192311035

M.RAVI SAI VINAY

EXPERIMENT-15

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

AIM:-

To design a C program to organize files using a two-level directory structure.

ALGORITHM:-

1. **Define Structures:**
    - Create a structure for the main directory, which contains subdirectories.
    - Each subdirectory contains a list of files and its name.
2. **Menu Options:**
    - Provide options to create subdirectories, add files, delete files, search files, and display the directory structure.
3. **Add Subdirectory:**
    - Prompt the user to enter the subdirectory name and create it.
4. **Add Files:**
    - Add files to a specific subdirectory if it exists.
5. **Delete Files:**
    - Search for the file in the specified subdirectory and delete it.
6. **Search Files:**
    - Check if a given file exists in the subdirectory.
7. **Display Structure:**
    - Print all subdirectories and their respective files.

CODE:-

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

#define MAX_SUBDIR 10
#define MAX_FILES 10

struct SubDirectory {
    char name[50];
    char files[MAX_FILES][50];
    int fileCount;
};

struct Directory {
    struct SubDirectory subDirs[MAX_SUBDIR];
    int subDirCount;
};

void addSubDirectory(struct Directory* dir, char* subDirName) {
    if (dir->subDirCount < MAX_SUBDIR) {
        strcpy(dir->subDirs[dir->subDirCount].name, subDirName);
        dir->subDirs[dir->subDirCount].fileCount = 0;
        dir->subDirCount++;
        printf("Subdirectory '%s' added successfully.\n", subDirName);
    } else {
```

```
        printf("Cannot add more subdirectories.\n");
    }
}

void addFile(struct Directory* dir, char* subDirName, char* fileName) {
    for (int i = 0; i < dir->subDirCount; i++) {
        if (strcmp(dir->subDirs[i].name, subDirName) == 0) {
            if (dir->subDirs[i].fileCount < MAX_FILES) {
                strcpy(dir->subDirs[i].files[dir->subDirs[i].fileCount], fileName);
                dir->subDirs[i].fileCount++;
                printf("File '%s' added to subdirectory '%s'.\n", fileName, subDirName);
            } else {
                printf("Subdirectory '%s' is full. Cannot add more files.\n", subDirName);
            }
            return;
        }
    }
    printf("Subdirectory '%s' not found.\n", subDirName);
}

void deleteFile(struct Directory* dir, char* subDirName, char* fileName) {
    for (int i = 0; i < dir->subDirCount; i++) {
        if (strcmp(dir->subDirs[i].name, subDirName) == 0) {
            int found = 0;
```

```
            for (int j = 0; j < dir->subDirs[i].fileCount; j++) {
                if (strcmp(dir->subDirs[i].files[j], fileName) == 0) {
                    found = 1;
                    for (int k = j; k < dir->subDirs[i].fileCount - 1; k++) {
                        strcpy(dir->subDirs[i].files[k], dir->subDirs[i].files[k + 1]);
                    }
                    dir->subDirs[i].fileCount--;
                    printf("File '%s' deleted from subdirectory '%s'.\n", fileName, subDirName);
                    return;
                }
            }
            if (!found) {
                printf("File '%s' not found in subdirectory '%s'.\n", fileName, subDirName);
            }
            return;
        }
    }
    printf("Subdirectory '%s' not found.\n", subDirName);
}

void searchFile(struct Directory* dir, char* subDirName, char* fileName) {
    for (int i = 0; i < dir->subDirCount; i++) {
        if (strcmp(dir->subDirs[i].name, subDirName) == 0) {
            for (int j = 0; j < dir->subDirs[i].fileCount; j++) {
```

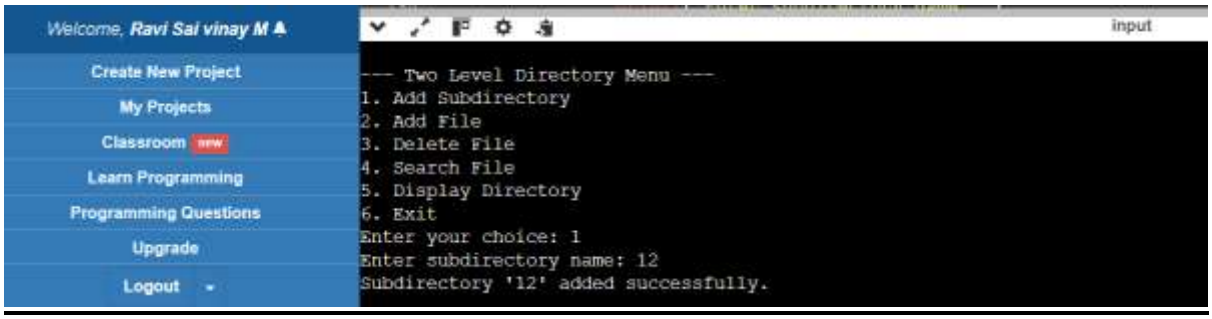
```
            if (strcmp(dir->subDirs[i].files[j], fileName) == 0) {
                printf("File '%s' found in subdirectory '%s'.\n", fileName, subDirName);
                return;
            }
        }
        printf("File '%s' not found in subdirectory '%s'.\n", fileName, subDirName);
        return;
    }
  }
  printf("Subdirectory '%s' not found.\n", subDirName);
}

void displayDirectory(struct Directory* dir) {
  if (dir->subDirCount == 0) {
    printf("Directory is empty.\n");
  } else {
    for (int i = 0; i < dir->subDirCount; i++) {
        printf("Subdirectory: %s\n", dir->subDirs[i].name);
        if (dir->subDirs[i].fileCount == 0) {
            printf("  No files in this subdirectory.\n");
        } else {
            for (int j = 0; j < dir->subDirs[i].fileCount; j++) {
                printf("  %d. %s\n", j + 1, dir->subDirs[i].files[j]);
            }
```

```
            }
        }
    }
}

int main() {
    struct Directory dir;
    dir.subDirCount = 0;

    int choice;
    char subDirName[50], fileName[50];

    do {
        printf("\n--- Two Level Directory Menu ---\n");
        printf("1. Add Subdirectory\n");
        printf("2. Add File\n");
        printf("3. Delete File\n");
        printf("4. Search File\n");
        printf("5. Display Directory\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
```

```
case 1:
    printf("Enter subdirectory name: ");
    scanf("%s", subDirName);
    addSubDirectory(&dir, subDirName);
    break;
case 2:
    printf("Enter subdirectory name: ");
    scanf("%s", subDirName);
    printf("Enter file name: ");
    scanf("%s", fileName);
    addFile(&dir, subDirName, fileName);
    break;
case 3:
    printf("Enter subdirectory name: ");
    scanf("%s", subDirName);
    printf("Enter file name: ");
    scanf("%s", fileName);
    deleteFile(&dir, subDirName, fileName);
    break;
case 4:
    printf("Enter subdirectory name: ");
    scanf("%s", subDirName);
    printf("Enter file name: ");
    scanf("%s", fileName);
```

```
            searchFile(&dir, subDirName, fileName);
            break;
        case 5:
            displayDirectory(&dir);
            break;
        case 6:
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 6);

  return 0;
}
```

OUTPUT:-



RESULT:-

The program successfully simulates a two-level directory structure. It supports the creation of subdirectories, adding and deleting files, searching for files, and displaying the directory hierarchy.