

D.GUNA VARDHAN - 192372041

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_FILES 10
```

```
#define MAX_DIRS 5

#define MAX_FILE_NAME 50

#define MAX_DIR_NAME 50
```

```
typedef struct {
    char file_name[MAX_FILE_NAME];
    int is_present;
} File;
```

```
typedef struct {
    char dir_name[MAX_DIR_NAME];
    File files[MAX_FILES];
} Directory;
```

```
Directory directories[MAX_DIRS];
```

```
void initialize_directory_structure() {
    for (int i = 0; i < MAX_DIRS; i++) {
        directories[i].dir_name[0] = '\0';
        for (int j = 0; j < MAX_FILES; j++) {
            directories[i].files[j].is_present = 0;
        }
    }
}
```

```
int create_directory(const char *dir_name) {
    for (int i = 0; i < MAX_DIRS; i++) {
        if (directories[i].dir_name[0] == '\0') {
```

```

        strncpy(directories[i].dir_name, dir_name, MAX_DIR_NAME);

        return 1;
    }
}

return 0;
}

int create_file_in_directory(const char *dir_name, const char *file_name) {
    for (int i = 0; i < MAX_DIRS; i++) {
        if (strcmp(directories[i].dir_name, dir_name) == 0) {
            for (int j = 0; j < MAX_FILES; j++) {
                if (directories[i].files[j].is_present == 0) {
                    strncpy(directories[i].files[j].file_name, file_name, MAX_FILE_NAME);

                    directories[i].files[j].is_present = 1;

                    return 1;
                }
            }
        }
    }

    return 0;
}

```

```

int delete_file_from_directory(const char *dir_name, const char *file_name) {
    for (int i = 0; i < MAX_DIRS; i++) {
        if (strcmp(directories[i].dir_name, dir_name) == 0) {
            for (int j = 0; j < MAX_FILES; j++) {
                if (directories[i].files[j].is_present == 1 && strcmp(directories[i].files[j].file_name,
file_name) == 0) {

```

```
        directories[i].files[j].is_present = 0;

        return 1;
    }
}

}

}

return 0;
}
```

```
void display_files_in_directory(const char *dir_name) {
    for (int i = 0; i < MAX_DIRS; i++) {
        if (strcmp(directories[i].dir_name, dir_name) == 0) {
            printf("Files in directory '%s':\n", dir_name);

            int found = 0;

            for (int j = 0; j < MAX_FILES; j++) {
                if (directories[i].files[j].is_present == 1) {
                    printf("%s\n", directories[i].files[j].file_name);

                    found = 1;
                }
            }

            if (!found) {
                printf("No files in this directory.\n");
            }
        }
    }
}
```

```
int search_file_in_directory(const char *dir_name, const char *file_name) {
```

```

for (int i = 0; i < MAX_DIRS; i++) {
    if (strcmp(directories[i].dir_name, dir_name) == 0) {
        for (int j = 0; j < MAX_FILES; j++) {
            if (directories[i].files[j].is_present == 1 && strcmp(directories[i].files[j].file_name,
file_name) == 0) {
                return 1;
            }
        }
    }
}
return 0;
}

int main() {
    int choice;

    char dir_name[MAX_DIR_NAME], file_name[MAX_FILE_NAME];

    initialize_directory_structure();

    while (1) {
        printf("\nMenu:\n");

        printf("1. Create a directory\n");

        printf("2. Create a file in a directory\n");

        printf("3. Delete a file from a directory\n");

        printf("4. Display files in a directory\n");

        printf("5. Search for a file in a directory\n");

        printf("6. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        getchar();

        switch (choice) {

```

case 1:

```
printf("Enter directory name to create: ");
fgets(dir_name, MAX_DIR_NAME, stdin);
dir_name[strcspn(dir_name, "\n")] = '\0';
if (create_directory(dir_name)) {
    printf("Directory '%s' created successfully.\n", dir_name);
} else {
    printf("No space for new directories.\n");
}
break;
```

case 2:

```
printf("Enter directory name to create file in: ");
fgets(dir_name, MAX_DIR_NAME, stdin);
dir_name[strcspn(dir_name, "\n")] = '\0';
printf("Enter file name to create: ");
fgets(file_name, MAX_FILE_NAME, stdin);
file_name[strcspn(file_name, "\n")] = '\0';
if (create_file_in_directory(dir_name, file_name)) {
    printf("File '%s' created in directory '%s'.\n", file_name, dir_name);
} else {
    printf("Directory not found or directory is full.\n");
}
break;
```

case 3:

```
printf("Enter directory name to delete file from: ");
fgets(dir_name, MAX_DIR_NAME, stdin);
dir_name[strcspn(dir_name, "\n")] = '\0';
printf("Enter file name to delete: ");
```

```
fgets(file_name, MAX_FILE_NAME, stdin);  
file_name[strcspn(file_name, "\n")] = '\0';  
if (delete_file_from_directory(dir_name, file_name)) {  
    printf("File '%s' deleted from directory '%s'.\n", file_name, dir_name);  
} else {  
    printf("File not found in directory '%s'.\n", dir_name);  
}  
break;
```

case 4:

```
printf("Enter directory name to display files: ");  
fgets(dir_name, MAX_DIR_NAME, stdin);  
dir_name[strcspn(dir_name, "\n")] = '\0';  
display_files_in_directory(dir_name);  
break;
```

case 5:

```
printf("Enter directory name to search for file: ");  
fgets(dir_name, MAX_DIR_NAME, stdin);  
dir_name[strcspn(dir_name, "\n")] = '\0';  
printf("Enter file name to search: ");  
fgets(file_name, MAX_FILE_NAME, stdin);  
file_name[strcspn(file_name, "\n")] = '\0';  
if (search_file_in_directory(dir_name, file_name)) {  
    printf("File '%s' found in directory '%s'.\n", file_name, dir_name);  
} else {  
    printf("File '%s' not found in directory '%s'.\n", file_name, dir_name);  
}  
break;
```

case 6:

```

    printf("Exiting the program.\n");

    return 0;

default:

    printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

```

OUTPUT:

```

OnlineGDB
online compiler and debugger for c/c++

Welcome, GUNA VARDHAN

Create New Project
My Projects
Classroom new
Learn Programming
Programing Questions
Upgrade
Logout

Menu:
1. Create a directory
2. Create a file in a directory
3. Delete a file from a directory
4. Display files in a directory
5. Search for a file in a directory
6. Exit
Enter your choice: 1
Enter directory name to create: TEJA
Directory 'TEJA' created successfully.

Menu:
1. Create a directory
2. Create a file in a directory
3. Delete a file from a directory
4. Display files in a directory
5. Search for a file in a directory
6. Exit
Enter your choice: 4
Enter directory name to display files: 6

Menu:
1. Create a directory
2. Create a file in a directory
3. Delete a file from a directory
4. Display files in a directory
5. Search for a file in a directory
6. Exit
Enter your choice: 6
Exiting the program.

...Program finished with exit code 0
Press ENTER to exit console.

```