = 0;
   }
   userCount++;
   printf("User '%s' added successfully.\n", userName);
}

// Add a file to a user's directory
void addFile() {
   char userName[MAX_NAME_LENGTH], fileName[MAX_NAME_LENGTH];
   printf("Enter the user name: ");
   scanf("%s", userName);

   int userIndex = findUser(userName);
```

```
    if (userIndex == -1) {
        printf("User '%s' does not exist.\n", userName);
        return;
    }

    if (userDirectories[userIndex].fileCount >= MAX_FILES) {
        printf("User's directory is full. Cannot add more files.\n");
        return;
    }

    printf("Enter the file name to add: ");
    scanf("%s", fileName);

    for (int i = 0; i < MAX_FILES; i++) {
        if (userDirectories[userIndex].files[i].isOccupied &&
            strcmp(userDirectories[userIndex].files[i].fileName, fileName) == 0) {
            printf("File '%s' already exists in '%s's directory.\n", fileName, userName);
            return;
        }
    }

    for (int i = 0; i < MAX_FILES; i++) {
        if (!userDirectories[userIndex].files[i].isOccupied) {
            strcpy(userDirectories[userIndex].files[i].fileName, fileName);
            userDirectories[userIndex].files[i].isOccupied = 1;
            userDirectories[userIndex].fileCount++;
            printf("File '%s' added successfully to '%s's directory.\n", fileName, userName);
            return;
        }
    }
}

// Search for a file in a user's directory
void searchFile() {
    char userName[MAX_NAME_LENGTH], fileName[MAX_NAME_LENGTH];
    printf("Enter the user name: ");
    scanf("%s", userName);

    int userIndex = findUser(userName);
    if (userIndex == -1) {
        printf("User '%s' does not exist.\n", userName);
        return;
    }

    printf("Enter the file name to search: ");
    scanf("%s", fileName);

    for (int i = 0; i < MAX_FILES; i++) {
        if (userDirectories[userIndex].files[i].isOccupied &&
            strcmp(userDirectories[userIndex].files[i].fileName, fileName) == 0) {
```

```
            printf("File '%s' found in '%s's directory.\n", fileName, userName);
            return;
        }
    }
    printf("File '%s' not found in '%s's directory.\n", fileName, userName);
}

// List all files in a user's directory
void listFiles() {
    char userName[MAX_NAME_LENGTH];
    printf("Enter the user name: ");
    scanf("%s", userName);

    int userIndex = findUser(userName);
    if (userIndex == -1) {
        printf("User '%s' does not exist.\n", userName);
        return;
    }

    printf("\nFiles in '%s's directory:\n", userName);
    if (userDirectories[userIndex].fileCount == 0) {
        printf("No files in the directory.\n");
    } else {
        for (int i = 0; i < MAX_FILES; i++) {
            if (userDirectories[userIndex].files[i].isOccupied) {
                printf("%s\n", userDirectories[userIndex].files[i].fileName);
            }
        }
    }
}

int main() {
    int choice;

    while (1) {
        printf("\nTwo-Level Directory Management System\n");
        printf("1. Add User\n");
        printf("2. Add File to User Directory\n");
        printf("3. Search File in User Directory\n");
        printf("4. List Files in User Directory\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addUser();
                break;
            case 2:
                addFile();
                break;
```

```
        case 3:
            searchFile();
            break;
        case 4:
            listFiles();
            break;
        case 5:
            printf("Exiting the program.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  }
}
```

OUTPUT: